UNIVERSITY
OF OSLO

Master's thesis

LARA: An automated web-based tool for privacy risk assessments

What is required for streamlining modular DPIAs

Katrine Feten

Informatics: Programming and System Architecture
60 ECTS study points

Department of Informatics
Faculty of Mathematics and Natural Sciences

Spring 2023

Katrine Feten

LARA: An automated web-based tool for privacy risk assessments

What is required for streamlining modular DPIAs

Supervisors:
Shukun Tokas
Hui Song
Ketil Stølen

Abstract

The implementation of the GDPR brought about the legal requirement of performing DPIAs on software systems. Consequently, many templates for performing DPIAs are available for use. However, common for most of them is that they lack a way to easily re-assess the system without starting from the beginning. This unmet need is the motivation for our thesis. We aim to design a digital tool which is able to streamline the privacy risk assessment, with the overarching goal of exploring what is needed to achieve this. To this end, we identify the needs of such a tool, formulate a set of requirements based on those needs, implement these requirements into the tool, and evaluate our implementation. As the time available is limited, we base the tool, which we name LARA, on a version of the LINDDUN privacy threat modeling methodology. We identified the needs based on the target group, which is those who require the use of a DPIA. Two evaluation methods were used to assess LARA. The first was a case study consisting of two use cases and a comparison to a similar, existing tool. LARA was evaluated by performing a limited risk assessment, then a re-assessment using the results of the first use case, then compared to a digital tool with a LINDDUN implementation. The second evaluation was of the intuitiveness of LARA through usability testing. Our findings based on these evaluations leads us to believe that LARA is able to increase the ease of performance for the privacy risk assessment through the automatization of LINDDUN's knowledge base. Based on the experience gained through the work, we conclude that improvements in the knowledge base, with the addition of a knowledge base for threat scenarios, is a requirement for further automatization. However, due to the expertise requirements of a tester for such a tool, we cannot guarantee the accuracy of our results. Finally, we present the recommendations for further improvements on the subject.

Keywords: DPIA, LINDDUN, privacy risk assessment, tool development, DPIA automation, streamline, threat refinement

Acknowledgements

I would like to thank my supervisors for their patience and assistance with completing this thesis. They always came through when I was stuck on what to do next, and always assured me when I was doubting myself.

I would like to thank my parents and brother, for their support over the years.

I am also thankful to my friends, who have been with me through my studies, for being wonderful and supporting people.

Finally, I wish to thank my recently departed dog Charlie, for his loving companionship of twelve years.

Contents

Chapter 1

Introduction

As Internet adoption became more widespread [27], our day-to-day lives have become significantly impacted by the digital technology. From the technophiles to the technophobes, it is quite rare to find a person in the western world who do not engage with the technology [23]. In the public sector, people can access services, such as healthcare, education, and tax filing, through online portals. Likewise, in the private sector, people can connect with businesses for various purposes, such as online shopping, banking, and entertainment. Digital technology has fostered stronger connections between people and various organizations, promoting convenience and efficiency in our daily lives.

As more people use digital services, more data is created. The question that arises is how the organizations and companies that collect this data will use it. Due to this massive amount of data collection, users are entitled to know what data about them are collected and used for. For this reason, entities who processes personal or sensitive data are required by the law to provide a privacy policy to their users [22]. However, as privacy policies have the unfortunate tendency to use advanced legal language and possess a great deal of text, users either fail to understand what they agree to, or skip reading the agreement altogether. Additionally, data is something which transcends country borders, so with each country deciding their own laws regarding how data will handled, it became difficult to enforce, allowing technology companies free reign to do as they pleased.

As privacy became a greater issue, so have the laws that govern how

data is to be collected, processed, stored and shared become stricter and more thorough. Culminating in May 2018, the GDPR (General Data Protection Regulation), which protects the rights of natural persons regarding the processing of personal data, became applicable in all member states of the European Union (EU) [10]. The GDPR is applicable to organizations established in the EU as well as non-EU established organizations, when personal data of individuals within the EU is being processed. Due to its comprehensive scope, strict requirements, and emphasis on privacy rights, GDPR is often considered the gold standard for privacy regulations.

The GDPR requires that, when the processing is likely to result in a high risk to natural persons, an assessment is required to evaluate what impact the intended processing would have. Such an assessment is called a Data Privacy Impact Assessment (DPIA).

The GDPR levies harsh fines on organizations that breach it. An example of such fines was given from the Dutch Supervisory Authority for Data Protection to the Dutch Social Insurance Institution (SVB) on January 19th 2023, with a fine on 150,000 euro for violating article 32(1)(2). The SVB suffered a data breach due to the system verifying identity being inadequate and the verification questions being too simple [2]. Similarly, Italian Data Protection Authority gave Eurosanità S.P.A. a fine on 120,000 euro for violating article 5, 9 and 32 on December 15th 2022, when they mistakenly provided an individual with the medical records of someone else [13]. In Norway, the toll company Ferde AS received a fine of 496,000 euro on September 27th 2021 from the Norwegian Supervisory Authority (Datatilsynet) for violating article 5(1f), 5(2), 28(3), 32(2) and 44. Ferde AS transferred license plates to China without performing a DPIA, as well as lacking a proper processor contract, thus making the data transfer take place without a valid legal basis [4]. These cases demonstrate that conducting a DPIA could have prevented the fines, which goes to show the importance of organizations and companies, especially smaller ones, being able to perform such assessments with ease.

Various templates for performing DPIAs have been created, including ones from ENISA [11], DistriNet [5], NIST [18], CNIL [3] and ICO [15]. Common for many of these examples are that they are defined for manually analysing the system once, which aligns with the waterfall development method [1]. None of these methodologies offer a way of addressing the incremental changes in system requirements, which is a unique aspect of software development, and that has contributed to the widespread adoption of agile

development [38]. Out of the methodologies listed here, DistriNet's methodology, LINDDUN, is the one which shows the greatest promise towards this premise. As LINDDUN is the most flexible methodology in its initial stages regarding inputted information, it contains the potential to handle incremental change.

This thesis aims to explore and address the lack of incrementality in the impact assessment process by developing a DPIA tool. Due to the time constraints for the thesis, we base the tool on a version of the LINDDUN privacy threat modeling methodology [5]. The objective is to streamline the time and effort required for conducting a DPIA, particularly during reassessments that build upon prior analyses.

Whereas the LINDDUN team discusses a similar topic in this paper [37], citing improvements in the semantics of the knowledge base as the core requirement for streamlining the privacy threat modeling process, we seek to explore other requirements for a tool-based approach, based on the practical experience of developing it.

The main outcome of this work is *LARA*, the **L**INDDUN-based **A**utomated **R**isk impact **A**ssessment tool. The main contributions this tool makes towards the goal of streamlining the process are the automated threat refinement, the vulnerability exclusion selection of the parameter form, as well as the automated threat mapping. The source code of LARA can be found on GitHub [12]. As the repositoy is located on the University of Oslo (UiO)'s local GitHub service, it requires a UiO user to access.

In this context, iterability and incrementality refers to the ability of the privacy risk assessment to be applied continuously in a non-linear manner. *An iterative privacy risk assessment should be able to incorporate changes to the system requirements at all stages of the assessment process without requiring a full re-assessment.* Automatization refers to *an action requiring manual input or calculation, such as textual descriptions, being enabled through other means*, such as integrated search based on inputted parameters.

The main technical contribution lies in the threat refinement 6.1, where we created an algorithm which turns a data flow diagram into a traversable graph 6.1.1, and an algorithm for traversing multiple connected tree graphs from the leaf ends 6.1.3. LARA is evaluated with a case study 7.1.2 which tests most of its functionalities as well as the interaction between them. Usability testing 7.2 is used to evaluate the intuitiveness of the tool.

This thesis is organized into the following chapters:

2. Chapter 2 contains an introduction to the sections of GDPR pertaining to data privacy impact assessments, as well as a description of how the LINDDUN methodology is performed, and our decision in basing our tool on it.

3. Chapter 3 contains the thesis statement, which further defines the objective of the thesis.

4. Chapter 4 defines the target user and the requirements of LARA, how the solution is to be produced, and how the evaluation is to be performed.

5. Chapter 5 contains a description of the architecture of LARA, the component design, external code libraries, as well as the functional choices made.

6. Chapter 6 contains a description of the threat refinement, as well as other functionalities which were discarded.

7. Chapter 7 contains the evaluation of the tool, where it is compared to another existing LINDDUN-based tool, in addition to performing a use case study.

8. Chapter 8 discusses the results of the evaluation, as well as the overall result of our thesis aims.

9. Chapter 9 contains the conclusion of the thesis, as well as a section on the future work of LARA.

10. The end of the thesis 9.2 contains an appendix with textual descriptions of the components of the tool, the full use case results, as well as the usability test tasks.

Chapter 2

Background

The GDPR defines how the privacy of individuals are to be handled when their personal data are processed, as well as the method to be used to evaluate the handling of the data. This chapter will introduce the relevant sections of the GDPR relating to DPIAs, as well as an introduction to the LINDDUN privacy threat modeling methodology, which is used as the basis for this work.

2.1 GDPR

Privacy has a long history, though it only became a generally accepted right in more recent history [16]. The Charter of Fundamental Rights of the European Union from 2012 provides its citizens with the protection of personal data. Per article 8, everyone has the right to the protection of personal data concerning themselves, that such data is processed fairly for specific purposes and on a legitimate basis such as consent or law, that they have access to the collected data and the right to rectify it, and that an independent authority may control that the rules are followed [8].

The GDPR [10], which replaced the Data Protection Directive of 1995 [9] in 2018, defines core data protection concepts. Article 4 of GDPR covers the definitions used in the legislation.

Personal data

Article 4(1) defines personal data as "any information relating to an identified or identifiable natural person". Examples of personal data are direct identifiers, such as data of birth, address, contact number, or online identifiers such as an IP address or a cookie identifier, or other factors specific to physiology, genetic, culture, economy etc.

There is a distinction between personal data, and sensitive data. Article 9(1) defines this as data revealing racial or ethnic origins, political opinions, philosophical beliefs, union membership, genetic and biometric data, data related to health, and data related to a person's sexual orientation and sex life. Health data is defined in article 4(15) as personal data related to physical and mental health of a natural person, which includes all that may reveal information of a person's health, such as any health care services they may have received. All data considered sensitive requires specific processing conditions. Depending on the purpose, the condition met must be, amongst others explicit consent, national or EU law, or a collective agreement, that a person's vital interests are at stake, publicly available data, data required for legal claims, substantial public interest, or research purposes, on basis of EU or national law per article 9(2).

Data subject

In the GDPR, a data subject is not defined explicitly, but rather defined as a part of personal data in Article 4(1), namely "an identified or identifiable natural person". It is the individual the data is about.

Data controller

A data controller is defined as "a natural or legal person, public authority, agency or other body which, alone or jointly with others, determines the purposes and means of the processing of personal data" in Article 4(7). This is to say that the controller is the one who decides the collection and processing, amongst others, of the data, and that the responsibility for following the GDPR requirements lie with the controller.

Article 25 - Data protection by design and by default

Article 25 introduces the obligation of data protection by design and by default on controllers. This Article in particular expresses the shift in who is responsible for privacy, from the users who agree to the terms of the data

controller, to the data controller who must accommodate for privacy as a basis of their service.

Article 25(1) makes the requirement that the data controller is to implement what is considered appropriate technical and organisational measures into the processing. It does however provide some leeway based on risk, implementation costs The nature and scope are taken into consideration as well, making it more flexible while maintaining the primary requirement.

Article 25(2) also makes it a requirement that, by default, the controller is to ensure through appropriate technical and organisational measures that only the necessary personal data is processed. This is for each specified purpose, and extends to collection, processing, storing and accessibility, with the last being explicitly stated.

Article 35 - Data protection impact assessment

Article 35 introduces the necessity of performing a data protection impact assessment. Article 35(1) requires that, when the data processing is likely to "result in a high risk to the right and freedom of natural persons", the data controller must perform an assessment of the intended processing. The purpose of this assessment is to ensure that that the personal data is adequately protected.

Article 35(7), specifically (a-c) and (d), further defines what such an assessment is required to contain. It requires a systematic description of the processing operations, as well as the purpose for the processing, including legitimate interests, an assessment of the necessity and the proportionality of the processing in relation to the purpose described, and an assessment of the risks towards the data subject. Additionally, it is to include a description of the measures intended be used to address the risks, which includes safeguards, security measures, and mechanisms to ensure that personal data is adequately protected. Demonstrating compliance towards the GDPR is also required of the assessment.

Finally, article 35(11) requires that, when a change of the risk represented by the processing operations occurs, the controller must carry out a review to assess if the processing occurs according to the DPIA where it is necessary.

Many forms of templates for performing data protection impact assessments exists. In this thesis, we focus on one specific DPIA, the LINDDUN methodology.

2.2 LINDDUN - Privacy Threat Modeling

LINDDUN[5], which is an acronym of its privacy threat categories, is an example of an existing privacy risk assessment methodology, as well as the methodology that our tool, *LARA*, is based on.

The LINDDUN methodology is a privacy threat modeling methodology, not focusing on identifying threats specific for one domain, instead focusing its efforts at helping its users identify common threats and providing general methods for risk mitigation. The privacy threat categories LINDDUN uses are **L**inkability, **I**dentifiability, **N**on-Repudiation, **D**etectability, **D**isclosure of Information, **U**nawareness, and **N**on-Compliance.

On April 15th 2023, DistriNet Research Group released an updated version of the LINDDUN methodology [6]. Our work will be referencing the old version [5].

In this section, we will briefly explain the steps of the LINDDUN methodology, with a deeper explanation of the parts most relevant to the thesis.

2.2.1 The LINDDUN steps

The LINDDUN methodology consists of six steps, which are divided into two sections, a problem space, and a solution space [35].

Problem space

The problem space contains the following steps:

1. Defining the system to be assessed through a Data Flow Diagram (DFD).
2. Mapping the privacy threats to the data elements defined in the DFD by using predefined assumptions.
3. Refining the identified threats further by using the LINDDUN threat trees in tandem with the user's own assumptions in regards to which parts of the system is considered trustworthy.

These steps then result in *threat scenarios*, which contains a description of the scenario, the involved parties, the consequences, and what threat trees and vulnerabilities are involved in the scenario.

A DFD consists of four different data elements. The *entity element*, which represents an external entity such as a user or third party services. The *process element* represents a unit that operates on the data. The *data store element* represents a form of container for information. The *data flow element* represents a flow of data between two DFD elements.

The *LINDDUN threat trees* are directed tree graphs. The root of the tree is the resultant materialization of the threat, while each node downwards are the compound pre-conditions for the threat to occur, ending in the leaf nodes, which are the individual vulnerabilities which can be exploited.

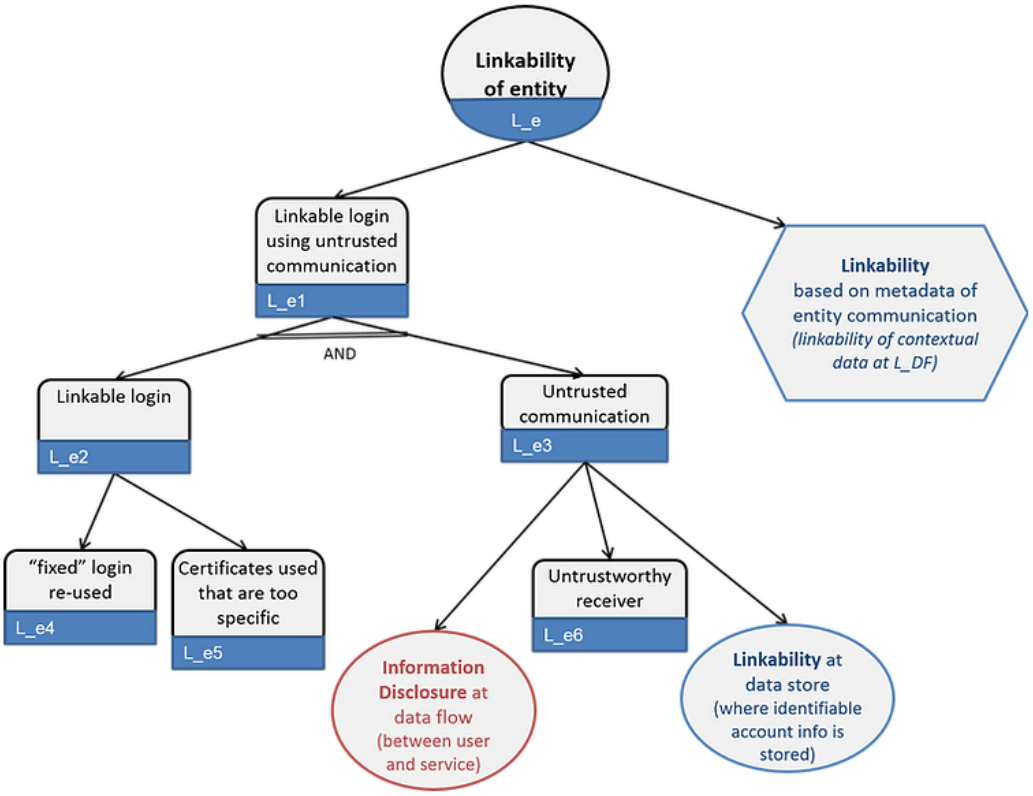

Figure 2.1: Example of a LINDDUN threat tree displaying the linkability threat for entity (from Wuyts et al. [36]).

A threat tree may refer to other threat trees. If the materialization of a threat is a pre-condition for a vulnerability, it is represented with an oval shape. If only part of the tree needs to be fulfilled to be a vulnerability, it is represented with a hexagonal shape.

We present an example using the left side of the tree. An entity is linkable if a linkable login is used together with untrusted communication. This occurs only if both a linkable login and untrusted communication is present. A linkable login can usually occur if either a fixed login is re-used, or if the certificates used are too specific. Untrusted communication may occur if either the receiver is untrustworthy, the information is disclosed while in transit from or to the user, or the data store is linkable where identifiable account info is stored.

