LINDDUN

PRIVACY THREAT MODELING

Tutorial

LINDDUN tutorial

Contents

Introduction

Privacy is gaining importance in software systems and software design. We therefore propose the LINDDUN threat modeling methodology which enables integration of privacy early on the development lifecycle.

This document provides a step-by-step tutorial of the LINDDUN methodology.

Before getting started, this section gives a brief summary of the LINDDUN methodology and describes what you can expect from this tutorial.

Privacy Necessity

With privacy becoming a key concern in modern society, it is important that privacy measures are strongly incorporated whenever digital data are involved. Unfortunately, privacy is often neglected when engineering software systems and only introduced as an afterthought. Retrofitting privacy concerns in an existing system is however not straightforward.

In recent years, a different attitude towards privacy has emerged, which is known as *'Privacy by Design.'* One of its core principles states that privacy should be embedded in the early stages of the software development lifecycle, rather than having it as an add-on. Hence, privacy should become an essential component in the software development process.

LINDDUN privacy threat modeling methodology

LINDDUN adheres to the Privacy by Design paradigm as it assists software engineers with limited privacy expertise to introduce privacy early on in the software development lifecycle.

> LINDDUN helps software engineers with limited privacy expertise to introduce privacy early on in the software development lifecycle

LINDDUN is a ***model-based*** approach that leverages a data flow diagram (DFD) as representation of the system to be analyzed. This DFD will serve as basis for the analysis, as each of its elements is systematically examined thoroughly for privacy threats. The methodology is also ***knowledge-based***. It provides an overview of the most common attack paths associated with the set of privacy threat categories contained in the acronym LINDDUN (Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of information, Unawareness, Non-compliance). The attack paths are represented as threat trees that detail possible causes of threats. Each tree presents the attack paths that are related to one threat category which is applied to one particular DFD element type (entity, data flow, data store, or process).

LINDDUN in a nutshell

LINDDUN consists of six steps, as illustrated in Figure 1. The first three steps are considered the core LINDDUN steps, as they focus on the problem space and aim at identifying the privacy threats in the system. The three remaining steps are more solution-oriented and aim at translating the threats that were identified into viable privacy strategies and solutions that can mitigate the threats.

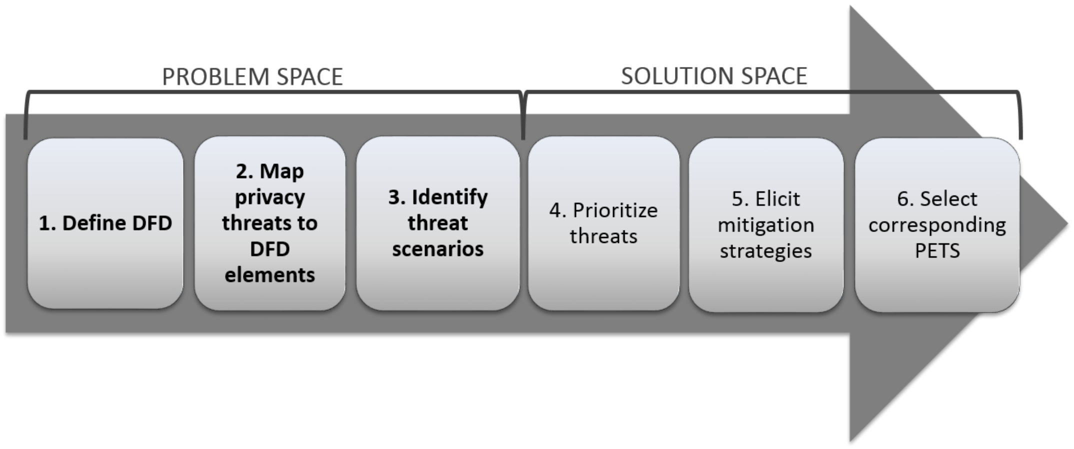

PROBLEM SPACE
SOLUTION SPACE
1. Define DFD
2. Map privacy threats to DFD elements
3. Identify threat scenarios
4. Prioritize threats
5. Elicit mitigation strategies
6. Select corresponding PETS

FIGURE 1: LINDDUN METHODOLOGY STEPS

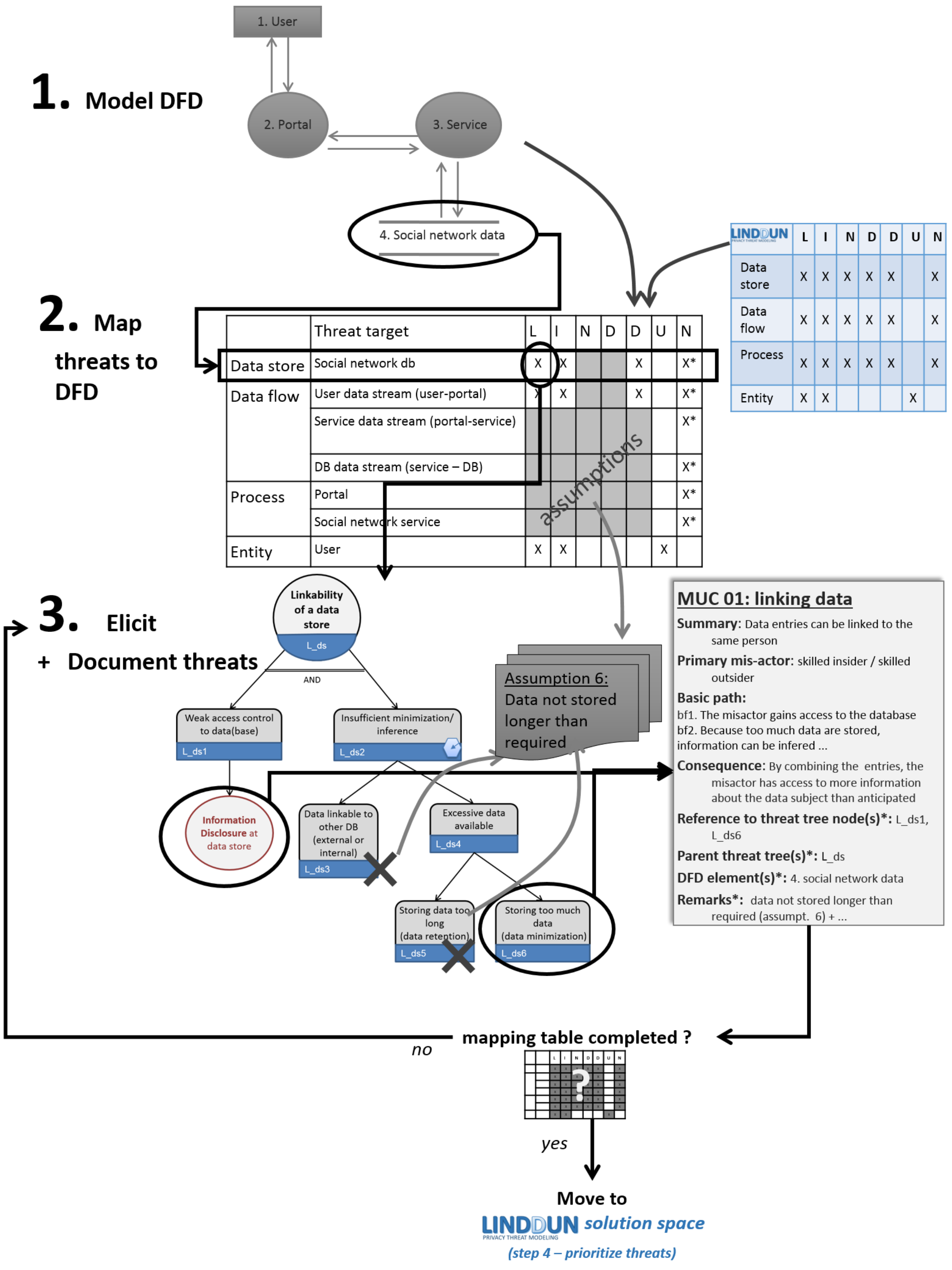
A more schematic overview of the different LINDDUN steps applied to an example is depicted in Figure 2 (problem space) and Figure 3 (solution space).

Problem-oriented steps

First of all, a data flow diagram is created based on system description, which can be either high-level at the requirements phase or a more detailed architecture *(step 1)*.

This is followed by mapping privacy threats to the DFD elements *(step 2)* using the LINDDUN mapping table template shown in step 2 of Figure 2 as a guide to determine the corresponding threats.

In particular, a number of privacy threat tree patterns are presented in the catalog of step 3 of the method to detail the privacy threat instances in a structured fashion, by providing an overview of the most common preconditions of each threat. The identified privacy threats that are relevant to the designated system are documented as misuse cases *(step 3)*.


1. Model DFD
1. User
2. Portal
3. Service
4. Social network data
2. Map threats to DFD
assumptions
3. Elicit + Document threats
Linkability of a data store
L_ds
AND
Weak access control to data(base)
L_ds1
Insufficient minimization/ inference
L_ds2
Information Disclosure at data store
Data linkable to other DB (external or internal)
L_ds3
Excessive data available
L_ds4
Storing data too long (data retention)
L_ds5
Storing too much data (data minimization)
L_ds6
Assumption 6: Data not stored longer than required
mapping table completed ?
no
yes
Move to LINDDUN solution space
(step 4 – prioritize threats)

LINDDUN	L	I	N	D	D	U	N
Data store	X	X	X	X	X		X
Data flow	X	X	X	X	X		X
Process	X	X	X	X	X		X
Entity	X	X				X	

	Threat target	L	I	N	D	D	U	N
Data store	Social network db	X	X	▨	▨	X		X*
Data flow	User data stream (user-portal)	X	X	▨	▨	X		X*
	Service data stream (portal-service)	▨	▨	▨	▨	▨		X*
	DB data stream (service – DB)	▨	▨	▨	▨	▨		X*
Process	Portal	▨	▨	▨	▨	▨		X*
	Social network service	▨	▨	▨	▨	▨		X*
Entity	User	X	X				X	

MUC 01: linking data

Summary: Data entries can be linked to the same person

Primary mis-actor: skilled insider / skilled outsider

Basic path:
bf1. The misactor gains access to the database
bf2. Because too much data are stored, information can be infered …

Consequence: By combining the entries, the misactor has access to more information about the data subject than anticipated