The user can exclude both leaves and branches of the threat trees from their evaluations based on the assumptions made about the system. That there are only trustworthy receivers in the internal section of the system, or communication with an entity being securely implemented, are examples of such assumptions.

The current implementation of LARA only implements the disclosure of information threat trees. Any nodes which point to other threat trees concerning security have been omitted.

Solution space

The solution space consists of measures to be taken once the threats to the system have been identified.

1. The fourth step is to prioritize which threats are the most important to handle, considering the likelihood of occurrence and the impact it will have.

2. The fifth step is to apply pre-defined mitigation strategies to the vulnerabilities that are part of the threat scenario.

3. The sixth step is to find PETs (Privacy Enhancing Technologies) that correspond to the selected mitigation strategies.

The LINDDUN mitigation strategies are made to correspond to the nodes of the LINDDUN threat trees. As an example using the diagram, LINDDUN connects the node L_e , the materialized threat, to the strategy “Protect ID”. This strategy then introduces the use of pseudonyms, handling attributes and properties as the methods of achieving this [35].

2.2.2 LINDDUN as the basis of this thesis

The key distinction between LINDDUN and the DPIAs mentioned in chapter 1 is that LINDDUN uses a model-based approach, rather than the mostly descriptive approach used by many other methodologies. Additionally, LINDDUN provides its users with a systematic, knowledge-based approach for translating the various components of a system or architecture into relevant threats, refining those threats into specific vulnerabilities, and then translating those vulnerabilities into a mitigation strategies and solution.

LINDDUN's approach provides more information for us to work with, especially for the first two steps of the methodology. The basic system information is defined in a systematic manner which is feasible to sort and store in a digital tool. Additionally, updates to these steps should be easily implementable, which provides for a good starting point for the thesis. Furthermore, part of the core of the LINDDUN methodology, the threat refinement, is a task which is manually taxing, but one that we believe is possible to automatize to great effect when provided with assumptions.

With these favorable attributes present in the methodology, and with the consideration that we have a limited amount of time and resources available, we concluded that the LINDDUN methodology was the most feasible methodology to use as a basis to achieve the thesis objectives.

Chapter 3

Problem Definition

This thesis aims to develop an automated privacy risk assessment tool that enables a more streamlined risk assessment. The goal is to allow for real-time updates in response to changes in the privacy risk assessment model, as well as acquiring knowledge about the essential components needed for such a tool to be practical and effective.

To achieve this goal, this thesis will focus on the following specific questions:

1. To what degree we can develop a digital tool that streamlines the privacy impact assessment, drawing from an existing methodology?

2. What are the requirements for a privacy impact assessment methodology to become more streamlined in a digital tool?

While such a tool does not replace the need for a Data Protection Officer, it can serve as a valuable aid in assisting them with their tasks. It provides insight into the prevalent risks associated with certain processes and the available countermeasures, as well as guidance on their implementation within the system. Its goal is to promote privacy as a natural part of software development.

Chapter 4

Research Method

As the goal of LARA is to ease the process of performing a DPIA throughout the lifecycle of a system, we intended to base the design process on user-centered design [31]. As such, this chapter will discuss how we went about identifying the requirements for LARA, outlining our approach to developing the solution, and describing the planned evaluation method.

4.1 Identifying the needs of the tool

The first step of the design process is determining LARA's intended purposes in order to identify the requirements for such a tool. A DPIA is required when performing personal data processing activities. In addition, a re-assessment is required when changes in the processing risk occurs. As the current methodologies contain little support for re-assessment without performing the DPIA from the beginning, we have identified that those in need of an automated digital tool are those who require the usage of a DPIA.

One of the greatest issues with performing a DPIA is the complexity of the assessment, especially when factoring in the need for re-assessment. As such, *LARA should allow the user to perform the risk assessment in a more streamline manner than a manual assessment, especially upon re-assessment.* One option for achieving this is automating the assessment process where possible, which is the main contribution of LARA. Furthermore, when parts of the process are automated, it is important that the results maintain a high degree of accuracy compared to what the manual results would be when the

same input is provided. LARA should also be able to make use of already inputted information from an earlier risk assessment in order to reduce the amount of work required to set up the a re-assessment. In order to create incrementality, LARA should avoid direct "steps" in the implementation. Data should be able to be inputted into the system at different times during the risk assessment without causing issues, such as not being able to progress, or additional data causing the current assessment to become obsolete. LARA should also be user-friendly and easy to use. Not much extra time should be spent having to figure out how to use the tool if the user has a basic understanding of the methodology it is based on.

4.2 Specifying requirements

Upon having identified the needs of the intended target group, we must turn these needs into a set of concrete requirements for LARA. These requirements, which we call success criteria, are the minimum required features of the tool, as well as the criteria used to evaluate how well LARA can fulfill its intended purpose. Each criterion will be briefly described in a few sentences to capture the essence of the requirement while maintaining flexibility in the implementation.

Criterion 1

The tool must be automated. The tool should implement automatization in the parts of the methodology that demand manual search.

Criterion 2

The tool must be iterable. The tool should enable the input of new and editing of old data at any point in the assessment without causing conflicts in the results.

Criterion 3

The tool must be able to refine threats accurately based on the provided parameters. The tool should be able to produce the same results from the threat refinement as a manual refinement, provided that the same assumptions are applied to both refinements.

Criterion 4

The tool must be intuitive to use. The tool should be formed, both with regards to the layout and the implemented functions, in a manner which makes the functionality intuitive for the user. While it is assumed that a user understands the steps of the LINDDUN methodology, the user should be able to use the tool with minimal textual description on how to make use of each functionality.

Criterion 5

The tool must require minimal effort to set up, as well as a low barrier of entry. By making the tool easy to set up, it will help in lowering the barrier of entry, so the user can make use of the tool immediately. An easy to set up tool would be one with minimal system and program requirements, such as a browser-based tool which would require at most one new program.

4.3 Creating a prototype

When the success criteria have been formulated, we can proceed with the process of transforming the requirements into an operational tool. Given the time constraints of the software development process, in addition to any challenges met, it was deemed most appropriate to use an agile approach to the project. Based on the insights gained during the development, the tool was refined as necessary to improve performance and ensure iterability.

The detailed implementation of the tool is covered in chapter 5 and section 6.1.

4.4 Evaluation

The final step of the research method involves determining the evaluation approach for the solution. The evaluation is an integral part of the research method, as it provides an objective means to evaluate whether the individual components and the overall solution meet the established requirements. While usability testing [29] is the most appropriate method of evaluation throughout the process, performing a DPIA, partial or full, is an extensive and time consuming task. This is especially true when multiple individuals need to conduct the DPIA repeatedly. Consequently, we considered the use of use cases [30] to be sufficient for evaluation, as it would

still allow us to obtain qualitative data. Usability testing will instead be used to evaluate how intuitive LARA is to use.

The evaluation is covered in chapter 7.

Chapter 5

Tool Design

In this chapter, we will explore the implementation of the tool that serves as the basis for this master's thesis [12]. The discussion will encompass LARA's underlying architecture, its components, and the technical and functional choices made throughout the project. This includes the selection of functions, components, and external code libraries.

5.1 Architecture

In this section, we discuss how data is interacted with and managed in LARA.

LARA is divided into a frontend and a backend. The frontend is the user interface (UI) the user interacts with, while the backend is where the data is stored and all operations are performed.

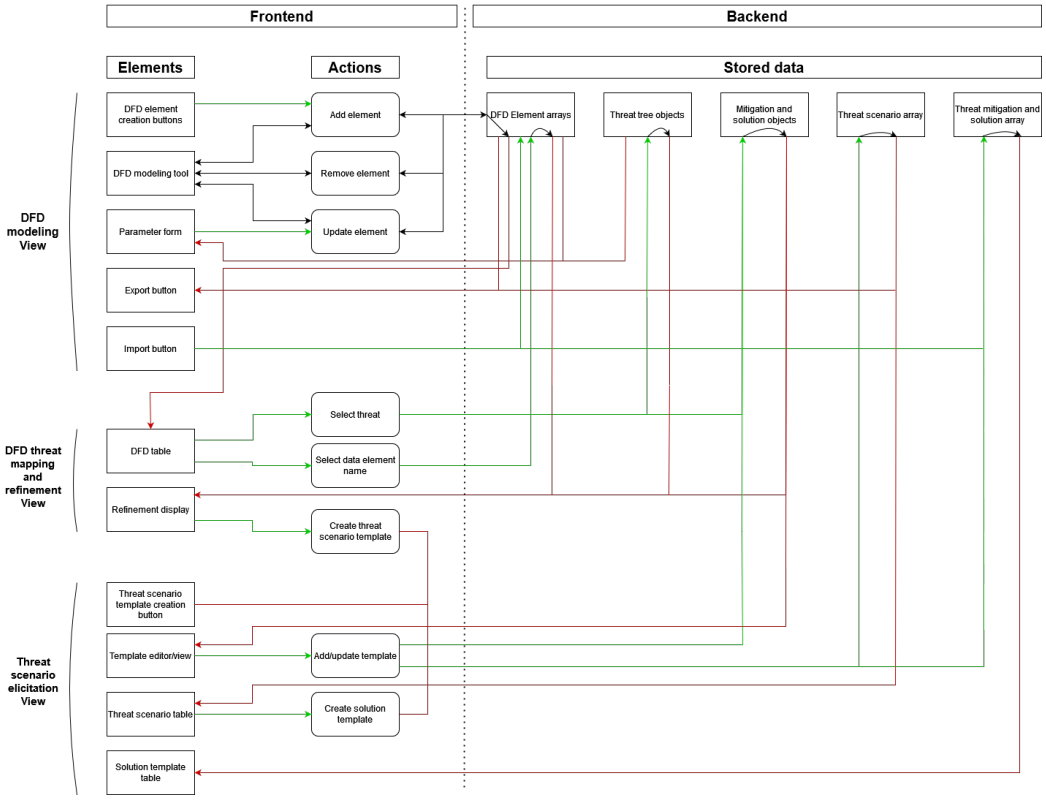

Figure 5.1: A diagram showing the main architecture of LARA. The components are split into the frontend and backend vertically, while the views and their functionalities are displayed horizontally. Green lines signify action, while red lines signify results. Black lines signify that the results of the action affect the element which made the action.

The frontend is divided into three views: the diagram view, the mapping and refinement view, and the template view. These views are separately displayed to the user through the use of tabs.

The backend consists of five data stores. The DFD element arrays, threat scenario and mitigation and solution arrays, the threat tree objects, and the mitigation and solution objects.

DFD modeling View

The DFD modeling View, where the user creates their system model, mainly interacts with the DFD element arrays. It is in the DFD view that DFD

elements are added, removed, and mainly has their data modified.

The DFD elements are created either through the use of buttons or the DFD modeling tool. As changes are made to the DFD element arrays, these are reflected in the DFD modeling tool, which is updated whenever a graphical change is made to the arrays. This can be actions such as the addition or removal of a DFD element, changing the coordinates of a DFD element, or making a change to a label or a color.

The parameter form receives data from a DFD element through the DFD modeling tool and displays it together with data it pulls from the threat tree objects. When edits have been made, it sends the updated data back to the DFD element arrays.

The export button pulls all data from the DFD element, threat scenario and threat mitigation and solution arrays, while the import button does the opposite, where it replaces the content of these arrays with the data that is to be imported.

DFD threat mapping and refinement View

The DFD threat mapping and refinement View only receive data which was added in the DFD modeling View.

The DFD threat mapping table is constantly updated based on changes made to the data of the DFD element arrays.

The refinement display receives data from the DFD table. Data from the DFD element arrays, the threat tree objects, and the mitigation and solution arrays are pulled and calculated before the results are displayed in the refinement display.

The DFD threat mapping and refinement View also contain a display for viewing and editing the parameter form of a DFD element. As it functions identically to the parameter form of The DFD modeling View, we have left it out of fig. 5.1.

Threat scenario elicitation View

The threat scenario elicitation View mainly interacts with the threat scenario array and the threat mitigation and solution array, in addition to pulling data from the mitigation and solution objects.

The only interaction between this and the other views occur in the template editor and view component, where it can receive data directly from the refinement display. The template editor and view can otherwise receive data from the threat scenario template creation button, as well as the threat scenario table. The template editor and view can add and edit the data of the threat scenario array, as well as add data to the threat mitigation and solution array.

The threat scenario table receives and displays data from the threat scenario array. The threat scenario table also sends data to the template editor and view component from threat scenario array or the mitigation and solution objects. It can also remove data from the threat scenario array.

The solution template table receives and displays the threat mitigation and solution array. It can also remove data from the array.

This architectural separation of the DFD modeling and refinement views and the threat scenario elicitation view is in part due to the latter being an addition to LARA, for the sake of rounding out the functionality to produce the same end results as the LINDDUN methodology. The separation of the functionalities still allows both parts to function as intended without causing undue interference with each other.

Component update differences

The core of LARA's functionality is the DFD element arrays. As can be seen in fig. 5.1, all components, with the exception of the threat scenario elicitation View components, interact with the DFD element arrays in some capacity.

The components differentiate between forms of updates. There are the components which are constantly updated whenever a change is made to the arrays, and the components which only update when specifically requested. The DFD modeling tool and the DFD mapping table makes use of the first type of update, as they are re-rendered, meaning that their graphics are updated, whenever a change in the data they pull from occurs. On the other hand, the parameter form and the refinement display does not change their content, even when changes are made. This is due to the data being fetched when the user selects on a component. As the data is only fetched once, rather than continuously, an update will only be reflected once the component is selected again and the data re-fetched.

As an example, we look at what occurs when a DFD element is added to the arrays.

1. The "Add entity" button is selected.
2. The new entity DFD element is added to the array.
3. Concurrently:
   (a) The diagram is updated with the new DFD element at the given position.
   (b) The DFD mapping table is updated with the new DFD element.

5.2 Components

LARA has four main functionalities:

- DFD modeling: The modeling of a DFD as a representative of the system to be analyzed.
- DFD threat mapping: The mapping of the DFD to a matrix table of the LINDDUN threats.
- Threat refinement: The automated refinement of identified threats, and finding threat mitigation strategies.
- Threat scenario elicitation: The creation of threat scenario templates using the refinement results, and defining solutions based on them.

Of the six LINDDUN steps, we have omitted the fourth step. As the prioritization of threat scenarios is a manual task, we decided against implementing it when other functions required prioritization.

As discussed in section 5.1, these functionalities are separated into three views:

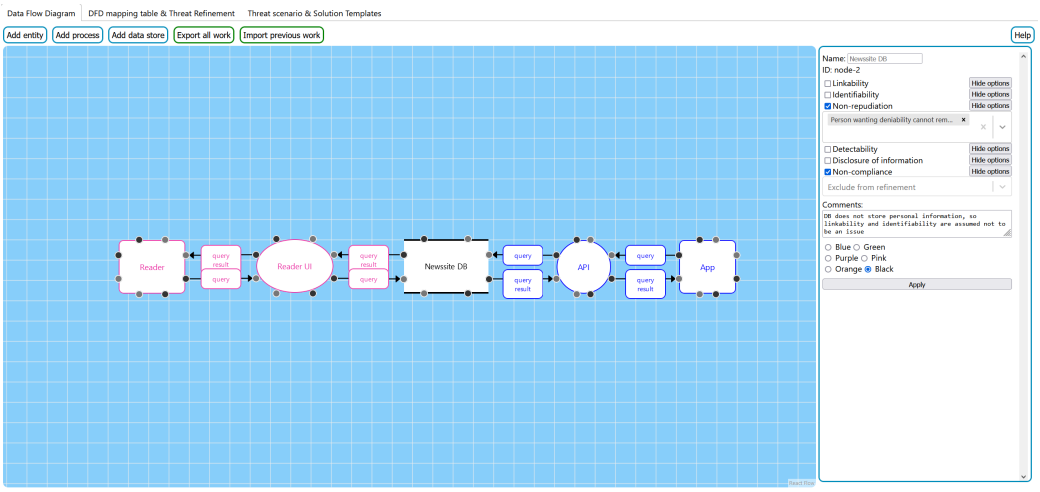

Figure 5.2: DFD modeling View.

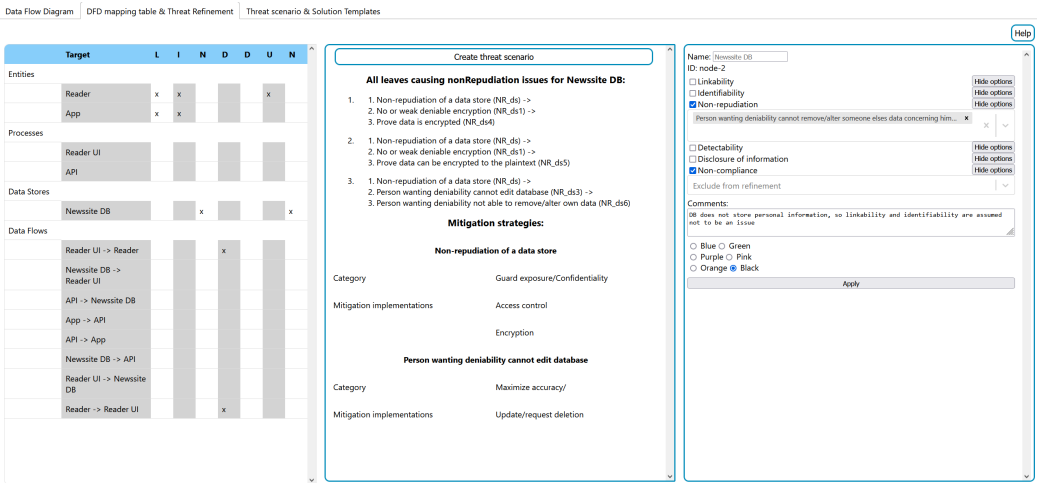

Figure 5.3: DFD threat mapping and threat refinement View.