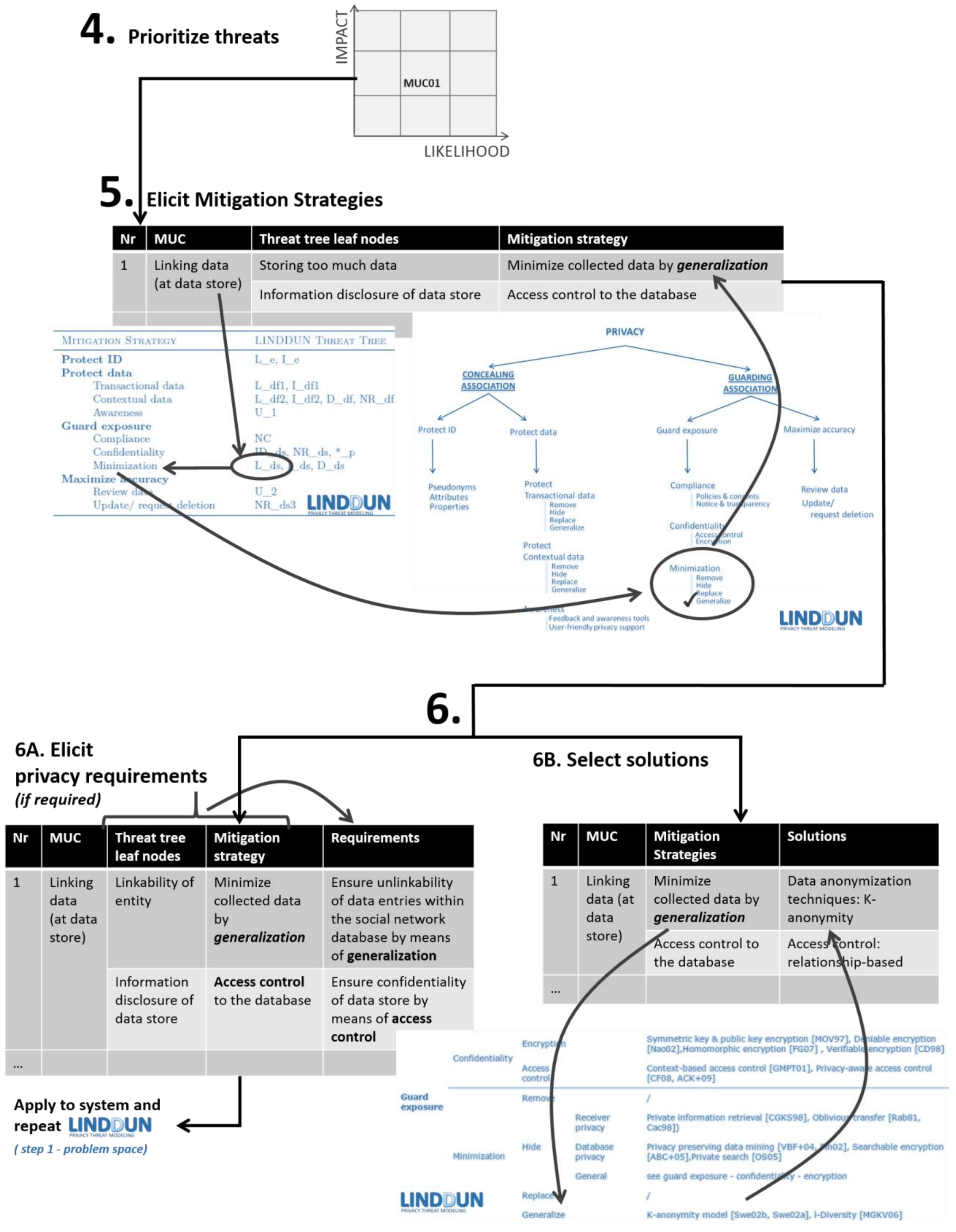
Reference to threat tree node(s)*: L_ds1, L_ds6

Parent threat tree(s)*: L_ds

DFD element(s)*: 4. social network data

Remarks*: data not stored longer than required (assumpt. 6) + …

FIGURE 2: EXAMPLE OF FIRST 3 LINDDUN STEPS (PROBLEM SPACE)



Nr	MUC	Threat tree leaf nodes	Mitigation strategy
1	Linking data (at data store)	Storing too much data	Minimize collected data by ***generalization***
		Information disclosure of data store	Access control to the database

Mitigation Strategy	LINDDUN Threat Tree
Protect ID	L_e, I_e
Protect data	
Transactional data	L_df1, I_df1
Contextual data	L_df2, I_df2, D_df, NR_df
Awareness	U_1
Guard exposure	
Compliance	NC
Confidentiality	ID_ds, NR_ds, *_p
Minimization	L_ds, I_ds, D_ds
Maximize accuracy	
Review data	U_2
Update/ request deletion	NR_ds3

Nr	MUC	Threat tree leaf nodes	Mitigation strategy	Requirements
1	Linking data (at data store)	Linkability of entity	Minimize collected data by ***generalization***	Ensure unlinkability of data entries within the social network database by means of **generalization**
		Information disclosure of data store	**Access control** to the database	Ensure confidentiality of data store by means of **access control**
…				

Nr	MUC	Mitigation Strategies	Solutions
1	Linking data (at data store)	Minimize collected data by ***generalization***	Data anonymization techniques: K-anonymity
		Access control to the database	Access control: relationship-based
…			

Guard exposure	Confidentiality	Encryption		Symmetric key & public key encryption [MOV97], Deniable encryption [Nao02],Homomorphic encryption [FG07] , Verifiable encryption [CD98]
		Access control		Context-based access control [GMPT01], Privacy-aware access control [CF08, ACK+09]
	Minimization	Remove		/
		Hide	Receiver privacy	Private information retrieval [CGKS98], Oblivious transfer [Rab81, Cac98])
			Database privacy	Privacy preserving data mining [VBF+04, Pin02], Searchable encryption [ABC+05],Private search [OS05]
			General	see guard exposure - confidentiality - encryption
		Replace		/
		Generalize		K-anonymity model [Swe02b, Swe02a], l-Diversity [MGKV06]

FIGURE 3: EXAMPLE OF FINAL 3 LINDDUN STEPS (SOLUTION SPACE)

Solution-oriented steps
Furthermore, all the potential privacy threats that are suggested by the privacy threat trees are evaluated and prioritized via risk assessment *(step 4)*. Note that LINDDUN does not provide explicit risk analysis support. We merely refer the analyst to established risk assessment techniques.

Hereafter, in order of importance determined by the prioritization step, the suitable mitigation strategy for each threat is determined *(step 5)*.

Finally, the classification of privacy enhancing technologies according to the mitigation strategies to which they adhere enables a more focused selection of suitable privacy enhancing solutions *(step 6)*. If required, the mitigation strategies can also be translated into privacy requirements, instead of being directly implemented as solutions.

A detailed step-by-step tutorial can be found in Section LINDDUN step-by-step.

LINDDUN tutorial

Intended audience

"LINDDUN is, in many ways, one of the most serious and thought-provoking approaches to privacy threat modeling, and those seriously interested in privacy should take a look at it."

- Adam Shostack
Threat Modeling: Designing for Security, Wiley, 2014
(Chapter 6)

This tutorial is mainly aimed at **software analysts** who want to include privacy into their system analysis.

Nevertheless, this document is also useful for *executives* who want to get a better understanding of privacy and LINDDUN. For these readers, we refer to Section LINDDUN in a nutshell.

Target software systems

LINDDUN is a generic privacy threat modeling technique that does not include any domain-specific knowledge[1]. It can therefore be applied to all software systems. Note however that it mainly focuses on the privacy of the data subject (i.e. the person the data are about). Rather than focusing on internal processes and flows, the analysis will mainly look for threats related to information flowing from the system to external entities.

LINDDUN adheres to the **Privacy by Design** paradigm[2]. It therefore primarily focuses **on software systems under development** in order to include privacy as early as possible in the development process. Nevertheless, LINDDUN can also be applied to **existing software systems for privacy risk assessment** by identifying the applicable privacy threats. Evidently, some of the proposed solutions will be less straightforward to implement for an existing system, as they can also impact the system's architecture.

Getting started

LINDDUN requires at least a basic understanding of the system to analyze. This implies that **at least the requirements of the system-under-development need to be determined** and an abstract view of the system (e.g. a high-level sketch) should exist.

Evidently, the more profound the understanding of the system is, the more detailed and specific the identified privacy threats can be. It is nevertheless important to consider privacy as early as possible during the development lifecycle, thus a *trade-off* will need to be made between early privacy analysis at the requirements level which will result in more generic threats, and a more detailed privacy analysis based on a full-fletched architecture which might introduce issues that should have been mitigated earlier on. We therefore advise to apply LINDDUN multiple times during the development lifecycle to iteratively identify and mitigate privacy issues.

Apply LINDDUN threat analysis multiple times throughout the software development lifecycle.

We suggest to apply LINDDUN already a first time to the high-level software system description at requirements level and use these initial high-level results to design a software architecture with a solid privacy foundation. Once the architectural description is stable, a second LINDDUN iteration can be executed to elicit more detailed, fine-grained privacy threats based on the extensive architectural documentation. These threats will result in an updated privacy-aware architecture to which, if desired, the LINDDUN methodology can be applied once more.

[1] LINDDUN can however be easily extended with domain- or application specific knowledge when required
[2] LINDDUN follows the Privacy by Design principle *from a technical perspective* (i.e. integrating privacy as early as possible in the development lifecycle). It currently does not (fully) integrate data protection objectives from a legal perspective.

LINDDUN can be applied both at the requirements phase and the architecture phase of the software system under development.

At requirements level
Clearly, no full-fledged system description is available during the requirements engineering stage. A high-level overview of the system and its internal and external interactions is however a minimal requirement for the LINDDUN methodology. This system overview will be translated into a data flow diagram that will be used as starting point for the privacy analysis. Obviously, the more detailed the system description is, the more detailed the analysis can be.

At architecture level
A fine-grained architectural description will result in the elicitation of more fine-grained privacy threats.
The architectural description should at least contain a detailed graphical representation of the software system. Also, the types of data that are used and communicated within the system should be documented. General usage scenarios should be documented (e.g. as use cases or sequence diagrams). We refer to Section Creating a DFD for more information about the translation of an architectural description to the required data flow diagram.

Existing systems
When analyzing an existing software system, the detailed architectural description of the system should be used as starting point. Clearly, the more detail can be provided to the analysis, the more fine-grained and system-specific the resulting privacy threats will be.

Outcome

When only applying the problem-oriented steps of LINDDUN, the methodology will result in an elicitation of privacy threats (which can be used for risk assessment). When applying the full methodology, including the final three solution-oriented step, the LINDDUN methodology will results in a selecting of mitigation strategies and corresponding privacy enhancing solutions.

Focus on privacy threats (risk assessment)
LINDDUN is primarily a privacy threat modeling methodology. When focusing on the first three problem-oriented steps of the method (as depicted in Figure 2), LINDDUN results in an elicitation of privacy threats. These threats can then be further examined and mitigated.

Move into solution space
In order to provide support to mitigate the elicited threats, LINDDUN also provides a number of solution-oriented steps (see Figure 3) that help determine the appropriate mitigation strategies and corresponding privacy enhancing technologies to alleviate the identified threats.

Privacy background

This section provides an overview of each of the 7 main privacy threat categories that are handled in LINDDUN. For each threat category, a definition is provided, together with a more detailed explanation and some examples. Also links to related work on the topic are provided when possible.

We refer to the LINDDUN website for the catalog of privacy threat threats (as used in *Step 3* of the methodology) that correspond to the threat categories presented in this section.

Linkability

Definition

The definition of linkability is based on the definition of unlinkability by Pfitzmann and Hansen [1].

Linkability

Being able to sufficiently distinguish whether 2 IOI (items of interest) are linked or not, even *without* knowing the actual identity of the subject of the linkable IOI.

Not being able to hide the link between two or more actions/identities/pieces of information.

Explanation

Linkability on its own is not necessarily a privacy issue. It can only result in severe privacy issues, when linkable data leads to identification (see Identifiability) or inference.

Leading to identification

Anonymous (or de-identified) data, are data of which the identity of the data subject cannot be deduced. For example, instead of using a person's full name and address, using a pseudonym and only the person's street name (no number). This makes the set of subjects that correspond to this pseudonym equal to the number of people living in that particular street.

It is however often possible to link several (pseudo-)anonymous data items together. For example, when the pseudonym of the previous example is linked to additional information (e.g. is the person male or female, in what year is he or she born, etc.), the anonymity set (i.e. the set of subjects corresponding to that pseudonym) becomes much smaller. Clearly, the more information is linked, the more specific this information becomes, and the smaller the anonymity set. When so much information is linked that the anonymity set consists of just one subject, this subject is identifiable[3].

Other examples include: anonymous letters written by the same person, web page visits by the same user, entries in two databases related to the same person.

Leading to inference

Often linkability involves IOIs that are linked because they belong to the same subject, however, IOIs can also be linked based on properties.

Inference is actually a technique that is applied to linkable data instead of a cause of linkability. In contrast to linkability threats that can lead to identifiability, inference is not limited to linking data that belongs to the same person. Inference will *deduce relationships from certain related properties* (e.g. people in the same location, people with the same disease, etc.) and try to generalize them. Inference can be used in a rather innocent fashion to determine the best way to organize groceries in a grocery store (e.g. people who buy hamburgers usually buy buns at the same time, hence they are stored close to each other), but can also have a more judgmental nature if it is used to discriminate a certain population (e.g. people living in a certain neighborhood have a higher chance of cancer, hence their health insurance fee is higher than the surrounding cities). Inference can thus lead to societal harm.