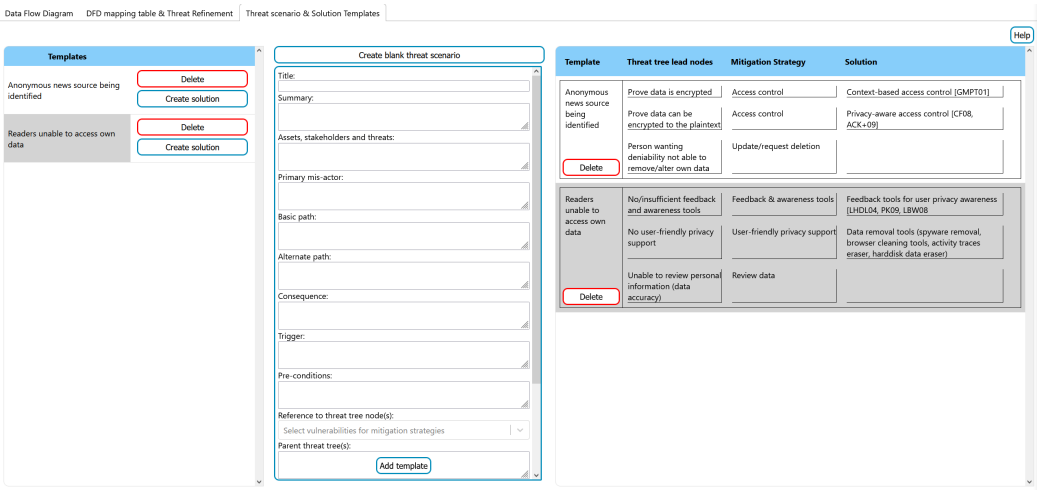

Figure 5.4: Threat scenario elicitation View.

Detailed descriptions of the components can be found in appendix A.

5.2.1 DFD modeling

The DFD modeling function has three components:

- DFD modeling tool: A interactive viewport where the DFD is created.
- DFD element, import and export bar: A bar of buttons for adding new DFD elements to the DFD, and for importing and exporting the diagram and related data.
- Parameter form: A display for viewing and changing the parameters of the DFD elements.

DFD modeling tool

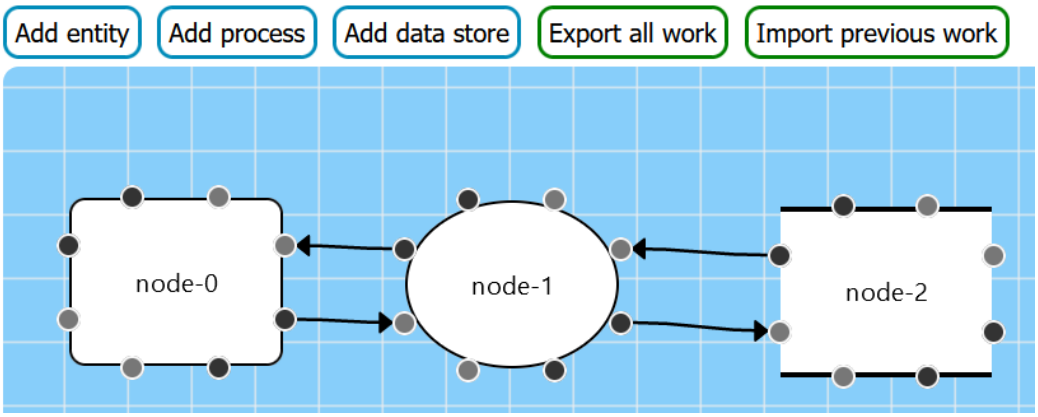

Figure 5.5: Diagram editor with interactive buttons.

The design of the DFD modeling tool comes from the React library used. We decided to give it a colored, patterned background in order to further mark the borders of the diagram editor. The decision of the placement of the buttons follows the common F shaped pattern [21], where a user's eye movement tend to start at the top of the page and read horizontally and downwards. By choosing this placement of the tabs and the buttons, we assumed that the user would become aware of their location on the page quickly.

A different color was chosen for the border of the import and the export buttons, as they serve a different purpose than the buttons for creating DFD elements. The buttons are placed there as the foremost data to export and import are the data entered into the DFD. The button labels are meant

to indicate that they export and import all data entered in LARA, rather than just the DFD. The parameter form display is set to the right of the DFD modeling tool as it is not the main focus of the functionality, and to prevent the need for scrolling on the page. Both the buttons and the DFD elements change color cursor shape when hovered over to indicate that they are interactable, while the drag points of each DFD element are differently colored to indicate which point is a source and target. The source is a darker gray, while the target is a lighter gray.

Parameter form

The parameter form is organized as a vertical list with the button to apply the parameters at the bottom, as it is what works best design-wise with the vertically aligned shape of the display. The threats, the comment field and the border coloring are grouped to create sections based on importance, with the threat checkboxes being sorted in the LINDDUN acronym. Each DFD element only displays the threats which are applicable to them, as defined by the LINDDUN tutorial table 1 [35].

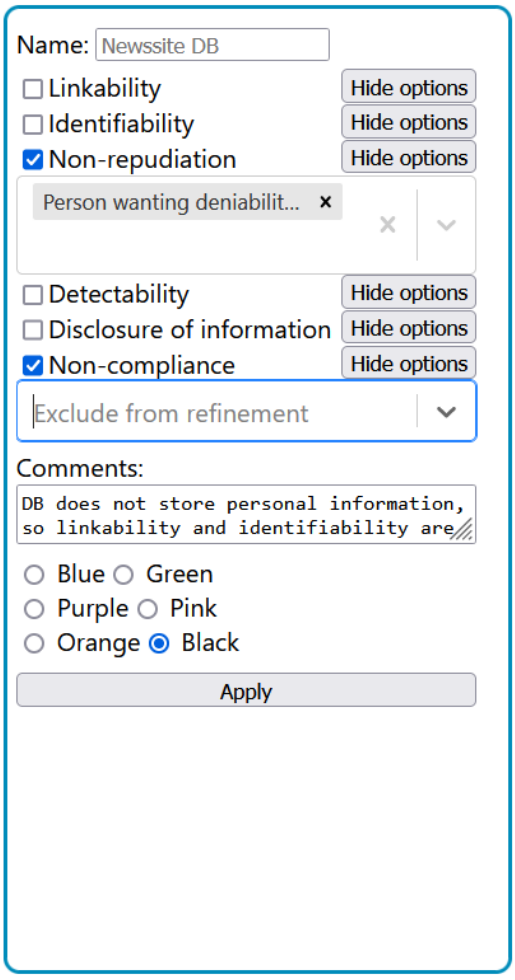

Figure 5.6: Editable DFD element parameter field.

Checkboxes are used to indicate if a threat has been applied, as they are commonly used and thus intuitive to understand. When a checkbox is checked, it causes a dropdown table to appear. These tables are initially hidden as they are irrelevant until the given threat is chosen, and only serves to clutter up the user interface. Likewise, the "hide options" button to the right of the threat re-

moves and returns the table to and from view. The more vulnerabilities the user chooses to exclude, the more surface the dropdown table will use, and so it provides the user with the option of removing them from view.

The options displayed for each threat are the base vulnerabilities of each threat tree. The only exception is if a branch vulnerability has a materialized threat from the same DFD element as its only cause. In that instance, the user is given the option of excluding the resulting vulnerability. This decision was made to account for these vulnerabilities being more general, in addition to them being the only cause for the vulnerability occurring.

The comment field, together with the checkboxes, corresponds to the third LINDDUN steps which documents assumptions [35]. The decision of providing a single comment field for each DFD element, rather than a singular list, was intended to make it easier to clarify what assumptions affect which element. This works in combination with applying the assumptions through dropdown tables. As the assumptions can be viewed from the threat map table as well, we believed this imitated a complete list, suiting the intended purpose.

The function of changing the color of the element border and text was inspired by an example analysis of a system provided by LINDDUN [34], where different parts of the DFD are colored. We chose to use radio buttons for this, with the default color set to the standard black.

5.2.2 DFD threat mapping and threat refinement

The DFD threat mapping function has three components:

- DFD threat mapping table: A matrix table of all DFD elements grouped by type.
- Parameter form display: An interface for viewing and editing the parameters chosen for a given DFD element.
- Threat refinement result display: An interface for examining the results of the threat refinement process and the identified mitigation strategies.

DFD threat mapping table

The design of the DFD mapping table is derived from figure 2 in the LINDDUN tutorial [35].

Devising an intuitive and efficient method to initiate the threat refinement process and present the results was one of the more significant challenges in this work. Originally, the idea was to have a "run" button beside the diagram editor. Once selected, it would refine all threats currently defined in the DFD. However, the issue which presented itself with this approach is twofold.

	Target	L	I	N	D	D	U	N
Entities								
	Reader	x	x				x	
	App	x	x					
Processes								
	Reader UI							
	API							
Data Stores								
	Newssite DB			x				x
Data Flows								
	Reader UI -> Reader				x			
	Newssite DB -> Reader UI							
	API -> Newssite DB							
	App -> API							
	API -> App							
	Newssite DB -> API							
	Reader UI -> Newssite DB							
	Reader -> Reader UI				x			

Figure 5.7: DFD threat mapping table.

The first was how we would present the generated data to the user. If all the results were to be presented at once, the user would need to sift through them to find the desired results. As the complexity of a DFD increases with the number of elements, so does the number of threats. Thus, this approach would become difficult to use. A more feasible option would be to present the vulnerabilities in a list per DFD element, where the user would select the desired DFD element and be shown all vulnerabilities sorted by the relevant threat. This approach leads to the second issue, which is efficiency.

When the data elements in the DFD are updated, the DFD mapping table is updated in tandem. When a change is introduced to the DFD, there is no guarantee that all the threat refinement results will remain the same. Thus, the whole DFD would have to be refined again, wasting computational resources.

As such, what we wanted to end up with was a manner to present the user with the option of refining only one threat at a time for a given DFD element.

The result was the idea of giving the DFD mapping table a second purpose, as the table already contained an up-to-date sorted list of DFD elements, separated into threat categories. By changing the color and cursor shape of the table elements when hovered over, we believe the table is presented as suitably intractable.

The option of refining the whole system at once could be useful to the analysis. However, it was not prioritized as the DFD threat mapping table is still capable of refining all threats.

Threat refinement result display

As LARA does not contain a function to display the threat trees, especially in a graphical manner, we decided to display the threat refinement results as lists containing the path from root to leaf vulnerability. By showing the entire path, the user would get an understanding of how the vulnerability causes the threat to materialize. The weakness of this solution is that it does not display when a vulnerability is composite of multiple vulnerabilities, thus requiring that all vulnerabilities are present. While the logic of threat refinement handles this, the user cannot view it in the results.

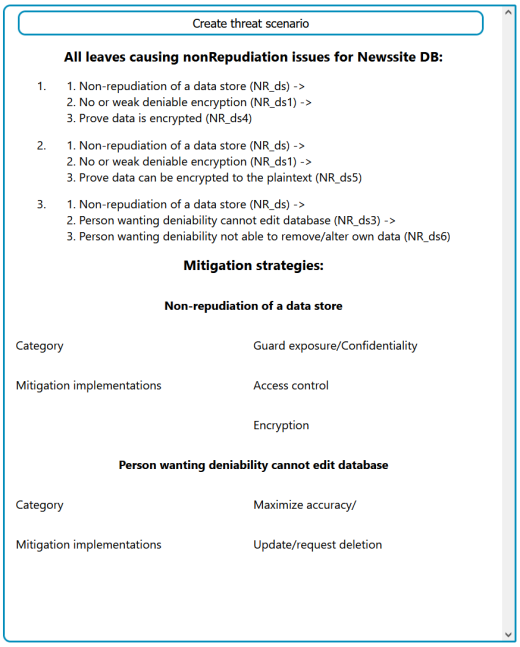

Figure 5.8: Example threat refinement results for non-repudiation in a data store.

The refinement process itself is described in-depth in section 6.1.

The calculation and display of the mitigation strategies at this step in the risk assessment is a result of the threat refinement being implemented before the decision to include the creation of threat scenarios and solutions was made. Due to this, we desired a way for the user to see the mitigation strategies with correspond to the vulnerabilities of the refinement.

The function was left as is, as it allows a user to preview the mitigation strategies available for the threat.

Parameter form display

The DFD threat mapping table has the additional functionality of displaying and editing attributes of a DFD element. Upon selecting the label of a DFD element in the table, the corresponding parameters will appear in the right-most display.

While originally intended as an extra functionality for simply viewing the inputted decisions made in the first view, we changed it to display the same form as the parameter form. This change was decided after performing a manual use case test. We learned that documenting general assumptions and applying the threats were easier from a sorted list. As such, a better solution was to have the form in both views.

5.2.3 Threat scenario elicitation

The threat scenario elicitation function has three components displayed.

- Template display: A display for creating, viewing and changing threat scenarios and create solutions.
- Threat scenario table: A table displaying all threat scenarios.
- Solution table: A table displaying all solutions.

Template displays

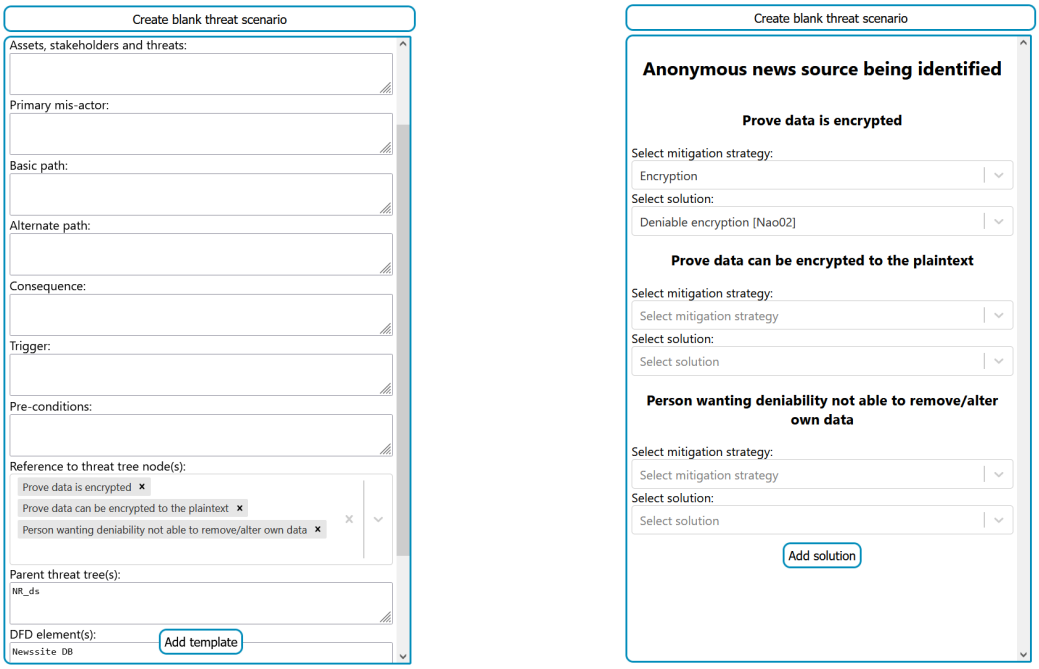

Figure 5.9: Threat and solution templates.

The template display can display two different templates: the *threat scenario template* and the *solution template*.

A threat scenario template can be created in two ways. The first is to create a threat scenario using the threat refinement results. This is done through using the button on top of the results. The view is then switched, and the threat scenario is filled in with the leaf vulnerabilities found, the threat tree, as well as the DFD element which was refined. The second is to use the button on top of the template display, which creates a blank template.

All fields except for the title are optional. The title is used as the visual identifier for all threat scenario entries, as well as the visual identifier of any solution table entry based on it. If this requirement is not met, the user will receive an alert which informs them of the requirement.

Although the original decision was to have the user fill in the leaf vulnerabilities manually using the node ID, we decided that using a dropdown table for selecting them was a better choice. By providing the user with a list, we avoid having the user accidentally writing something in the wrong format, in

addition to having to look up the node IDs externally. The dropdown table contains the leaf vulnerabilities of all LINDDUN threat trees.

The leaf vulnerabilities entered by using the refinement results are only the nodes which are part of the refined tree. As any external vulnerability is either the materialized threat itself or a branch vulnerability, finding all the leaf nodes would require more operations. As a user can easily look at the leaf vulnerabilities in the parameter form or refine the threat itself to find the leaf vulnerabilities, it was determined to be an adequate solution. The user can then add the leaf vulnerabilities to the threat scenario through the threat scenario template.

The threat scenario template is a one-to-one description of the template provided by the LINDDUN methodology [35]. The problem which occurs when attempting to automatize a methodology which relies on written text is that many of the functions does not allow themselves to be easily replaced. Thus, the decision to allow the user to edit the entered threat tree nodes or to create a blank template specifically is to handle the possibility of a composite threat scenario. This allows the user to edit the threat scenario as they please, as the threat scenarios would break if a sudden change were made to the underlying assumptions for the threat. It is instead better for the user to manually change the scenario to reflect the new situation or delete it themselves.

The design of the solution template is straight forward, as the options are limited. Like the threat mitigation displayed in the threat refinement results, the vulnerability is displayed on top, with the mitigation strategy and PET solution presented vertically to create a flow of direction.

Threat scenario and solution tables

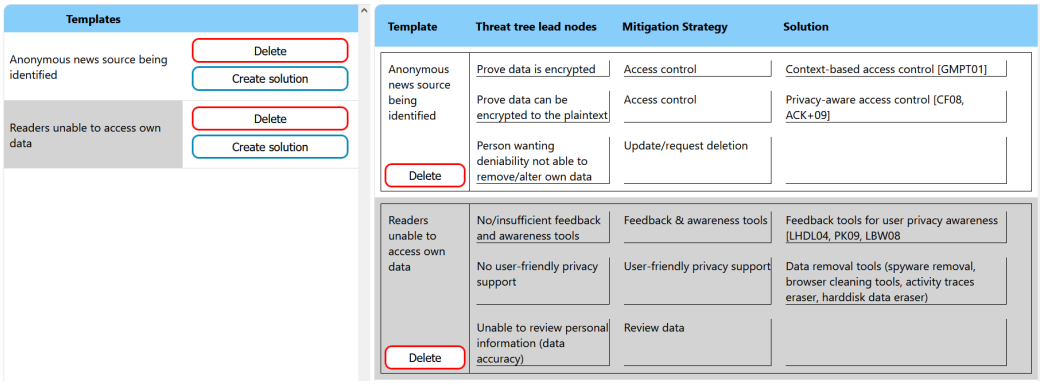

Figure 5.10: Threat scenario and solution tables.