Linkability and inference are however very related and the impact of inference should not be underestimated.

Identifiability

Definition

Identifiability is related to anonymity and pseudonymity as defined by Pfitzmann and Hansen [1].

Identifiability

Being able to sufficiently identify the subject within a set of subjects (i.e. the anonymity set).

Not being able to hide the link between the identity and the IOI (an action or piece of information).

[3] Anonymity and identifiability are further discussed in Section Identifiability. More information about anonymity and anonymity sets is available in the work of Pfitzmann and Hansen [1].

Anonymity refers to hiding the link between an identity and an action or a piece of information. Examples are the anonymous sender of an email, writer of a text, person accessing a service, person to whom an entry in a database relates, and so on. Anonymity is defined as: "Anonymity of a subject from an attacker's perspective means that the attacker cannot sufficiently identify the subject within a set of subjects, the anonymity set" [1].

Anonymity can also be described in terms of unlinkability. If one considers sending and receiving of messages as attributes, the items of interest (IOIs) are who has sent or received which message. Then, "anonymity of a subject with respect to an attribute may be defined as unlinkability of this subject and this attribute." For instance, sender anonymity of a subject means that to this potentially sending subject, each message is unlinkable. Correspondingly, recipient anonymity of a subject means that to this potentially receiving subject, each message is unlinkable.

Pseudonymity suggests that it is possible to build a reputation on a pseudonym and use multiple pseudonyms for different purposes. Examples include a person publishing comments on social network sites under different pseudonyms and a person using a pseudonym to subscribe to a service.

Pfitzmann and Hansen [1] define pseudonymity as: "A pseudonym is an identifier of a subject other than one of the subject's real names. Pseudonymity is the use of pseudonyms as identifiers. A subject is pseudonymous if a pseudonym is used as identifier instead of one of its real names." Pseudonymity can also be perceived with respect to linkability. Whereas anonymity and identifiability (or accountability) are the extremes with respect to linkability to subjects, pseudonymity is the entire field between and including these extremes. Thus, pseudonymity comprises all degrees of linkability to a subject.

Explanation

Identifiability is usually a consequence of linking data to the same subject.

Data are considered de-identified when no identifiers (such as social security number, full name and address, birth date, etc.) are stored. Pseudo-identifiers (such as birth year (instead of birthdate), city (instead of full address), etc.) can however also often lead to an identification. As already summarized in the subsection Leading to identification of the Linkability property, the more information is linked together, the smaller the anonymity set will be. It is hence important to minimize the amount of information that can be linked to a certain subject in order to avoid identification. Identifiers and pseudo-identifiers should be handled with special care, but even information that at first sight does not seem to lead to identifiability (e.g. a rating of a song, a review of a movie) can potentially lead to identification, as was demonstrated when analyzing the search history of an anonymized AOL searcher [2].

Examples of identifiability are: identifying the reader of a web page, the sender of an email, the person to whom an entry in a database relates, etc.

Non-Repudiation

Definition

Non-repudiation

Having irrefutable evidence concerning the occurrence or non-occurrence of an event or action. [3]

Non-repudiation is related to the privacy property *plausible deniability*.

Plausible deniability ensures that "an instance of communication between computer systems leaves behind no unequivocal evidence of its having taken place. Features of communications protocols that were seen as defects from the standpoint of non-repudiation can be seen as benefits from the standpoint of this converse problem." [3]

Explanation

The attacker can prove a user knows, has done or has said something. He can gather evidence to counter the claims of the repudiating party.

Note that this threat is actually a security goal. The security property 'non-repudiation' and the privacy property 'plausible deniability' are mutually exclusive. This should however not cause any conflicts, as systems will either require strong non-repudiation properties to ensure accountability, while others will require plausible deniability.

For e-commerce applications, non-repudiation is an important security property. Imagine a situation where a buyer signs for a purchased item upon receipt, the vendor can later use the signed receipt as evidence that the user received the item.

For other applications, such as off-the-record conversations or online voting, participants may desire plausible deniability for privacy protection such that there will be no record to demonstrate the communication event, the participants and the content. In this scenario, non-repudiation is a privacy threat.

Typical non-repudiation examples exist in the context of anonymous online voting systems, and whistleblowing systems where plausible deniability is required.

Detectability

Definition

Detectability is related to undetectability and unobservability as defined by Pfitzmann and Hansen [1].

Detectability

An attacker can sufficiently distinguish whether an item of interest (IOI) exists or not.

Undetectability and unobservability refer to hiding the user's activities.

Undetectability of an item of interest (IOI) from an attacker's perspective means that the attacker cannot sufficiently distinguish whether it exists or not. If we consider messages as IOIs, this means that messages are not sufficiently discernible from random noise' [1].

Unobservability of an item of interest (IOI) means undetectability of the IOI against all subjects uninvolved in it and anonymity of the subject(s) involved in the IOI even against the other subject(s) involved in that IOI [1].

The definition suggests that unobservability is undetectability by uninvolved subjects combined with anonymity even if IOIs can be detected. Consequently, unobservability implies anonymity, and unobservability implies undetectability. In LINDDUN, we only focus on undetectability by taking into account detectability threats, as unobservability is in fact a combination of undetectability (cfr. Detectability threats) and anonymity (cfr. Identifiability threats).

Explanation

Detectability does not imply information disclosure. Detectability concerns IOIs of which the content is not known (to the attacker). Also, when compared to anonymity and unlinkability, undetectability differs. For anonymity and unlinkability, not the IOI, but only its relationship to the subject or other IOIs is protected. For undetectability, the IOIs are protected as such.

The most important consequence of detectability is *inference*. By detecting whether an IOI exists, one can deduce certain information, even without actually having access to that information. Knowing that, for example, a celebrity has a health record in a rehab facility, you can deduce the celebrity has an addiction, even without having access to the actual health record.

Examples of detectability include: knowing whether an entry in a database corresponds to a real person, being able to distinguish whether someone or no one is in a given location, knowing whether a message was sent, etc.

Disclosure of Information

In the latest version of LINDDUN, the information disclosure threats are less emphasized than the other LINDDUN threat categories. Information disclosure is actually a threat category borrowed from STRIDE, and it is advised to perform a full security analysis in addition to LINDDUN, rather than only considered information disclosure threats in LINDDUN.

Definition

Information disclosure is the counterpart of confidentiality, which refers to hiding the data content or controlled release of data content. Examples include transferring encrypted email, applying access control to a classified document or a database containing sensitive information.

NIST [4] describes confidentiality as "preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information."

Information disclosure can thus be defined as follows:

Disclosure of Information

Exposing information to someone not authorized to see it.

Explanation

Although confidentiality is a security property, it is also important for preserving privacy properties, such as anonymity and unlinkability. Therefore, confidentiality is also considered an important privacy objective.

We however advise to perform a full security analysis (e.g. STRIDE), rather than only taking information disclosure threats into account. Security is indispensable for privacy and deserves the required attention.

Unawareness

Definition

Awareness is a rather broad concept.

Endsley [5] defined situation awareness as "*the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future*." Awareness in computer-supported cooperative work has been defined as "*an understanding of the activities of others, which provides a context for your own activities*" [6]. Sohlenkamp [7] defined awareness as "*an understanding of the state of a system,* *including past activities, present status and future options*." In the LINDDUN methodology, awareness refers to an understanding of the consequences of sharing personal information in the past, present and future.

Unawareness

Not understanding the consequences of sharing personal information in the past, present, or future.

Explanation

With the emerging of Web 2.0 technologies, users tend to provide excessive information to service providers and lose control of their personal information. Therefore, the unawareness threats are related to users who should be aware of their personal data. Also, only the minimum necessary information should be sought and used by software systems to allow for the performance of the function to which it relates.

The more personal identifiable information a data subject discloses, the higher the risk is for privacy violation. To ensure awareness, a number of technical enforcement tools have been developed. For instance, the concept of personal information feedback tools [8] [9] has been promoted to help users gain privacy awareness and determine themselves which personal data to disclose.

The Platform for Privacy Preferences Project (P3P) [10] has been designed to allow websites (as data controllers) to declare their intended use of the information that they collected about the browsing users (as data subjects). P3P addresses the awareness property by making users aware of how personal data are processed by the data controller (i.e. the individual or group who determines the purposes and means of the processing of personal data).

> This property does not imply that it is the sole responsibility of the user to ensure his own privacy. On the contrary, we would encourage the service providers to facilitate tools to increase user awareness.

Currently, it is unfortunately still often common practice to hold the user responsible for his privacy; e.g. "informed" consents where the user has to inform himself, (hidden) opt-out consents which the user should figure out himself, or the general belief that the user should just know to not overshare information. LINDDUN aims at a more system-centric awareness as we want to have the service providers implement as much support as possible to improve the user's privacy awareness.

The system itself should aid the user in privacy decision making by providing more transparency or even nudging the user into a privacy-friendly decision. LINDDUN's privacy analysis cannot impact the entire society, and will therefore focus on the provisions the system itself can take to guide and educate the user concerning his data sharing and nudge the user into a more privacy-aware use of the system.

The goal is to prevent the sharing of excessive information at the source. Note that this is hard to prevent from a technical perspective as the user himself decides what information he will precisely share. Technology can only play a supportive role by, for example, providing feedback concerning the (additional) data the user wants to add, having privacy-friendly default settings, and providing user-friendly privacy tools. The main challenge will be to educate people about their (online) sharing behavior, however this is out of scope of the LINDDUN methodology.

To summarize, **the awareness property focuses on the user's consciousness regarding his own data. The user needs to be aware of the consequences of sharing information.**

Non-compliance

Definition

Non-compliance

Not following the (data protection) legislation, the advertised policies or the existing user consents

The compliance property requires the whole system as data controller to inform the data subject about the system's privacy policy, and allow the data subject to specify consents in compliance with legislation, before users accessing the system. According to the definitions from the EU Directive 95/46/EC [11]:

Controller shall mean "the natural or legal person, public authority, agency or any other body which alone or jointly with others determines the purposes and means of the processing of personal data."

The **data subject's consent** shall mean "any freely given specific and informed indication of his wishes by which the data subject signifies his agreement to personal data relating to him being processed."

Explanation

A **policy** specifies one or more rules with respect to data protection. These are general rules determined by the stakeholders of the system. A **consent** specifies one or more data protection rules as well, however, these rules are determined by the user and only relate to the data regarding this specific user. The non-compliance threat category essentially examines whether the system's policy and the user's consent (usually specified in textual form) are indeed implemented and enforced.

One example of consent compliance can be found in an e-health context. In some countries, healthcare professionals are only allowed to access medical information if the data subject has given informed consent (or in case of emergency).

This category is also related to **legislation**. There are a number of legal frameworks addressing the raised concerns of data protection, such as the Health issued the Insurance Portability and Accountability Act (HIPAA) [12] in the United States, the Data Protection Directive 95/46/EC [11] in Europe, and the OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data [13].

There are initiatives to protect data subjects and create openness[4]. It is clearly important to ensure that internal rules actually comply with those promised in policies and consents. Unfortunately, few technical solutions exist to guarantee compliance.

A possible non-technical solution is to use employee contracts to enforce penalties (e.g. get fired or pay fines) to ensure compliance. Another solution is to hire an auditor to check policies compliance. Eventually, necessary legal actions can be taken by data subjects in the case of non-compliance.

Note that (at least in this version of LINDDUN), **in order to obtain full legal compliance, the additional regulatory threats should be analyzed, preferably by a legal expert.**

[4] Tools like Privacy Bird have already emerged that inform users about how information they provide to websites could be used. These tools interpret the website's privacy policies and translate them to information that is easier to grasp (as privacy policies are hardly ever read by users because they tend to be very extensive and hard to comprehend).

LINDDUN step-by-step

This section provides a detailed overview of each of the six steps of the LINDDUN methodology.