The threat scenario and solution tables cover the fifth and sixth steps of the LINDDUN methodology.

The threat scenario table was added to create an easy way to display the threat scenarios without cluttering the UI. Selecting the title allows the user to view and edit the threat scenario. This was considered a good solution to prevent unnecessary use of space.

The solution table is entirely dependent on the vulnerabilities entered by the user in the threat scenario. The table format the finished solution is displayed on is taken from the LINDDUN tutorial [35].

By adding the templates in such a manner, LARA can still function as intended by the LINDDUN methodology, even if the results of the automatization fail the user's expectations.

5.3 Technical decisions

The technical decision section discusses the technical implementation which made use of external libraries.

5.3.1 React

LARA is a JavaScript-based program made using React [17], which is a JavaScript library for building UIs. React's greatest feature is it's focus on

making interactive user UIs easily. React was chosen as the framework for this work, as the work is based on interactivity with the user. Additionally, React provides a plethora of libraries made available by other users. There is always a need for assessing if what a library provides of readily available functionality outweighs the risk of future dependency issues, though this is not an issue for this particular work. This enabled us to focus solely on the development of the functionalities, rather than having to spend time developing our own components with both functionality and design. For this work, three libraries were used.

5.3.2 React Flow

React Flow [33] is a library for building interactive diagrams, featuring customizable node and edge types. In this work, React Flow is used to provide an easy way of creating an interactive diagram modeling tool.

The work particularly needed a diagram modeling tool which allowed us to change the appearance of the node elements to match the data elements of the DFD, as well as create edges between nodes which were stored as unique elements. React permits additional parameters to the components, which made it easy to add our own customized parameters for storing the selected LINDDUN threats and excluded vulnerabilities.

The most difficult requirement was to find a library which permitted the degree of modification of the data edges as we required, due to the data flow being its own data element in the DFD. Most libraries create an edge between nodes by dragging a line from one node to another, which does not pose a problem. The issue appears when the modification of the edge proves difficult, due to its parameters being set in the library module, rather than being made available to the user.

React Flow was the library we found which most closely matched the requirements for the diagram. React Flow allows the user to define both the component appearance and functionality, and the parameters of the elements.

5.3.3 React Tabs

As described earlier, the UI is divided into three views: DFD modeling view, DFD threat mapping and threat refinement view, and threat scenario elicitation view. The decision to split the UI into three sections was made due to the spatial requirements of the functionalities. A functionality of LARA is

for the user to model their system, which requires providing enough space for them to properly view the section of the diagram they are working on. Furthermore, the user can view the threats they have added to the data elements from the diagram, ensuring that no functionality is compromised by splitting the functionalities into views. The template functionality is separate from the functionalities, and thus separated into its own view as well.

React Tabs [28] was the React library chosen for creating the sections, as it only requires four components to create the simple interface required for the tabs.

5.3.4 React Select

Dropdown tables are used in multiple parts of LARA for displaying options. However, what they require of the dropdown table varies. For the data element parameter form, the dropdown tables requires that multiple options can be selected at the same time while only permitting the option to be selected once, as the user is selecting vulnerabilities to exclude. The solution template on the other hand requires that only one option may be selected, as the user only needs to set a single mitigation strategy and a single PET. Rather than creating a component that fulfills all these requirements, React Select [32] was chosen to fill in these needs.

Chapter 6

Core Functionality

In this chapter, we will explore the functionalities which forms the core of this thesis' contribution. This encompasses the threat refinement algorithm, as well as an examination of any additional features that were contemplated but ultimately not incorporated.

6.1 Threat Refinement

The LINDDUN tutorial describes the third step, eliciting privacy threats, as "considered the core execution step of the LINDDUN threat modeling methodology" [35]. Of the three parts this step consists of, LARA focuses on the threat refinement using the LINDDUN threat trees.

The goal of the threat refinement is to find out if the vulnerabilities applied to the DFD elements, those which were not excluded, result in materializing the threat. The resulting vulnerabilities and their paths are then presented to the user.

Each threat has a threat tree for a given DFD element. The exception is in the event where the threat is considered not to apply for the DFD element. One such threat is unawareness, being exclusively a threat for entities, thus not applying to data flows, data stores and processes.

As can be seen in fig. 6.1, a threat tree consists of a materialized threat at the root, with branches representing composite vulnerabilities, ending in leaf vulnerabilities. These vulnerabilities can, alone or in combination with other vulnerabilities, be the factors which cause a threat to materialize.

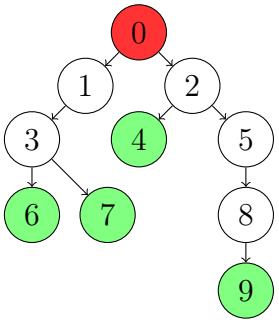

Figure 6.1: An example of the tree structure of the threat trees. The leaf nodes have been marked in green, while the root node is marked in red.

In LARA, we have implemented the LINDDUN threat trees described on the LINDDUN web-site, which includes STRIDE trees [7]. We did not include the trees which go beyond the DFD elements, instead opting to remove the nodes referring to them.

In the LINDDUN threat trees, the leaf nodes are the vulnerabilities most commonly exploited by attackers. We traverse the threat trees by starting at each leaf node which is not explicitly trusted by the user. Because we evaluate the tree upwards, only a simple recursive algorithm is needed.

6.1.1 Threat refinement steps

Step 1: Identifying the data flow of the DFD element

The first step of the threat refinement is to turn the DFD into a subset of DFD elements that might affect the refinement of the chosen DFD element. By turning the DFD into a graph on the form $G = (V, E)$, it becomes easier to traverse using the Depth-First Search (DFS) Algorithm [14] when finding data flows between elements.

The DFS algorithm is used to traverse a graph by choosing a starting point, a root, and traveling down one path at a time until it reaches the end before moving on to the next path. Using DFS with the DFD element to be refined set as the root node, we limit the number of DFD elements in the subset by using limiting parameters 6.1.4 to decide when the algorithm will stop the recursive calls. This is to prevent an unrelated DFD element from being part of the refinement. As an example, an entity should not be

Algorithm 1 createGraph

```
Input: V = DFD nodes, E = DFD edges
Output: G = (V, E)
graph := []
for all n ∈ V do
    edges := []
    for all e ∈ E do
        //If the edge is an edge of the node, add it to the list of edges for the node.
        if source(e) = id(n) or target(e) = id(n) then
            edges.insert(e)
        end if
        //If the edge is not in the list of DFD elements in the graph, add it.
        if e ∉ graph then
            graph.insert(e)
        end if
    end for
    graph.insert(n)
end for
return graph
```

Figure 6.2: Psuedo code for creating a graph from the DFD.

adversely affected by a data store it does not have a data flow with.

Step 2: Initiating recursion

The second step in the refinement is starting the recursive call of each leaf node in the threat tree. If the vulnerability lies in the DFD element itself, it is checked to see if it has been determined not to be a problem by the user, and thus excluded from refinement. If the vulnerability does not originate from the threat tree, and the resultant interal vulnerability has not been excluded, the threat refinement starts for the vulnerability.

What complicates the threat refinement are the vulnerabilities caused by other DFD elements, which can be handled differently from the refinement of the chosen DFD element. Unlike the chosen threat which we are refining, the refinement of all other threats need only be verified on whether the threat or branch vulnerability materialize. As discussed in section 5.2.2, we only use the results of the tree being refined, and thus we can make the refinement more efficient.

If a vulnerability has external causes, its threat category and DFD element type are found. The subset of DFD elements is then searched for matches where the correct DFD element type has the threat marked as potentially present. For each match found, it is checked to see if that combination of threat and specific DFD element has already been refined.

Each time a threat refinement starts, LARA creates and maintains a list of refinements performed. This is done in order to prevent potential re-refinements should more than one refinement reference the same DFD element and threat. This list maintains the level of refinement performed as well, as there are two forms of vulnerability refinement. The first is the full refinement of the threat three, while the other is the refinement of a subset of the threat tree. If the full threat tree has already been refined, a partial refinement is not needed. If only a partial refinement has been performed, a full refinement must be performed. If the refinement evaluates to be true, the threat refinement starts for the vulnerability. Tree traversal is further described in section 6.1.2.

Step 3: Threat refinement

The third step is the refinement. Once the leaf node vulnerabilities have been verified as present, each leaf node creates a list which it adds itself to, thus starting its path. It then recursively starts finding its parent node in

the tree. The parent node adds itself to the list, for then to add that list to its own list. For each path that visits a node, a new list is added to that node's list until it reaches the root.

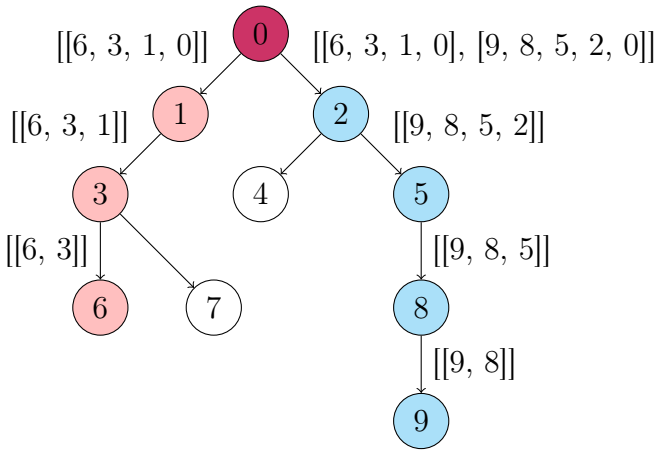

Figure 6.3: Example of a refined threat tree with its paths found from two leaf nodes. The left branch is refined before the right.

As shown in fig. 6.3's root node, node 0, when the left branch is refined, the root node contains the path $[6, 3, 1, 0]$. Once the right branch is refined, the root node then contains both path $[6, 3, 1, 0]$ and $[9, 8, 5, 2, 0]$. As such, once the refinement is completed, the root node contains a list of all paths which result in the materialization of the threat.

As mentioned in section 2.2.1, a vulnerability may require several pre-conditions to be fulfilled in order to materialize. In fig. 2.1, it can be seen that "Linkable login using untrusted communication" requires the pre-conditions "Linkable login" and "Untrusted communication" to evaluate as true before it can recursively move upwards on the tree again. When this occurs, the recursive calls that do not fulfill the conditions will add itself to the list, then terminate.

Should a recursive call which fulfills the condition reach the node, it will continue by using that list, thereby also continuing the other paths as a single recursive call. If the condition is never fulfilled, the recursive call ends there, which prevents the path from being registered as a path which materializes the threat.

The root node has its own check for whether the pre-conditions are fulfilled. As the root node is checked for paths which reached it, it is required

to manually check if there are conditions for them to materialize the threat.

Step 4: Mitigation strategy elicitation

The fourth step is finding the mitigation strategies which correspond to the subset of vulnerabilities. As mentioned in section 2.2.1, LINDDUN links each mitigation strategy to specific vulnerabilities. Thus, the paths found during the threat refinement are searched for vulnerabilities which have mitigation strategies provided. These are then added to a separate list together with the mitigation strategies and returned together with the printable paths to be displayed to the user.

6.1.2 Tree traversal solution

One of the main questions to solve was how to traverse the threat trees. As a tree structure where every node has a reference to its parent node and its child nodes, the initial thought was to traverse it from the root using DFS. However, it was determined that this would be difficult to implement, as the vulnerability evaluation moves upwards, and there would be no proper way to evaluate a path until the leaf node was reached. As such, it was determined that the most feasible way to traverse and evaluate a branch at the same time was to find all the leaf nodes and start the recursion there in a form of reverse DFS.

The refinement of a threat and the refinement of a vulnerability is performed differently. As the goal of the vulnerability refinement is to verify whether it materializes, the refinement ends when a vulnerability path has been verified to reach the root.

The greatest difference is that the vulnerability refinement can search for two different conditions. It can search for the root of the tree, or it can search for a specific pre-conditional vulnerability within the tree. This requires specific handling, as a threat tree may materialize if even one pre-condition is present, while a pre-condition may be present even if the threat tree in its entirety does not materialize. Additionally, because a threat tree may refer to other threat trees, a loop may occur in LARA, where two or more threat trees endlessly calls on each other if the target of the threat refinement is not specified.

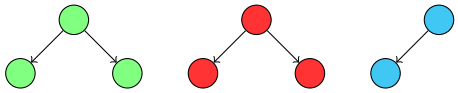

Figure 6.4: Example of three different threat trees.

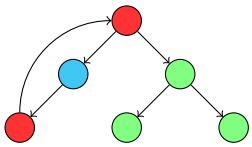

Figure 6.5: Example of a composite threat tree using fig. 6.4 where the red leaf nodes reference the root nodes of the green and blue threat trees, and the blue leaf node references back to the root node of the red threat tree.

This is solved with a two steps. The first is the aforementioned split of the vulnerability refinement into a full tree refinement and a pre-condition refinement. This is done by separating the pre-condition branch into its own, temporary threat tree and refining it in the same manner as the full threat tree. By doing so, LARA avoids attempting to call a loop. The second is that LARA has a check to avoid calling on the initial threat tree, unless the refinement is for the pre-condition branch. What both of these implementations prevent is a call for a threat tree root that is higher up on the composite tree structure, as the tree branches downwards with each call for another threat tree, and a call to a higher tree root would never be able to finish its calculation.

6.1.3 Threat refinement: psuedo code

This section contains the psuedo code for finding the vulnerabilities to be refined, as well as the psuedo code for traversing the threat trees.

Algorithm 2 refineThreatTree

Input: G = Tree graph,
X = Excluded nodes from G,
L = Diagram graph subset,
E = List of data elements,
D = Original data element,
T = Original threat,
M = List of refinements
Output: $visitedNodes(rootNode)$
//Find all nodes of G where the node has no child nodes.
$leaves := \forall u \in G$ if $leaves(u) := empty$
for all $u \in leaves$ **do**
  if $internal(u)$ and $u \notin X$ **then**
    $visited(u).insert(u)$
    $recursiveAscent(u, G)$
  end if
  if not $internal(u)$ and $u.root \notin X$ **then**
    //Find data elements of the same datatype as the leaf's threat.
    $elems := \forall n \in L$ if $datatype(n) = datatype(u)$
    for all $e \in elems$ **do**
      //If the threat is set as true for the data element.
      if $threat(u, threat(e)) = true$ **then**
        //If the refinement has not been done yet.
        if $partial\_refinement(u)$ and $M(u) = false$ **then**
          $M := refineSubTree(u)$
        else if $M(u) = false$ **then**
          $M := refineFullTree(u)$
        end if
        if $M(u) = true$ **then**
          $visited(u).insert(u)$
          $recursiveAscent(u, G)$
        end if
      end if
    end for
  end if
end for
if $opr(root)$ = 'AND' and $leaves(root) \notin visited(root)$ **then**
  return []
end if
return $visited(root)$

Algorithm 3 recursiveAscent

```
Input: u = Tree node,
       G = Tree graph
       R = Result
Output: recursiveAscent(u, G)
//Find root of u.
root := v ∈ G where v := root(u)
//If the refinement is not for a vulnerability
if ¬R then
  //Visited contains the path from the leaf nodes.
  //If the root has not been added to the path, do so, and add the paths to the root.
  for all e ∈ visited(u) do
    if root ∉ e then
      e.insert(root)
      visited(root).insert(e)
    end if
  end for
  if opr(root) = 'OR' or opr(root) = 'AND' and visited(root) ⊂ leaves(root) then
    //If the root is an element in the graph, continue recursion
    if root(root) ∈ G then
      recursiveAscent(root, G)
    end if
  end if
else
  if root ∉ e then
    root.insert(e)
  end if
  if opr(root) = 'OR' or opr(root) = 'AND' and visited(root) ⊂ leaves(root) then
    //If root is reached
    if root(root) ∉ G then
      if complete(R) then
        root(R) := true
      else
        subroot(R) := true
      end if
    else
      if externalVulnerability(root) then
        subroot(R) := true
      end if
      recursiveAscent(root, G, R)
    end if
  end if
end if
```

6.1.4 Data flow subset restrictions

An important part of the refinement process is creating a subset of DFD elements which may affect the threat refinement by causing vulnerabilities. This prevents redundant refinements, as well as lowering the accuracy of the refinement results, as the only candidates for refinement share a data flow with the DFD element to be refined.

The parameters set for direct connections are as follows:

- An entity is affected by all DFD elements in the data flow between it and all data stores it connects to.
- A data store is affected by all DFD elements in the data flow between it and all entities it connects to.
- A process is affected by all DFD elements in the data flow to the entity and data store it connects to.
- A data flow is affected by all DFD elements in the data flow to the entity and data store it connects to.
- The DFD element can only be affected by DFD elements of other types.

An entity being only connected up to the nearest data store and vice versa is intuitive, as it is contrary for a DFD element to be directly affected by a DFD element it does not share a data flow with.

The most difficult parameter to set was for the data flow. While a data flow element is one-directional, from one DFD element to another, it is part of a greater flow between elements. As the external vulnerabilities for a data flow element mostly comes from a data store element, it was decided that the most correct option would be that all data flow elements would have access to the data store they connect.

A weakness of this solution is that each data flow from entity to data store is required to be closed. In an example from the example analysis [34], each entity sends a token to the same "Authentication" process, which sends the validation back to the respective portal UI process. The issue this generates is that, while it is intuitive to a human reader of the diagram, LARA does not understand which process belongs to which data flow. When the "Patient" entity communicates with "Authentication", it does not comprehend that "Patient" should not have access to either the "Researcher Portal" or "Nurse Portal" processes, nor the "Researcher" and "Nurse" entities. As

such, LARA requires the user to create a separate "Authentication" process for each data flow, even though a user logically understands that each process is a single component.

6.2 Discarded functionality

This section discusses the functionality which was intended to be implemented into LARA, but had to be cut due to time constraints and complexity.