The first three steps are *problem-oriented* steps that aid in identifying privacy threats in a software systems. These steps are considered the core of the LINDDUN methodology, as shown in Figure 2.

The final three steps are more *solution-oriented* and aid in translating the elicited threats into privacy mitigation strategies and solutions, as depicted in Figure 3.

1. Creating a DFD

First, a model of the system to analyze needs to be created. This model will be the basis of the analysis. The software system being analyzed is decomposed into relevant (either logical or structural) components, and for each of these parts the corresponding threats are analyzed. This process is repeated over an increasingly refined model until a level is reached where the residual threats are acceptable. As model representation, LINDDUN requires a data flow diagram or **DFD**[5].

This DFD model is based on the available system description, which can be at the level of both requirements and architecture.

> A DFD is a structured, graphical representation of the system using 4 major types of building blocks: external entities, data stores, data flows, and processes.

The DFD describes the way information flows throughout the system, using 4 types of building blocks, as shown in Figure 4:

- *Process (P)*: a unit of work that operates on the data
- *Data flow (DF):* a named flow of data through a system of processes
- *Data store (DS):* a logical repository of data (a passive container of information)
- *External entity (E):* a source or destination of data, such as system users or 3rd party services

Optionally, trust boundaries can be added to the diagram to indicate the places where parties with different privileges interact. A typical example is the boundary that separates the internal system from the external parties.

[5] DFD is also used by the SDL threat modeling process. Hence by deploying the same modeling technique, an interesting synergy can be created between the proposed framework and the STRIDE process.

The creation of the DFD is an important part in the analysis. If the DFD would be incorrect, the analysis results would be inconsistent with the actual system. Since privacy focuses on the protection of a user's personal information, it is important to consider where the information will be stored or passed by, as these are the crucial elements for building in privacy.

The granularity of the DFD will therefore also impact the level of detail of the elicited threats.

Running example

In this tutorial we will use, for the sake of simplicity, a very high-level representation of a social network system, to illustrate the different LINDDUN steps.

This Social Network 2.0 application is an abstract representation of a social network, where online users share personal information such as relationship status, pictures, and comments with their friends. In our running example, Alice is a registered user of a social network. Each time Alice updates her friends list, she first connects to the social network's web portal. Accordingly, the portal communicates with the social network's service, and eventually, the friendship information of Alice and all other users of that social network is stored in a database.

The DFD for the Social Network 2.0 application is shown in Figure 4. In the DFD, the user is represented as an entity to interact with the system. The Social Network 2.0 application contains two processes (the portal and the service) and one data store containing all the personal information of the users. Note that the model in this tutorial has been kept high-level for the sake of simplicity in this example. In practice, the DFD will be more detailed (references to more extensive examples can be found in Section Examples of applying LINDDUN). Evidently, the more fine-grained the DFD, the more detailed the elicited threats will be.

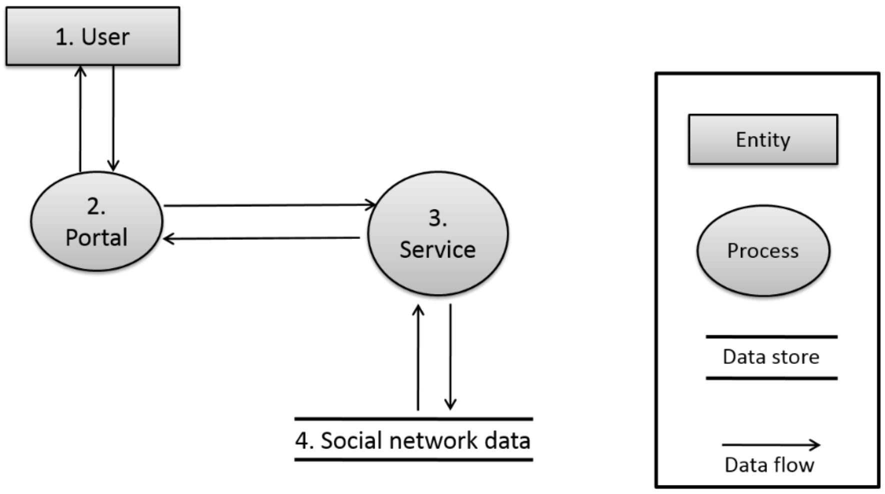


FIGURE 4: EXAMPLE DFD OF SOCIAL NETWORK APPLICATION

From requirements to DFD

You can start by creating a *level 0* DFD, which corresponds to a context diagram. Create **one main process** to represent the system. Next, connect this process to the actors of your system (the users of the system, but also components and services from 3th parties). These are the **external entities** of the DFD.

In the next step, you can decompose the process into a more detailed representation of your system. Think about the **data store(s)** that collect the information in the system. Also add more detail to the internal processes. You will likely have processes that are designated to communicate to external entities (in the form of facades, portals, websites, etc.). Also, there will be processes responsible for the databases, like repository processes (as data stores are passive containers of information which require a process to actually manage the information). And also the internal operation of the system will require multiple processes in order to represent its complexity.

Evidently, you will need to add **data flows** to connect all of the above DFD elements to one system. It is possible to have multiple data flows between two elements. Multiple flows can indicate that several types of information are shared between these elements, which can result in multiple threats.

From architecture to DFD

It will be even easier to create a DFD based on an existing architectural description.

> Experience has shown that it is rather straight-forward to translate a component-connector view to a DFD.

First the **external entities** are identified by determining the connectors that link to external "contracts". Alternatively, the external entities can also be deduced from the actors of the use cases, or from the system's context diagram.

These external entities will, in general, be connected to the internal system via some kind of frontend, portal, facade, etc. These will be the first *processes* in the DFD.

Similarly, the databases can be easily extracted from the component-connector view and instantiated as **data stores** in the DFD. Depending on the granularity of the analysis (and thus of the DFD), the portals can be directly connected to the data stores by means of data flows, or additional internal processes can be added which will manage the internal business flows and computations. ***Note that a data store is a passive container of information and will always require***

a process to retrieve or insert data. A data store can thus never be directly connected to another data store or external entity. Depending on the granularity of the DFD, a data store-specific process can be added that will act as database manager (or repository), or this functionality can be assumed to be integrated within the internal process that communicates directly with the data store.