Threat grouping

As mentioned in section 2.2.1, the third LINDDUN step has an activity which regards documenting assumptions about whether or not to trust the components of the system to behave as expected, as well as general assumptions in regards to their behaviour [35]. A function we intended to implement was one that allowed the user to group elements together under a fellow assumption. This would allow the user to create an assumption for a threat category and a DFD element type with exclusions, then add DFD elements along with a text explaining the assumption made. The assumptions would then be applied to the DFD elements.

An issue which appears is the overwriting of already set parameters. If a user were to manually overwrite the parameters for a DFD element which is part of an assumption, it would create an inconsistency in the assumption. The issue would then become how to prioritize the applied threats and vulnerabilities.

While we believe that this idea is feasible, the complexity involved led us to prioritize more important functionality, resulting in this particular feature being postponed and ultimately omitted.

Branch exclusion

Unlike our solution, the LINDDUN methodology allows the user to exclude entire branches of the threat tree. While the choice of excluding multiple vulnerabilities at once is more practical, we opted against this approach. Per the requirements of the data library used for dropdowns, the selected options are stored in a list. For the user to exclude entire vulnerability branches, the entire tree except for the root would have to be options. The problem occurs when a branch vulnerability is selected. All options in the given

branch should then be removed as well, and be returned upon deselecting the branch. While this is doable, the threat refinement which uses them starts the calculation from the leaf vulnerabilities. As such, the refinement would also have to handle starting the refinement from anywhere on the tree, which quickly became complicated to achieve.

Due to time constraints, it was omitted.

Predefine DFD elements

As explained in section 6.1.4, one of the limitations of the threat refinement is the requirement of single-use processes. An idea to mitigate the issue of repetitively having to recreate the same DFD element was the option of predefining a data element.

In a similar manner to how a blank DFD element is created when a button is clicked, the user would be given the option of saving a DFD element to a dropdown table, where selecting the element would add this to the DFD editor.

The issue which complicates this relatively simple solution is that not all data elements allow themselves to be created in that manner. Data flow elements must be created through either dragging from a source node to a target node or be presented with a source and a target beforehand. Likewise, as all DFD elements are connected though data flows, a predefined DFD element would require the predefinition of its connecting data flows to facilitate the process.

Therefore, since this functionality is practical but not essential, it was omitted.

Chapter 7

Evaluation

Once the prototype of the tool is completed, the next step is to evaluate it. In this chapter, we will evaluate LARA in two ways.

The functionality of LARA will be tested through the use of a case study. The case itself will based on TelluCare, which is a commercial system developed by Tellu AS. It is in use by, amongst others, home care services in Bergen, Indre Østfold and Grimstad. Creative liberties are taken regarding the details and complexity of the system, for the purpose of better fitting with the use case.

The case study will consist of the following steps:

1. Use case testing, where the results of LARA are evaluated based on the expected results.
    - As the main contribution of LARA lies in the problem space, the testing will be performed up to and including the threat refinement.
2. A comparison between the functionalities of LARA and the LINDDUN implementation of the OWASP Threat Dragon tool.

The usability of LARA will be tested through user testing with an imaginary scenario.

7.1 Case study

7.1.1 Use case example system - Tellu

The system the use cases will be based on is one presented by Tellu AS [25], which regards remote patient monitoring.

The use cases centers around the Tellu platform, which is a part of the TelluCare sandbox [26]. As described by Tellu themselves [24], the Tellu platform "can be integrated with services, medical equipment, sensors, applications and digital solutions from Tellu or third parties" as well as having the functionality for "single sign-on through Azure AD, Health ID or ID Porten".

The remote patient monitoring is implemented through the use of applications. The patients report their own health using the patient app though a mobile unit. The patient answers health questionnaires and takes relevant measures using sensors connected to a medical gateway, which encrypt and transfer the data to the Tellu platform, as well as communicating with health personnel. Health personnel use Tellu's web app to obtain access to the follow-up centre containing all the reported data, from which they monitor and follow patients up. Data may also be securely shared with other practitioners.

Because we only have a textual description of the Tellu system, we do not know the architecture in-depth, and will instead make use of our own assumptions to build an understanding of the system which suit our needs for testing LARA.

7.1.2 Use case testing

Based on the description of the Tellu system, we make some additional assumptions for the DFD:

1. The sign-in methods have been generalized into a single authentication process, with the assumption that the methods have been implemented in the same manner.

2. The measurements provided by the user to the sensors are one directional. In the scope of the system, any manual feedback such as the display of a bathroom scale are uninteresting.

3. The medical sensor only send data to the medical gateway. We assume that, as third party tools, any updates goes through those third parties.
4. Though Tellu doesn't describe it, we assume the gateway receives hardware updates from the platform.
5. We assume that the Tellu Platform contains the entirety of the TelluCare architecture, but we only focus on the parts relevant to the case, which is the data provided by the users.
6. We assume that patients and health personnel communicate through the Tellu Platform.
7. We assume data is provided from the Tellu backend by using the user's credentials, such as a unique ID for the specific API.

Use case 1 - Perform a partial DPIA

Use case	1
Scenario	Performing a threat refinement of the full system
Use case steps	1. The user creates a DFD based on the system description. 2. The user maps each DFD element to a table, then fills in each threat the DFD elements are susceptible to based on type. 3. The user makes a list of assumptions about what parts of the system to trust. 4. The user refines the threats based on these assumptions.
Prerequisites	Knowledge of how to use the LINDDUN methodology.
Expected results	The threat refinement results are consistent with the results of a manual refinement.

1. We create the DFD in LARA:

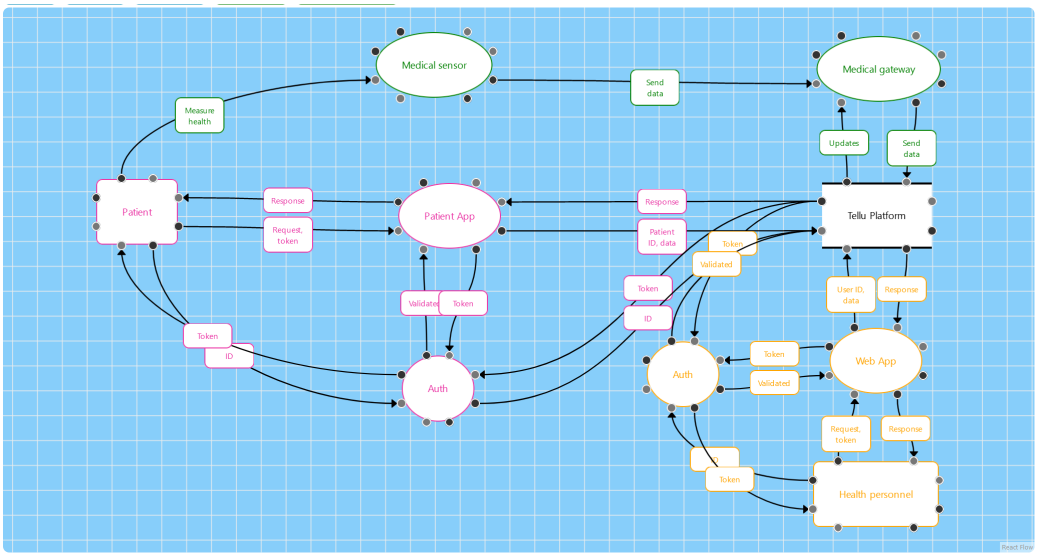

Figure 7.1: DFD of the Tellu system.

2. Due to LARA only allowing the user to select the threats which each DFD element is susceptible to, based on table 1 in [35], we decide that we don't need to fill in the threats yet.

3. We record the assumptions for each DFD element:

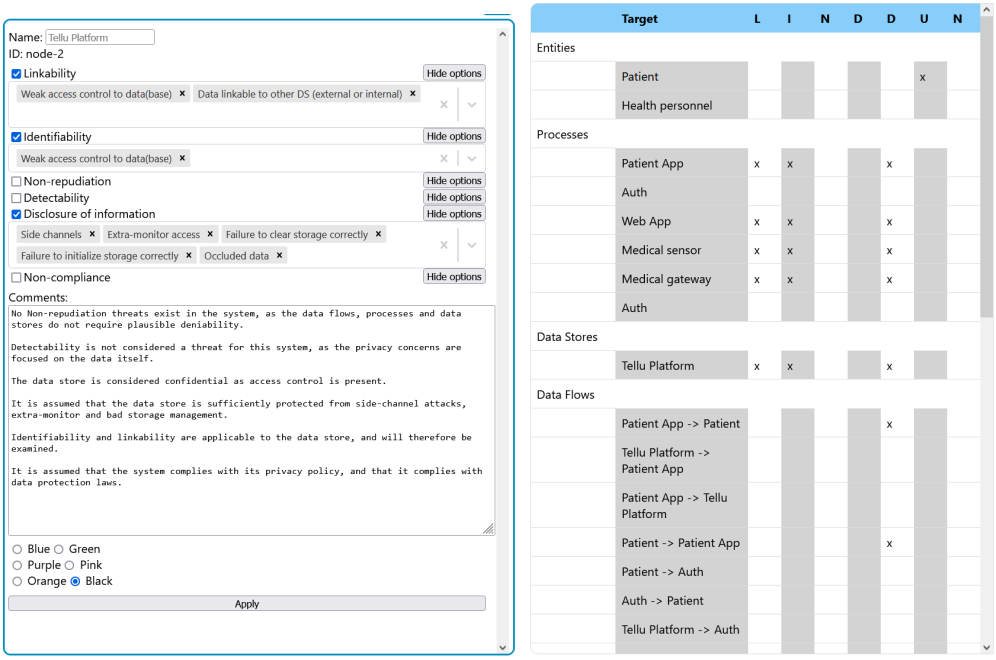

Figure 7.2: Example of the assumption documentation and parameter setting, as well as the DFD threat mapping of the Tellu system. The full mapping can be found in appendix B.1 and appendix B.1.

4. We then refine each DFD element:

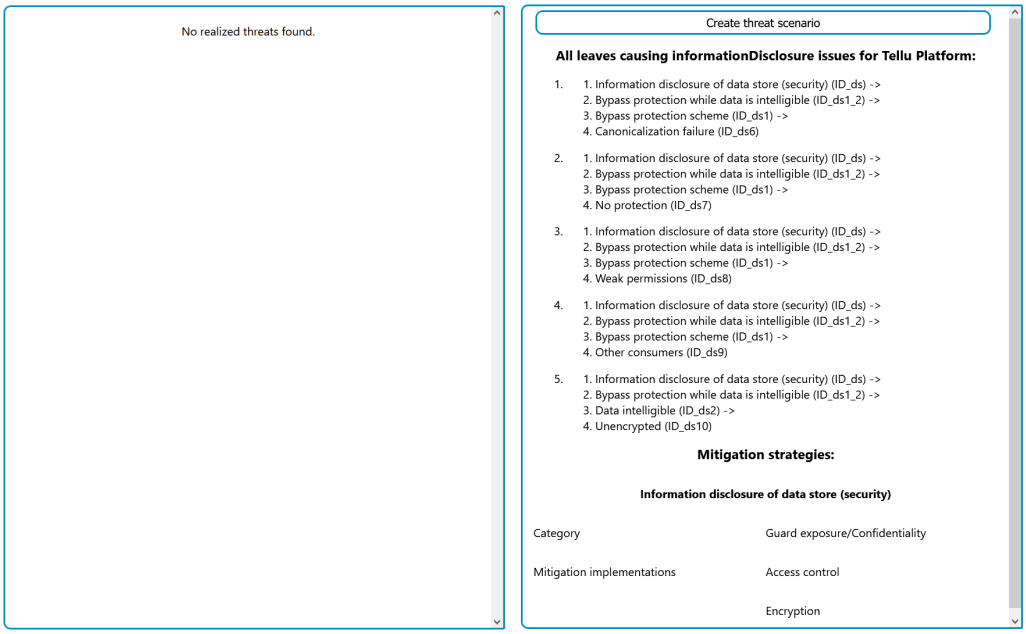

Figure 7.3: The threat refinement results for linkability and information disclosure of the Tellu Platform data store as shown in LARA.

fig. 7.3 displays the results of the linkability and information disclosure threat refinements. The results of the linkability threat refinement show that the linkability threat does not materialize. The results of the information disclosure threat refinement show that the threat materializes. This is shown to be due to the bypassing of the protection scheme, which is caused by both canonicalization failure, no protection, weak permissions, and other consumers. Additionally, the data is intelligible due to being unencrypted.

The results of the threat refinement for use case 1 matches with the manual refinement using the same specifications. The full results can be found in appendix B.1 and appendix B.2.

Use case 2 - Re-assessing use case 1 with changed requirements

In the second use case, we assume that changes have been made to the system:

1. Tellu noted the information disclosure threat to the data store, and implemented encryption per the recommended mitigation strategy.
2. Tellu has implemented an interface for 3rd party services to connect to the platform.
3. Due to the interface being in the early stages of development, the communication between the interface and service is untrusted.

Use case	2
Scenario	Re-assessing use case 1 with changed requirements
Use case steps	1. The user updates the DFD to incorporate the changes. 2. The user maps each DFD element to a table, then fills in each threat the DFD elements are susceptible to based on type. 3. The user updates the list of assumptions about what parts of the system to trust. 4. The user refines the threats based on these assumptions.
Prerequisites	Knowledge of how to use the LINDDUN methodology. The results of use case 1.
Expected results	The threat refinement results should match the results of a manual refinement. The re-assessment should not have required the recreation of any present DFD elements.

1. We update the DFD with the new DFD elements:

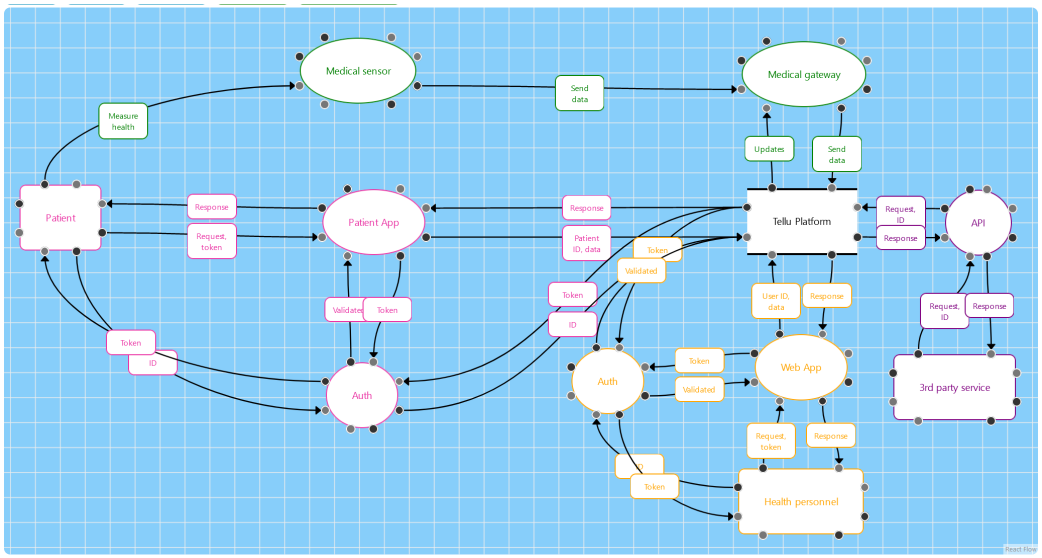

Figure 7.4: Updated DFD of the Tellu system.

2. Not necessary to perform for the same reason as in use case 1.
3. Altered and added assumptions:
   (a) Tellu Platform
      i. The Tellu Platform implements encryption as mitigation.
   (b) 3rd party service
      i. No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.
      ii. Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.
      iii. Identifiability of services is not considered a threat, as they do not possess any personal data in the system.
      iv. Linkability of services is not considered a threat, as they do not possess any personal data in the system.

v. Unawareness of services is not a threat, as it is not a natural person.

(c) Tellu Platform ↔ API

i. No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

ii. Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

iii. Data flows between the data store and processes are considered trusted, as it is assumed the communication occurs over a secure communication line.

iv. It is assumed that the system complies with its privacy policy, and that it complies with data protection laws.

(d) API ↔ 3rd party service

i. No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

ii. Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

iii. Data flows between an entity and an app process is not trusted, as it involves communication over an insecure communication line.

iv. Identifiability and linkability are applicable to the data flow, and will therefore be examined.

v. As side-channel attacks require a lot of analysis, it is assumed to be unlikely for them to occur at data flows.

vi. It is assumed that the system complies with its privacy policy, and that it complies with data protection laws.

4. Changed threat results:

(a) 3rd party service have no materialized threats.

(b) The disclosure of information threat for the Tellu Platform no longer materializes.

(c) The data flows between the Tellu Platform and the API have no materialized threats.

(d) The data flows between the 3rd party service and the API materializes threats for:

i. Linkability: L_df8, L_df9, L_df10, L_df11, L_df12, L_df13, L_df14, L_ds2 and ID_df.

ii. Identifiability: I_df8, I_df9, I_df10, I_df11, I_df12, I_df13, I_df14, I_ds2 and ID_df.

iii. Information disclosure: ID_df4, ID_df5, ID_df6 and ID_df7.

The results of the threat refinement for use case 2 matches with the manual refinement using the same specifications. The full results of the manual refinement can be found in appendix B.2.

7.1.3 Threat Dragon comparison

The OWASP [20] Threat Dragon [19] is a threat modeling tool which supports LINDDUN.

The Threat Dragon tool implements a modeling tool with the four DFD elements. Additionally, it contains components for displaying a trust boundary.

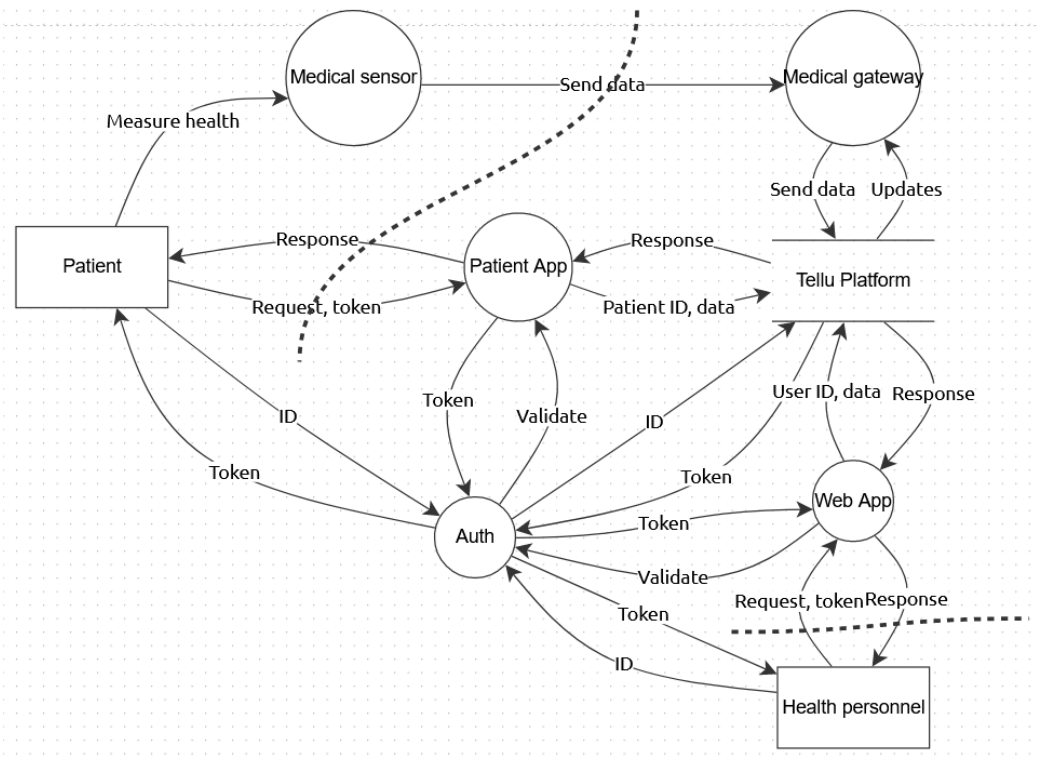

Figure 7.5: Example DFD of the Tellu use case using the OWASP Threat Dragon modeling tool. The stippled lines are the trust boundary

The Threat Dragon tool's most prominent feature is the addition of threats to a DFD element. Just as in LARA, the threat categories applicable to a DFD element are limited to those designated by LINDDUN. The user chooses the threat category, a score for the threat, a priority, as well as providing a description of the threat and how to prevent it. Finally, the user can set the status of whether the threat has been mitigated or not. Like how LARA combines several LINDDUN steps into a single functionality, this feature combines the threat scenario, priority ranking and mitigation.

The greatest difference between the Threat Dragon tool and LARA is that LARA actively makes use of the LINDDUN knowledge base, placing focus on automating the processes where the knowledge base is applied. This includes both the problem space evaluated in the use case tests, as well as the solution space of LARA.

If we compare the two tools using a similar scenario, the greatest difference to the threat modeling process would be the degree of which the user

leans on the LINDDUN methodology to perform the threat modeling and risk analysis. While the Threat Dragon tool offers the user the ability to assign multiple threats and mitigations to only one DFD element each, LARA offers the combined use of both the threat trees and vulnerability refinement to create threat scenarios resulting from multiple threats. Additionally, LARA offers the user mitigation strategies and PETs in the tool. A user with extensive knowledge of various threat scenarios and mitigation strategies could be seen as not requiring this functionality from LARA, however other users would in some form be required to look these up in some way. As such, LARA offers an in-depth analysis of the system, rather than merely a tool for documenting them.

7.2 Usability testing

The participants of the usability testing were five people employed in the Sustainable Communication Technologies department at Sintef. None of the participants had prior experience with LARA or the LINDDUN methodology.

The test questions can be found at appendix C.

Procedure

The tasks the participants were to perform were formulated for the testing to last for around 10 minutes each. As the purpose was to test the intuitiveness of LARA, the participants were given no further instuctions other than the tasks. During the testing, the participants were allowed to ask questions if they were unable to progress or had difficulty in understanding the tasks. Instances where the users experienced difficulties and were assisted in some form were recorded in accordance with the task they were performing.

Results

Usability testing results											
Task	S1		S2		S3		S4		S5		
	time	i*	time	i*	time	i*	time	i*	time	i*	Avg time
T1	02:30	Apply button	02:11		02:06	DFD Elements	02:38		01:54		02:15
T2	01:17		01:01		01:11		01:30		02:42	Arrow direction	01:32
T3	00:29		01:56	Place to mark	01:07		00:55		01:05		01:06
T4	01:42	Click flow	01:58	"Nothing to do". Click flow	01:21	Link (read the help himself)	01:21		03:11	Click flow	01:55
T5	03:28	"L" location	01:58	"L" location	00:18		00:04		00:32		01:16
T6	00:30		01:05		00:39		00:44		00:42		00:44
T7	00:20		00:58	"Table"	00:42		00:39		00:49		00:42
T8	00:20		00:52		00:34		00:33		00:26		00:33
Total time	09:51		11:50		08:01		09:02		11:35		10:04

Table 7.1: Table displaying the results. i* is short for interaction, SX is the subject, and TX is the task performed.

The times of T1 vary by approximately 45 seconds. This use of time can be attributed to the participants familiarizing themselves with the layout of the tool. An issue which occurred at T1 was one participant not noticing the "apply" button for the parameter form. Another was a participant becoming confused by the which DFD element was of a certain type.

T2 was straight-forward compared to the T1, which can be seen by the time results. The result which stands out at 2:42 was caused by the participant confusing which way the arrows went, which made the participant want to remove it, which proved difficult. A common occurrence with the task was that most participants expected the topmost arrow to point right,

which caused confusion regarding whether they had made a mistake.

T3 was performed quickly as well. This is assumed to be due to the participants already having looked over the parameter form during T1. The only result which differs is caused by issues with finding out which checkbox was to be marked, due to not having read the task properly. The participants had little issue with excluding vulnerabilities from the threat refinement as well.

T4, which was similar to T3 as well, was more challenging for the participants. The first issue which occurred was being able to interact with the data flow elements, as they were of a smaller size than the other DFD elements. The second was that none of the participants understood how to apply "trusted with no threats applicable". Although the correct answer was that no threats needed to be marked, as it was written in the task, many mistook it as something which required action.

The results of T5 vary greatly. The participants which spent the longest time on this task struggled with understanding which column in the threat mapping table corresponded to "linkability". Beyond that, the task was easily achieved by all participants.

T6 was performed by all participants with no issues occurring.

T7 went smoothly for most participants as well. Some misunderstood that "clicking on the title in the template table" meant the title of the threat scenario, and instead tried to click the titles of the threat scenario table and the threat solution table.

The T8 was achieved with great ease as well.

On average, the participants spent 10 minutes as estimated.

Chapter 8

Discussion

In this chapter, we will discuss the results of the evaluation, the validity and reliability of the results, as well as to what degree we have achieved the thesis aims.

8.1 Evaluation results

8.1.1 Case study results

Case study evaluation

When we compare the execution of the use cases in section 7.1.2 and section 7.1.2, to the manual LINDDUN methodology, the process has been smoothed out overall.

The implementation of the automated threat refinement simplifies the process of performing the partial DPIA. By directly applying the decision of trusting parts of the process onto the given DFD element next to the documentation, it clarifies exactly which vulnerabilities the assumptions correspond to. The automated threat refinement proves the most helpful when the cause of the threat occurs outside the threat tree, as well as when multiple re-refinements are necessary. An example of an external threat occurs in the data flows between the 3rd party service and the API. While both the linkability, identifiability and information disclosure threats do not materialize for the Tellu Platform, due to the threat itself being possible, we are made

aware that the inference vulnerabiliy is still present and propagates into a vulnerability elsewhere. As the Tellu Platform has already been refined at that point, it can be easy to miss this vulnerability in the myriad of refinements. This, in addition to preventing the need for tracing the threat trees multiple times when re-assess the DFD elements when checking for changes improves the iterability of the partial DPIA.

As such, we conclude that the process has achieved an increased streamlining of the assessment process.

Points of improvement

The biggest point of improvement for the tool is the lack of additional restrictions. As can be seen in fig. 7.1 and fig. 7.4, in order to prevent a connection between Patient and Web App through the Auth process, the authentication has to be represented through two processes in the diagram. This could be avoided if the user was able to define which DFD elements are part of a given flow, instead of relying on the internal logic described in section 6.1.4. The data flow between the medical sensor and the medical gateway too, pick up on the inference vulnerability, even though it is not affected by it. As such, this is a point for the tool to improve on.

Another issue which can be improved is the lack of generalization with regards to the DFD elements. An example from the use cases is that all data flows from the data store to the processes use the same technology and thus have the same assumptions applied to them. In the current implementation of the tool, a user is required to apply the same assumptions and documentation to all data flows. A possible solution to this issue was discussed in section 6.2.

The final noticeable point of improvement is being able to view the entire threat tree. Due to the current implementation only displaying the leaf vulnerabilities, it can be difficult to evaluate which vulnerabilities to remove from the system as their larger context is unavailable for review. While each path from leaf to root can be seen when refining a threat, it is a heavy handed approach which could be refined further.

Validity and reliability

Due to the difficulty of privacy threat modeling, the limited expertise of the tester, as well as the limits of the use cases, the reliability of the test results relating to increased incrementality are questionable. While we attempted to be as objective as possible in our evaluation, the time constraints and lack

of objective testers made this form of testing the best we could do with what we were given. Additionally, considering the extensiveness of a full DPIA, and the variations in experience of the analyst [37], we cannot guarantee the reliability of the results, though they would provide more accuracy from a realistic performance.

The evaluation of the accuracy of the threat refinement can be concluded to have a high degree of reliability. As the tests of the automated threat refinement (appendix B.1, 4) all returned the expected results based on the manual refinement (appendix B.2), we conclude that there are few errors in the threat refinement algorithm.

8.1.2 Usability testing results

Usability testing evaluation

When we review the results of the usability testing 7.2, we conclude that LARA fulfills the requirements to be called intuitive.

Some of the issues the participants encountered, such as confusing DFD element types and not understanding that the "L" of the threat mapping table refers to the linkability threat, can be tied to their lack of knowledge of the LINDDUN methodology. As we have set a basic understanding of the methodology as a requirement to use LARA, we choose to disregard those results. As these shapes are taken directly form the LINDDUN methodology, we assume that someone with knowledge of it would not have run into these issues. This rings especially true as most of the participants did not encounter the problem.

Many of the other issues were either the result of small cosmetic mistakes, such as the threat scenario and solution tables just being named "Templates".

The greatest issue the participants encountered was the difficulty of interacting with the data flow elements. As this is an issue of the modeling tool, and not of the participants not understanding how to use it, we choose to set this problem aside for this evaluation.

The participants also proved that they easily learned how to make use of the functionality once they grained some experience with the functions and their locations.

As such, we conclude that LARA is intuitive to use.

Points of improvement

Some of the issues encountered by the participants directly relates to the React library used for creating diagrams 5.3.2. Many participants found it difficult to interact with the data flow elements due to the small size. This issue has a work-around due the parameter form being present in the second view, though it is not intuitive unless one is aware it is there.

Some participants also struggled with the issue of the diagram not reflecting changes immediately. The modeling tool requires some form of interaction before it can display changes made through the parameter form. This caused some of the users to become confused about whether the changes they had made were registered in LARA. While we are unable to change the library used, applying feedback by making the form disappear upon applying the modifications could mitigate the issue.

Another issue which we were made aware of during the testing was that the data flows which had not been given a label were difficult to separate when using the parameter form. A possible solution for this could be to display the labels of the DFD elements being linked, rather than merely the ID. Another would be to change the color of the line when a data flow element is focused on.

Validity and reliability

The reliability of the usability testing can vary depending on how one chooses to view it.

The reliability can be considered to be high when considering the relative ease a test group consisting of participants with no prior knowledge performed the tasks. On the other hand, a test group with knowledge of the methodology might have found other issues which have gone undiscovered.

The reliability of the results can also be questioned based on the age of the participants. The fastest test results with little required assistance, which speaks in LARA's favour, were produced by the youngest participants, while the older ones struggled more. This could be an issue of how used the participants are in using digital tools in general, which affects the test results.

Finally, the formulation of the test questions could have caused issues which lowered the results. An example is the marking of a data flow element with no applicable threats in task four. This is a task requiring knowledge of the LINDDUN methodology, and the participants stumbled on.

As such, while there are many variations which can produce flaws in the reliability of the results, we feel confident that the results produced are usable for objectively evaluating the intuitiveness of LARA.

8.2 Success criteria

In this section, we will present our results in comparison with the success criteria we defined in section 4.2. We will then evaluate to which degree we believe we have been able to achieve the overall goals of this thesis.

Criterion 1 - The tool must be automated

We consider this criterion 4.2 to be partially achieved. The most easily automatable parts of the LINDDUN methodology are those which made use of the knowledge base. This was achieved through implementing the threat refinement algorithm, the DFD threat mapping table, and the parameters. It falls short due to the lack of automatization of the threat scenarios, which is caused by a lack of knowledge base for common threat scenarios resulting from a combination of vulnerabilities. As we were able to automatize parts of the process, we consider this criterion partially achieved.

Criterion 2 - The tool must be iterable

We consider this criterion 4.2 partially achieved as well. The evaluation results of the second use case showed that there were no issues with adding new DFD elements to the DFD, nor with updating the assumptions and refining them. However, we were unable to implement support for iterability for the threat scenario functionality, as a result of the lack of automatization. This led to the sectioning of functionalities, rather than the desired connectivity between all functions. As such, this requirement, while achieved in some parts still has potential for further work.

Criterion 3 - The tool must be able to refine threats accurately based on the provided parameters

We consider this criterion 4.2 to be achieved. When we performed a manual refinement of the system described in the first use case, the results were accurate based on the conditions entered into the system. The automated refinement even discovered a few errors made in the manual refinement. As such, this requirement is considered fulfilled.

Criterion 4 - The tool must be intuitive to use

We consider the criterion 4.2 to be achieved. While the reliability of the testing 8.1.2 can be uncertain, we feel comfortable that the positive results hold. As such, this requirement is considered fulfilled.

Criterion 5 - The tool must require minimal effort to set up, as well as a low barrier of entry

We consider this criterion 4.2 to be achieved. By making the tool web-based, the tool does not depend on any singular system, as most web-browsers are cross-platform. We tested the tool in the web-browser Firefox on both Windows 10 and 11 without issues. As such, we consider the tool to be easily accessed and thus the criterion to be achieved.

8.3 Problem evaluation

When we compare the final state of the tool to the needs we identified, we conclude that, while the current implementation of the tool does not reach up to the desired degree of efficiency, it has streamlined the process, and still contains the capability of further improvement on the field.

What we found to be the greatest requirements for increasing the degree of efficiency in a privacy impact methodology is the presence of the knowledge base which LINDDUN makes use of. By introducing automatization of the knowledge base, the flow between the steps of the methodology become more streamline. When such a knowledge base is linked through the entire risk assessment process, we believe that a non-linear risk assessment cycle is possible to achieve.

As was mentioned in 1, the LINDDUN team describes the improved semantics of the knowledge base as a requirement for an increased streamlining of privacy threat modeling [37]. Based on the overall impression we have gotten from this work, we would like to suggest a potential addition to the knowledge base. It is our experience that the greatest obstruction in creating a full solution is the lack of possible automatization of threat scenarios. As such, we would suggest the implementation of threat scenarios as part of the knowledgebase. A suggestion for how it would be implemented in LARA is found in section 9.2.

Chapter 9

Conclusion

9.1 Thesis conclusion

The world has come far regarding ubiquitous technology in the last few decades, and so has the need for methods to handle the rights and responsibilities which results from it. With the introduction of the GDPR, this need has been given a shape.

This thesis work came to be due to the unmet need for an increase in efficiency in the performance of privacy risk assessment methodologies. We identified the needs of such a tool to be a non-linear incrementality, aided by automatization to ease the process further. The resulting tool, *LARA*, is based on a version of the LINDDUN methodology, and is to a high degree capable of reproducing the results of the manual methodology. LARA's greatest contribution towards meeting the needs formulated are the automatization of the threat refinement. Along with the vulnerability exclusion form and automated threat mapping table, it aids in reducing the linearity of the risk assessment. While the validity and reliability of our evaluation results could have been higher, our work have led us to believe that the requirements for a more efficient streamlining lies in automatization through the knowledge base.

9.2 Future Work

Due to the time limitations for this thesis, this section includes methods for improving the functionality of the tool and its evaluation.

Sections of evaluation

As discussed in the evaluation of the use cases, the greatest weakness of current implementation of the tool is the lack of user-defined sections of evaluation.

Currently, the threat refinement makes use of internal logic to define which elements are a part of a single evaluation. This logic makes sense for the case of a threat in a part of the system propagating itself into other parts. However, just having that logic results in the issue seen in fig. 7.1, where a single process, authentication, must be present twice in order to prevent the logic error of connecting the Patient to the Web App. By implementing a sectioning function, where the user overrides the default logic and defines which DFD elements are part of a single interaction, this problem can be solved.

Generalizing assumptions

Another weakness discussed in the use case evaluation 8.1.1 was the lack of generalized assumptions which can be applied to the system when multiple DFD elements are assumed to share the same threats and trust.

Inspired by section 6.2, a way this could be implemented would be that if a DFD element is added to such a list, the user would then be unable to alter the threat category of the DFD element further.

This suggested functionality, in addition to the sectioning functionality, would help remove the need for predefined DFD elements.

Pre-define threat scenario templates

As mentioned in section 8.3, one of the greatest limits to the iterability of the tool is the lack of automatization for the threat scenarios once a change in their conditions change.

One suggestion to implement this is the addition of pre-defined threat scenarios to the knowledge base for the user to choose from. Using the

community threat scenario examples [34] as inspiration, the user could choose a scenario which matches the threats generated. In order to specify the threat scenario further, the user should be able to change the title and the flow to reflect that specific scenario. Further, that specific threat scenario could be connected to the specific threats it is based on and could alert the user if a change has been made to the basis for the scenario.

Branch exclusion

As discussed in both the discarded functionality section 6.2, as well as in the use case evaluation 8.1.1, the ability to trust entire sections of a threat tree would be ideal. This feature could be implemented in a way which displays the threat tree for the user as well, such as through multiple level dropdown tables.

Testing

The greatest flaw of the testing done through the thesis work is the lack of it, which is caused by the requirements of those doing the testing. As discussed in section 4.4 and section 8.1.1, a DPIA is both tedious to perform, and requires a privacy expert. As such, any further testing should be done by someone who fulfills those criteria in order to get more accurate results.

Bibliography

[1] Adobe. *Waterfall Methodology: A Complete Guide*. URL: https://business.adobe.com/blog/basics/waterfall (visited on 30/03/2023).

[2] Autoriteit Personsgegevens. *Social Insurance Bank (SVB) fined for inadequate identity checks*. URL: https://www.autoriteitpersoonsgegevens.nl/en/news/social-insurance-bank-svb-fined-inadequate-identity-checks (visited on 11/05/2023).

[3] Commission Nationale de l'Informatique et des Libertés. *Privacy Impact Assessment (PIA)*. URL: https://www.cnil.fr/sites/default/files/atoms/files/cnil-pia-1-en-methodology.pdf (visited on 29/03/2023).

[4] Datatilsynet. *Ferde AS fined*. URL: https://www.datatilsynet.no/en/news/2021/ferde-as-fined/ (visited on 11/05/2023).

[5] DistriNet Research Group. *LINDDUN privacy engineering*. URL: https://www.old.linddun.org/ (visited on 07/05/2023).

[6] DistriNet Research Group. *LINDDUN privacy threat modeling*. URL: https://linddun.org/ (visited on 11/05/2023).

[7] DistriNet Research Group. *Privacy threat trees catalog*. URL: https://www.old.linddun.org/linddun-threat-catalog (visited on 14/05/2023).

[8] European Parliament and Council of the European Union. *Charter of Fundamental Rights of the European Union*. URL: https://eur-lex.e

uropa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:12012P/TXT&from=EN (visited on 10/04/2023).

[9] European Parliament and Council of the European Union. *DIRECTIVE 95/46/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL*. URL: https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:EN:HTML (visited on 10/04/2023).

[10] European Parliament and Council of the European Union. *REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679 (visited on 10/05/2023).

[11] European Union Agency for Cybersecurity. *Evaluating the level of risk for a personal data processing operation*. URL: https://www.enisa.europa.eu/risk-level-tool/risk (visited on 29/03/2023).

[12] Katrine Feten. *LARA: LINDDUN-based Automated Risk Assessment Tool*. Requires a UiO user to access. URL: https://github.uio.no/katrife/MasterProject.

[13] Garante per la protezione dei dati personali. *Ordinanza ingiunzione nei confronti di Eurosanità S.P.A. - 15 dicembre 2022 [9870788]*. URL: https://www.garanteprivacy.it/web/guest/home/docweb/-/docweb-display/docweb/9870788 (visited on 11/05/2023).

[14] GeeksforGeeks. *Depth First Search or DFS for a Graph*. URL: https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/ (visited on 28/03/2023).

[15] Information Commissioner's Office. *Data protection impact assessments*. URL: https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-

gdpr/accountability-and-governance/data-protection-impact-assessments/ (visited on 30/03/2023).

[16] Adrienn Lukács. *What is privacy? The history and definition of privacy.* 2016. URL: http://publicatio.bibl.u-szeged.hu/10794/7/3188699.pdf.

[17] Meta Platforms, Inc. *React.* URL: https://react.dev/ (visited on 15/03/2023).

[18] National Institute of Standards and Technology. *NIST Privacy Framework: A Tool for Improving Privacy through Enterprise Risk Management.* URL: https://www.nist.gov/system/files/documents/2020/01/16/NIST%20Privacy%20Framework_V1.0.pdf (visited on 29/03/2023).

[19] OWASP. *OWASP Threat Dragon.* URL: https://www.threatdragon.com/#/ (visited on 07/05/2023).

[20] OWASP. *Who is the OWASP® Foundation?* URL: https://owasp.org/ (visited on 07/05/2023).

[21] Kara Pernice. *F-Shaped Pattern of Reading on the Web: Misunderstood, But Still Relevant (Even on Mobile).* URL: https://www.nngroup.com/articles/f-shaped-pattern-reading-web-content/ (visited on 24/03/2023).

[22] Proton Technologies AG. *What is GDPR, the EU's new data protection law?* URL: https://gdpr.eu/what-is-gdpr/ (visited on 09/05/2023).

[23] Statista. *Penetration rate of smartphones in Europe from 2013 to 2028.* URL: https://www.statista.com/forecasts/1147144/smartphone-penetration-forecast-in-europe (visited on 21/03/2023).

[24] Tellu. *E-helseplattform.* URL: https://tellu.no/hva-vi-leverer/e-helseplattform/ (visited on 26/04/2023).

[25] Tellu. *TelluCare.* URL: https://tellu.no/tellucare/ (visited on 26/04/2023).

[26] Tellu. *TelluCare Sandbox*. URL: https://tellu.no/tellucare-sandbox/ (visited on 26/04/2023).

[27] The National Science and Media Museum. *A short history of the internet*. URL: https://www.scienceandmediamuseum.org.uk/objects-and-stories/short-history-internet (visited on 14/04/2023).

[28] Daniel Tschinder. *React Tabs*. URL: https://github.com/reactjs/react-tabs (visited on 24/03/2023).

[29] U.S General Services Administration. *Usability Testing*. URL: https://www.usability.gov/how-to-and-tools/methods/usability-testing.html (visited on 14/04/2023).

[30] U.S General Services Administration. *Use Cases*. URL: https://www.usability.gov/how-to-and-tools/methods/use-cases.html (visited on 14/04/2023).

[31] U.S General Services Administration. *User-Centered Design Basics*. URL: https://www.usability.gov/what-and-why/user-centered-design.html (visited on 14/04/2023).

[32] Jed Watson. *React Select*. URL: https://github.com/JedWatson/react-select (visited on 24/03/2023).

[33] Webkid GmbH. *React Flow*. URL: https://reactflow.dev/ (visited on 15/03/2023).

[34] Kim Wuyts. *Patient Community System - Example Privacy Analysis*. URL: https://www.old.linddun.org/_files/ugd/cc602e_b4f5b1fc19da49a9bb8e39f0933cadab.pdf (visited on 09/05/2023).

[35] Kim Wuyts and Wouter Joosen. *LINDDUN privacy threat modeling: a tutorial*. 2015. URL: https://www.cs.kuleuven.be/publicaties/rapporten/cw/CW685.pdf.

[36] Kim Wuyts, Riccardo Scandariato and Wouter Joosen. *LIND (D) UN privacy threat tree catalog*. 2014. URL: https://www.cs.kuleuven.be/publicaties/rapporten/cw/CW675.pdf.

[37] Kim Wuyts, Laurens Sion, Dimitri Van Landuyt and Wouter Joosen. "Knowledge is power: Systematic reuse of privacy knowledge for threat elicitation". In: *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE. 2019, pp. 80–83.

[38] Zenkit Blog. *Waterfall Methodology: A Complete Guide*. URL: `https://zenkit.com/en/blog/agile-methodology-an-overview/` (visited on 30/03/2023).

Appendices

Appendix A

Tool components

A.1 DFD modeling

DFD modeling tool: fig. 5.5

As described in section 2.2.1, A DFD has four types of data elements: entity, process, data store, and data flow.

There are tree different form of interactions the user can have with the DFD itself:

1. Creating DFD elements
    - Entities, processes and data stores are added to the DFD by selecting their respective buttons.
    - Data flows are added by dragging a line from one DFD element to another, from a dark source to a light source.
2. Importing and exporting the current work
    - When the export button is selected, an alert will appear on screen, informing the user that the current work has been added to the computer clipboard, and to save the diagram to a *.json file*, though a *.txt* file type is fine as well. The current work includes the DFD, the parameters set for the DFD elements, as well as any threat scenarios and solutions.

- The import button opens the file explorer of the computer, prompting the user to pick a file to be import into the tool. The file contents will then overwrite the current data stored in the tool.

3. Deleting a DFD element

    - A DFD element is deleted when focusing on the element, which is done by selecting it, then selecting the backspace button on the keyboard.

Parameter form: fig. 5.6

All DFD elements contain additional parameters, which the user may assign according to their preferences.

- All DFD elements can be given a label, which is displayed as the element label.

    – All DFD elements, with the exception of the data flow elements, are initially given their unique ID as their label when generated, in order to provide a unique label before a new one is assigned. Because the label is not required to be unique, the ID is displayed beneath the label for identification.

- All DFD elements can be attributed with (0 ... *) LINDDUN privacy threats through the use of checkboxes. If a checkbox is unmarked, it is the same as assuming that the threat don't apply for the DFD element.

    – Each checkbox, once marked, have a dropdown list connected to it, which contains a list of privacy vulnerabilities that directly affect the DFD element, and may cause the materialization of the threat. By selecting a vulnerability, the user makes the assumption that this vulnerability does not apply for the DFD element. (0 ... *) vulnerabilities may be selected.

- All DFD elements come with a comment field, in which the user may enter any information they desire. The intended use of this field is to document decisions made regarding the DFD element.

- The border color of a DFD element can have its color changed to a selection of predefined colors. The intended use is to create a distinction between sections of the program for easier visualization.

A.2 DFD threat mapping and threat refinement

DFD threat mapping table: fig. 5.7

The DFD threat mapping table is a matrix table with the DFD elements and their threats listed in a row horizontally, and with each threat category listed vertically.

The table contains nine columns. The first column is for separating the DFD elements into their type categories, the second for the label of the DFD element, and the rest for each of the seven LINDDUN threats. For each DFD element, an "X" is filled into the column representing a threat, if the threat was marked as possible in the parameter settings. For each row, the DFD elements are grouped by type.

Threat refinement result display: fig. 5.8

Similarly to the display for viewing parameters, the middle display is for viewing the results of the threat refinement and the mitigation strategies found.

A threat having been marked as possible is represented by an "X" in the given threat field, and by selecting the threat for a DFD element, the user starts the threat refinement process. The form of the results outputted depends on the results. If the refinement leads to the conclusion that the threat did not materialize, the outputted message will simply read that no results were found.

If the threat materialized, the vulnerabilities will be displayed in a list, where each item displays the entire path from the tree root to the tree leaf, which is the vulnerability. If a cause is the result of a full or partial threat materializing in one or more DFD elements, or another threat affects the same DFD element, this will be displayed behind each vulnerability.

Below this list is the list of mitigation strategies found for threats on the threat tree. They are grouped for each tree node, with the threat they address at the top, the category of the strategy, then the suggested mitigation strategies to employ.

Additionally, a button appears on the top of the display, which provides the user with the opportunity to transfer the threat tree and found nodes

into a threat scenario template, which is covered in the next section. Upon creating the threat scenario template, the tab is switched to the template view.

A.3 Threat scenario elicitation

Template displays: fig. 5.9

The template display may display two different templates; the threat scenario template and the solution template.

A threat scenario template has the following parameter fields:

- The title of the threat scenario, as well as the title of a solution made with the given threat scenario.
- A summary of the threat scenario.
- A list of assets, stakeholders and threats for the scenario.
- The primary mis-actor of the scenario.
- The basic path of the threat scenario.
- A potential alternate path for the scenario to occur.
- The consequences of the realized scenario.
- A trigger for the scenario to occur.
- Any pre-conditions required for the scenario to occur.
- References to threat tree node(s). This is filled in at threat scenario template creation using the threat tree leaf nodes found in the threat refinement.
  - This field consists of a dropdown table where the user can pick any leaf node from any threat tree.
- Parent threat tree(s). This is filled in at threat scenario template creation using the threat tree which was refined in the threat refinement.
- DFD element(s). This is filled in at threat scenario template creation using the DFD element which was refined in the threat refinement.

- Any remarks about assumptions made with regards to the scenario. May overlap with assumptions made for the DFD elements.
- A button to store the threat scenario to the threat scenario table.

The solution table has the following parameters:

- A title, which corresponds to the title entered in the threat scenario.
- A dropdown for each threat tree node entered into the threat scenario, which contains the mitigation strategies of all threat tree nodes from the given node and towards the root which has a mitigation strategy. Only one strategy may be selected.
- A dropdown for each threat tree node entered into the threat scenario, which contains the associated Privacy Enhancing Technologies (PETs) for the mitigation strategy selected. If no mitigation strategy has been chosen, or the strategy has no corresponding PET, no options will be available to select.
- A button to store the solution template to the solution table.

Threat scenario and solution tables: fig. 5.10

When a threat scenario is saved, it is added to the threat scenario table. Each row of the threat scenario table contains the following elements:

- The title of the threat scenario.
  - If selected on, it will enter the threat scenario into the template display for either viewing or updating the parameters. The "add" button is replaced by the "update" button.
- A button for deleting the given threat scenario.
- A button for creating a solution template with the parameters from the given threat scenario.

When a solution template is saved, it is added to the solution table. Each row of the threat scenario table contains the following elements:

- The title of the solution, which is the title of the template it is based on.
- A button for deleting the given row.

- A row for each threat tree node. Each row contains the following elements:
  - The threat tree node vulnerability.
  - The selected mitigation strategy.
  - The selected PET solution.

Appendix B

Use case results

B.1 LARA results

DFD

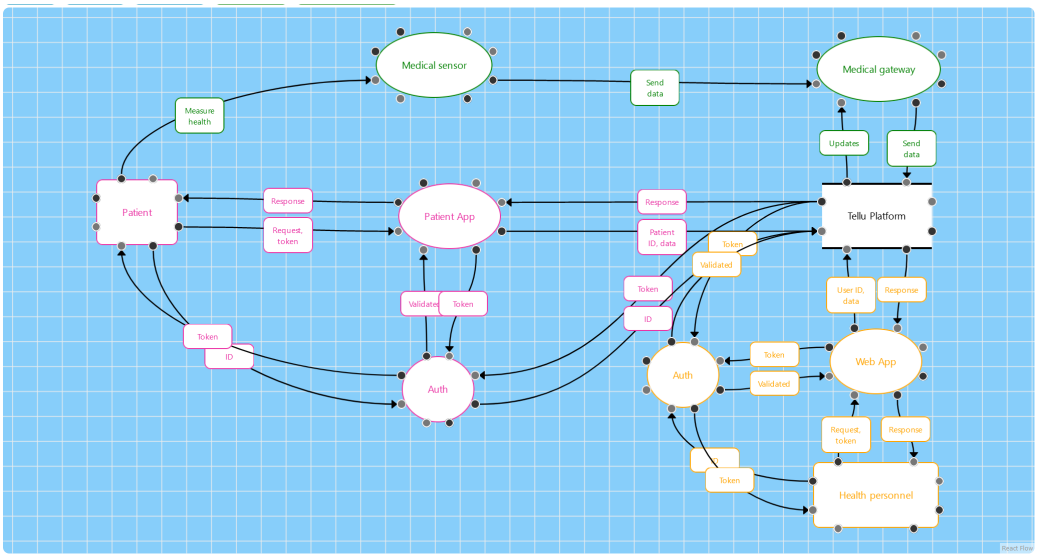

Figure B.1: DFD of the Tellu system.

Assumptions

1. Patient

(a) No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

(b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

(c) Identifiability of patients is not considered a threat, as they all possess a unique identifier, and the use of the system is not considered an issue.

(d) Linkability of patients is not considered a threat, as they all possess a unique identifier, and the use of the system is not considered an issue.

2. Health personnel

(a) No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

(b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

(c) Identifiability of patients is not considered a threat, as they all possess a unique identifier, and the use of the system is not considered an issue.

(d) Linkability of patients is not considered a threat, as they all possess a unique identifier, and the use of the system is not considered an issue.

(e) Unawareness does not apply to health personnel, as it is assumed they receive training in the use of the system and review it regularly.

3. Patient App

(a) No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

(b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

(c) It is assumed that the system complies with its privacy policy, and that it complies with data protection laws.

4. Auth (Pink)

    (a) No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

    (b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

    (c) The authentication process is assumed to be well implemented and secure.

    (d) It is assumed that the system complies with its privacy policy, and that it complies with data protection laws.

5. Web App

    (a) No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

    (b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

    (c) It is assumed that the system complies with its privacy policy, and that it complies with data protection laws.

6. Medical sensor

    (a) No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

    (b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

    (c) It is assumed that the system complies with its privacy policy, and that it complies with data protection laws.

7. Medical gateway

    (a) No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

    (b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

    (c) It is assumed that the system complies with its privacy policy, and

that it complies with data protection laws.

8. Auth (Orange)

   (a) No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

   (b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

   (c) The authentication process is assumed to be well implemented and secure.

   (d) It is assumed that the system complies with its privacy policy, and that it complies with data protection laws.

9. Tellu Platform

   (a) No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

   (b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

   (c) The data store is considered confidential as access control is present.

   (d) It is assumed that the data store is sufficiently protected from side-channel attacks, extra-monitor and bad storage management.

   (e) Identifiability and linkability are applicable to the data store, and will therefore be examined.

   (f) It is assumed that the system complies with its privacy policy, and that it complies with data protection laws.

10. Tellu Platform $\leftrightarrow$ Patient App

    (a) No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

    (b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

    (c) Data flows between the data store and processes are considered trusted, as it is assumed the communication occurs over a secure

communication line.

(d) It is assumed that the system complies with its privacy policy, and that it complies with data protection laws.

11. Patient ↔ Patient App

(a) No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

(b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

(c) Data flows between an entity and an app process is not trusted, as it involves communication over an insecure communication line.

(d) Linkability and identifiability are not a threat to the data flows between entities and app processes because of assumptions about patients and health personnel.

(e) As side-channel attacks require a lot of analysis, it is assumed to be unlikely for them to occur at data flows.

(f) It is assumed that the system complies with its privacy policy, and that it complies with data protection laws.

12. Auth ↔ Patient

(a) No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

(b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

(c) Data flows between the authentication are considered trusted, as it is assumed the communication occurs over a secure communication line.

(d) It is assumed that the system complies with its privacy policy, and that it complies with data protection laws.

13. Auth ↔ Tellu Platform (Pink)

(a) No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

(b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

(c) Data flows between the authentication are considered trusted, as it is assumed the communication occurs over a secure communication line.

(d) It is assumed that the system complies with its privacy policy, and that it complies with data protection laws.

14. Patient ↔ Medical sensor

(a) No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

(b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

(c) The data flow between a patient and a sensor is local and thus irrelevant for the system.

(d) It is assumed that the system complies with its privacy policy, and that it complies with data protection laws.

15. Medical sensor ↔ Medical gateway

(a) No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

(b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

(c) Data flows between the medical sensors and a medical gateway is not trusted, as it involves communication over an insecure communication line.

(d) As side-channel attacks require a lot of analysis, it is assumed to be unlikely for them to occur at data flows.

(e) It is assumed that the system complies with its privacy policy, and that it complies with data protection laws.

16. Tellu Platform ↔ Medical gateway

(a) No Non-repudiation threats exist in the system, as the data flows,

processes and data stores do not require plausible deniability.

(b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

(c) Data flows between the data store and processes are considered trusted, as it is assumed the communication occurs over a secure communication line.

(d) It is assumed that the system complies with its privacy policy, and that it complies with data protection laws.

17. Tellu Platform ↔ Web App

(a) No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

(b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

(c) Data flows between the data store and processes are considered trusted, as it is assumed the communication occurs over a secure communication line.

(d) It is assumed that the system complies with its privacy policy, and that it complies with data protection laws.

18. Health personnel ↔ Web App

(a) No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

(b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

(c) Data flows between an entity and an app process is not trusted, as it involves communication over an insecure communication line.

(d) Linkability and identifiability are not a threat to the data flows between entities and app processes because of assumptions about patients and health personnel.

(e) As side-channel attacks require a lot of analysis, it is assumed to be unlikely for them to occur at data flows.

(f) It is assumed that the system complies with its privacy policy, and that it complies with data protection laws.

19. Auth ↔ Tellu Platform (Orange)

    (a) No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

    (b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

    (c) Data flows between the authentication are considered trusted, as it is assumed the communication occurs over a secure communication line.

    (d) It is assumed that the system complies with its privacy policy, and that it complies with data protection laws.

20. Auth ↔ Health personnel

    (a) No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

    (b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

    (c) Data flows between the authentication are considered trusted, as it is assumed the communication occurs over a secure communication line.

    (d) It is assumed that the system complies with its privacy policy, and that it complies with data protection laws.

21. Auth ↔ Patient App

    (a) No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

    (b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

    (c) Data flows between the authentication are considered trusted, as it is assumed the communication occurs over a secure communication line.

(d) It is assumed that the system complies with its privacy policy, and that it complies with data protection laws.

22. Auth ↔ Web App

    (a) No Non-repudiation threats exist in the system, as the data flows, processes and data stores do not require plausible deniability.

    (b) Detectability is not considered a threat for this system, as the privacy concerns are focused on the data itself.

    (c) Data flows between the authentication are considered trusted, as it is assumed the communication occurs over a secure communication line.

    (d) It is assumed that the system complies with its privacy policy, and that it complies with data protection laws.

DFD threat mapping table

DFD threat mapping table								
Data Type	Target	L	I	NR	D	ID	U	NC
Entity	Patient						X	
Entity	Heath personnel							
Process	Patient App	X	X			X		
Process	Auth							
Process	Web App	X	X			X		
Process	Medical sensor	X	X			X		
Process	Medical gateway	X	X			X		
Process	Auth							
Data Store	Tellu platform	X	X			X		
Data Flow	Patient App → Patient					X		
Data Flow	Tellu platform → Patient App							
Data Flow	Patient App → Tellu platform							
Data Flow	Patient → Patient App					X		
Data Flow	Patient → Auth							
Data Flow	Auth → Patient							
Data Flow	Tellu platform → Auth							
Data Flow	Auth → Tellu platform							
Data Flow	Patient → Medical sensor							
Data Flow	Medical sensor → Medical gateway	X	X			X		
Data Flow	Tellu platform → Medical gateway							
Data Flow	Medical gateway → Tellu platform							
Data Flow	Web App → Tellu platform							
Data Flow	Tellu platform → Web App							
Data Flow	Heath personnel → Web App					X		
Data Flow	Web App → Heath personnel					X		
Data Flow	Tellu platform → Auth							
Data Flow	Auth → Tellu platform							
Data Flow	Heath personnel → Auth							
Data Flow	Auth → Health personnel							
Data Flow	Auth → Patient App							
Data Flow	Patient App → Auth							
Data Flow	Web App → Auth							
Data Flow	Auth → Web App							

Threat refinement results

1. Patient
   (a) Unawareness
      i. No/insufficient feedback and awareness tools (U_3).
      ii. No user-friendly privacy support (U_4).
      iii. Unable to review personal information (data accuracy) (U_5).
2. Health personnel
   (a) No realized threats found.
3. Patient App
   (a) Linkability
      i. Information disclosure of a process (Patient App).
   (b) Identifiability
      i. Information disclosure of a process (Patient App).
   (c) Information Disclosure
      i. Side channels (ID_p2).
      ii. Input validation failure (ID_p3).
      iii. Access to memory (ID_p4).
4. Auth (Pink)
   (a) No realized threats found.
5. Web App
   (a) Linkability
      i. Information disclosure of a process (Web App).
   (b) Identifiability
      i. Information disclosure of a process (Web App).

(c) Information Disclosure

i. Side channels (ID_p2).

ii. Input validation failure (ID_p3).

iii. Access to memory (ID_p4).

6. Medical sensor

(a) Linkability

i. Information disclosure of a process (Medical sensor).

(b) Identifiability

i. Information disclosure of a process (Medical sensor).

(c) Information Disclosure

i. Side channels (ID_p2).

ii. Input validation failure (ID_p3).

iii. Access to memory (ID_p4).

7. Medical gateway

(a) Linkability

i. Information disclosure of a process (Medical gateway).

(b) Identifiability

i. Information disclosure of a process (Medical gateway).

(c) Information Disclosure

i. Side channels (ID_p2).

ii. Input validation failure (ID_p3).

iii. Access to memory (ID_p4).

8. Auth (Orange)

(a) No realized threats found.

9. Tellu Platform
    (a) Linkability
        i. No realized threats found.
    (b) Identifiability
        i. No realized threats found.
    (c) Information Disclosure
        i. Canonicalization failure (ID_ds6).
        ii. No protection (ID_ds7).
        iii. Weak permissions (ID_ds8).
        iv. Other consumers (ID_ds9).
        v. Unencrypted (ID_ds10).
10. Tellu Platform ↔ Patient App
    (a) No realized threats found.
11. Patient ↔ Patient App
    (a) Information Disclosure
        i. No message confidentiality (ID_df4).
        ii. Weak message confidentiality (ID_df5).
        iii. Man-In-The-Middle (ID_df6).
        iv. No channel confidentiality (ID_df7).
12. Auth ↔ Patient
    (a) No realized threats found.
13. Auth ↔ Tellu Platform (Pink)
    (a) No realized threats found.
14. Patient ↔ Medical sensor

(a) No realized threats found.

15. Medical sensor ↔ Medical gateway

    (a) Linkability

        i. Based on IP address (L_df8).

        ii. Based on computer ID (L_df9).

        iii. Based on session ID (L_df10).

        iv. Based on behavioral patterns (time, frequency, location) (L_df11).

        v. Traffic analysis possible (L_df12).

        vi. Active attacks possible (L_df13).

        vii. Passive attacks possible (L_df14).

        viii. Linkability of content (inference at L_DS) (Tellu Platform).

        ix. Information Disclosure of data flow (Medical sensor -> Medical gateway).

    (b) Identifiability

        i. Based on IP address (I_df8).

        ii. Based on computer ID (I_df9).

        iii. Based on session ID (I_df10).

        iv. Based on behavioral patterns (time, frequency, location) (I_df11).

        v. Traffic analysis possible (I_df12).

        vi. Active attacks possible (I_df13).

        vii. Passive attacks possible (I_df14).

        viii. Identifiability of content (weak anonymization at I_DS) (Tellu Platform).

        ix. Information Disclosure of data flow (Medical sensor -> Medical gateway).

(c) Information Disclosure

i. No message confidentiality (ID_df4).

ii. Weak message confidentiality (ID_df5).

iii. Man-In-The-Middle (ID_df6).

iv. No channel confidentiality (ID_df7).

16. Tellu Platform ↔ Medical gateway

(a) No realized threats found.

17. Tellu Platform ↔ Web App

(a) No realized threats found.

18. Health personnel ↔ Web App

(a) Information Disclosure

i. No message confidentiality (ID_df4).

ii. Weak message confidentiality (ID_df5).

iii. Man-In-The-Middle (ID_df6).

iv. No channel confidentiality (ID_df7).

19. Auth ↔ Tellu Platform (Orange)

(a) No realized threats found.

20. Auth ↔ Health personnel

(a) No realized threats found.

21. Auth ↔ Patient App (Pink)

(a) No realized threats found.

22. Auth ↔ Web App (Orange)

(a) No realized threats found.

B.2 Manual threat refinement results

Use case 1

appendix B.1 and appendix B.1 also apply for the manual refinement.

Patient

Based on the assumptions, the following threats and vulnerabilities are removed:

1. Linkability is removed as a threat.
2. Identifiability is removed as a threat.

As such, it can be assumed that the threat(s) are:

1. Unawareness by U_3, U_4 and U_5.

Health personnel

Based on the assumptions, the following threats and vulnerabilities are removed:

1. Linkability is removed as a threat.
2. Identifiability is removed as a threat.
3. Unwareness is removed as a threat.

As such, it can be assumed that there are no threats.

Tellu platform

Based on the assumptions, the following threats and vulnerabilities are removed:

1. Weak access control to database is removed as a vulnerability for linkability.
2. Weak access control to database is removed as a vulnerability for identifiability.
3. Non-repudiation is removed as a threat.
4. Detectability is removed as a threat.
5. Side-channels, extra-monitor access, and storage management are removed as vulnerabilities.

6.Non-compliance is removed as a threat.

As such, it can be assumed that the threat(s) are:

1.Information disclosure by ID_ds6, ID_ds7, ID_ds8, ID_ds9 and ID_ds10.

Patient App, Web App

Based on the assumptions, the following threats and vulnerabilities are removed:

1.Non-repudiation is removed as a threat.

2.Detectability is removed as a threat.

3.Non-compliance is removed as a threat.

As such, it can be assumed that the threat(s) are:

1.Information disclosure by ID_p2, ID_p3 and ID_p4.

2.Linkability as a result of ID_p.

3.Identifiability as a result of ID_p.

Medical sensor

Based on the assumptions, the following threats and vulnerabilities are removed:

1.Non-repudiation is removed as a threat.

2.Detectability is removed as a threat.

3.Non-compliance is removed as a threat.

As such, it can be assumed that the threat(s) are:

1.Information disclosure by ID_p2, ID_p3 and ID_p4.

2.Linkability as a result of ID_p.

3.Identifiability as a result of ID_p.

Medical gateway

Based on the assumptions, the following threats and vulnerabilities are removed:

1.Non-repudiation is removed as a threat.

2.Detectability is removed as a threat.

3.Non-compliance is removed as a threat.

As such, it can be assumed that the threat(s) are:

1.Information disclosure by ID_p2, ID_p3 and ID_p4.

2.Linkability as a result of ID_p.

3.Identifiability as a result of ID_p.

Authentication

Based on the assumptions, the following threats and vulnerabilities are removed:

1.Linkability is removed as a threat.

2.Identifiability is removed as a threat.

3.Non-repudiation is removed as a threat.

4.Detectability is removed as a threat.

5.Disclosure of Information is removed as a threat.

6.Non-compliance is removed as a threat.

As such, it can be assumed that there are no threats.

Tellu platform ↔ Medical gateway, Patient App, Web App

Based on the assumptions, the following threats and vulnerabilities are removed:

1.Linkability is removed as a threat.

2.Identifiability is removed as a threat.

3.Non-repudiation is removed as a threat.

4.Detectability is removed as a threat.

5.Information Disclosure is removed as a threat.

6.Non-compliance is removed as a threat.

As such, it can be assumed that there are no threats.

Patient App, Web App ↔ Patient, Health personnel

Based on the assumptions, the following threats and vulnerabilities are removed:

1.Linkability is removed as a threat.

2.Identifiability is removed as a threat.

3.Non-repudiation is removed as a threat.

4.Detectability is removed as a threat.

5.Non-compliance is removed as a threat.

As such, it can be assumed that the threat(s) are:

1.Information disclosure by ID_df4, ID_df5, ID_df6 and ID_df7.

Authentication ↔ Tellu Platform

Based on the assumptions, the following threats and vulnerabilities are removed:

1.Linkability is removed as a threat.

2.Identifiability is removed as a threat.

3.Non-repudiation is removed as a threat.

4.Detectability is removed as a threat.

5.Information Disclosure is removed as a threat.

6.Non-compliance is removed as a threat.

As such, it can be assumed that there are no threats.

Authentication ↔ Patient App, Web App

Based on the assumptions, the following threats and vulnerabilities are removed:

1.Linkability is removed as a threat.

2.Identifiability is removed as a threat.

3.Non-repudiation is removed as a threat.

4.Detectability is removed as a threat.

5.Information Disclosure is removed as a threat.

6.Non-compliance is removed as a threat.

As such, it can be assumed that there are no threats.

Authentication ↔ Patient, Health personnel

Based on the assumptions, the following threats and vulnerabilities are removed:

1.Linkability is removed as a threat.

2.Identifiability is removed as a threat.

3.Non-repudiation is removed as a threat.

4.Detectability is removed as a threat.

5.Information Disclosure is removed as a threat.

6.Non-compliance is removed as a threat.

As such, it can be assumed that there are no threats.

Medical sensor → Medical gateway

Based on the assumptions, the following threats and vulnerabilities are removed:

1.Non-repudiation is removed as a threat.

2.Detectability is removed as a threat.

3.Non-compliance is removed as a threat.

As such, it can be assumed that the threat(s) are:

1.Information disclosure by ID_df4, ID_df5, ID_df6 and ID_df7.

2.Linkability by L_df8, L_df9, L_df10, L_df11, L_df12, L_df13, L_df14, L_ds2, ID_df.

3.Identifiability by I_df8, I_df9, I_df10, I_df11, I_df12, I_df13, I_df14, I_ds2, ID_df.

Use case 2

3rd party service

Based on the assumptions, the following threats and vulnerabilities are removed:

1. Linkability is removed as a threat.
2. Identifiability is removed as a threat.
3. Unwareness is removed as a threat.

As such, it can be assumed that there are no threats.

Tellu platform

Based on the assumptions, the following threats and vulnerabilities are removed:

1. Weak access control to database is removed as a vulnerability for linkability.
2. Weak access control to database is removed as a vulnerability for identifiability.
3. Non-repudiation is removed as a threat.
4. Detectability is removed as a threat.
5. Side-channels, extra-monitor access, unencrypted data, and storage management are removed as vulnerabilities.
6. Non-compliance is removed as a threat.

As such, it can be assumed that there are no threats.

Tellu platform $\leftrightarrow$ API

Based on the assumptions, the following threats and vulnerabilities are removed:

1. Linkability is removed as a threat.
2. Identifiability is removed as a threat.
3. Non-repudiation is removed as a threat.
4. Detectability is removed as a threat.

5.Information Disclosure is removed as a threat.

6.Non-compliance is removed as a threat.

As such, it can be assumed that there are no threats.

API ↔ 3rd party service

Based on the assumptions, the following threats and vulnerabilities are removed:

1.Non-repudiation is removed as a threat.

2.Detectability is removed as a threat.

3.Non-compliance is removed as a threat.

As such, it can be assumed that the threat(s) are:

1.Information disclosure by ID_df4, ID_df5, ID_df6 and ID_df7.

2.Linkability by L_df8, L_df9, L_df10, L_df11, L_df12, L_df13, L_df14, L_ds2, ID_df.

3.Identifiability by I_df8, I_df9, I_df10, I_df11, I_df12, I_df13, I_df14, I_ds2, ID_df.

Appendix C

Usability test

Scenario

A newspaper company has a system. The system consists of a data store where news articles are stored, an interface for readers to connect to, a representative user, and data flows between them.

Task 1

Model the system data flow. The dataflow model should consist of 3 DFD (data flow diagram) elements:

- 1 *data store* named "**DB**".
- 1 *process* named "**UI**".
- 1 *entity* named "**Reader**".

Double click the node to edit.

Click "Apply" after each step.

Task 2

Four data flows between the three DFD elements:

- "DB" provides data to "UI".

- "UI" provides news to "Reader".
- "Reader" gives requests to "UI".
- "UI" sends queries to "DB".

Task 3

Mark privacy risks on the DFD (data flow diagram) element.

- **DB**: Set the data store as susceptible to **linkability**, **identifiability**, and **disclosure of information**.
  - *Exclude* "linkable to other databases" from refinement at linkability.

Task 4

Mark the DFD links.

- Data flow from DB to UI: They are considered trusted with no threats applicable.
- Data flow from Reader to UI: Set the data flow as susceptible to **linkability**.

Task 5

Switch the tab to "DFD mapping and threat refinement" at the top of the page.

Refine the linkability ("**L**") threat for DB by clicking the "X" in the DFD threat refinement mapping table.

Task 6

Create a **threat scenario** using the threat refinement results.

The threat scenario should contain the following title: The database stores too much data.

Add the threat scenario, by clicking the button "add template" at the bottom of the template.

Task 7

Edit the scenario (interactive/incremental privacy analysis) by clicking on the title in the template table. Change the title to: "The database stores too much data about sources".

Task 8

Create a solution for the threat scenario by clicking on the "create solution" button for the scenario in the template table.

The data retention vulnerability should have the mitigation of generalize, and the solution of I-diversity.

Add the solution.