In general, **processes** correspond to the components on the component-connector view. Depending on the granularity required for the analysis, it can be decided to combine several detailed components into one DFD process. Alternatively, one component can be decomposed into multiple, more fine-grained DFD processes. This might be useful when one component processes and outputs several types of data to multiple components.

Evidently, the **data flows** represent the connections between the different components. It can be useful to have multiple data flows between two DFD elements to clearly indicate the different types of data that are transmitted, as different types of data will likely correspond to different threats (or at least to different threat priorities).

Naming DFD elements

It is advised to use an ID (i.e. a number or other abbreviation) for each DFD element, to enable easy reference.

A suggestion is to describe each entity as E1, E2, etc.; a data flow as DF1, DF2, etc.; a data store as DS1, DS2, etc.; and a process as P1, P2, etc.

In the following steps of the methodology, these identifiers can then be used for quick (and short) reference.

Side notes

Our experience has shown that, for a privacy threat analysis, we are mainly interested in the information that is collected by the system (i.e. information external entities provide to the system), how it is stored (in the data stores), and, evidently, also in what information is shared with external entities. This means that **the analysis will focus less on the internal processes. It is hence not required to decompose the internal operations of the system in too much detail.**

On the other hand, the key focus of the privacy analysis will be on the information that traverses the system. It is thus important to **describe as precisely as possible what data are transmitted over a data flow**. For example, "user information" can correspond to several types of data. Be more specific and label the flow with the actual information that is being communicated, for example, "email address, username, and password".

2. Mapping the DFD to LINDDUN threat categories

As shown in Table 1, the methodology considers seven types of threats. *LINDDUN* is the mnemonic acronym that we use. Our threat categorization, based on the categories discussed in Section Privacy background, considers privacy threats from the attacker's perspective.

The relationship between the LINDDUN privacy threat categories and the DFD element types is depicted in Table 1. Each DFD element type is potentially subject to specific privacy threats. In summary, LINDDUN has identified the following 7 high-level categories:

- **Linkability (L)** occurs when one can sufficiently distinguish whether 2 items of interest (IOI, such as requests from a user) are related
- **Identifiability (I)** occurs when it is possible to pinpoint the identity of a subject (e.g., a user)
- **Non-repudiation (Nr)** occurs when it is possible to gather evidence so that a party cannot deny having performed an action
- **Detectability (D)** occurs when one can sufficiently distinguish whether an IOI exists, e.g., in a system
- **Disclosure of information (Di)** is the exposure of information to individuals who are not supposed to have access to it
- **Unawareness (U)** occurs when the user is unaware of the information he is supplying to the system and the consequences of his/her act of sharing
- **Non-compliance (Nc)** occurs when the system is not compliant with the (data protection) legislation, its advertised policies and the existing user consents

More information about these threat categories can be found in Section Privacy background.

In Table 1, a 'X' means that the DFD element type in the corresponding row is susceptible to the privacy threat category of the selected column. For example, only an external entity (e.g. a user) can be aware of the consequences of introducing personal information into the system. Therefore, the concrete threats for that category should be examined in relation to the elements of that type in a DFD built during the previous step. In essence, each DFD element is subject to certain privacy threats, and the nature of the potential privacy threat is determined by the DFD element type. For example, a data flow is subject to a number of privacy threats such as identifiability, linkability, detectability, non-repudiation, and information disclosure.

As shown in Table 1, LINDDUN provides a template that indicates which threat categories are applicable to each DFD element type.

Threat categories	E	DF	DS	P
Linkability	X	X	X	X
Identifiability	X	X	X	X
Non-repudiation		X	X	X
Detectability		X	X	X
Disclosure of information		X	X	X
Unawareness	X			
Non-compliance		X	X	X

TABLE 1: MAPPING TEMPLATE

MAPPING LINDDUN COMPONENTS (PRIVACY THREATS) TO DFD ELEMENT TYPES (E-ENTITY, DF-DATA FLOW, DS- DATA STORE, P-PROCESS)

Use the mapping template from Table 1 to create a personalized mapping table that corresponds to the system you are analyzing. The resulting mapping table should contain one row for each element in the system's DFD. This table should then be used as checklist throughout the analysis, as each 'X' in the table represents a potential threat posed to a specific DFD element of the system. Therefore, **each 'X' should either be documented as (at least one) threat, or an assumption should be explicitly written down to explain why the threat is not susceptible to the given element.**

Running example

Considering the Social Network 2.0 application, the list of generic privacy threats to the modeled system is depicted in Step 2 in Figure 2. This is obtained by gathering the elements from the DFD (Step 1) and then determining the susceptible threats with Table 1 (also shown in the right of Step 2 in Figure 2).

The intersections marked in grey in the table of step 2 in Figure 2 are potential threats that have been considered as irrelevant to the specific usage scenario in this example. Each intersection that is indicated with an 'X' shows that there will be a privacy threat at the corresponding DFD element. These items are the threats which we will actually consider.

For ease of example, we assume that DFD elements within the system boundaries are trustworthy. We trust the processes as well as all data flows between trusted processes and between processes and data store. Therefore, we will not discuss linkability, identifiability, and information disclosure threats on these elements. We however do not trust the user and its communication with the portal and we also want to protect the data store containing all the user's information.

Note that these kind of assumptions should be made explicit during the analysis (as explained below). If it is later on decided that certain assumptions do not hold after all, it should be easy to trace the impact of that decision throughout the analysis process.

In the following step, each of the 'X' elements will be examined more closely to elicit the detailed threats. In the example of Figure 2, linkability threats for the data store are being scrutinized.

Side notes

'X's that are less likely to cause threats

Our experience has shown that certain 'X's in the table are less likely to result in actual high-priority privacy threats. We provide an overview below. Note that when you decide to discard 'X's, you should still document this decision as an explicit assumption (including the rationale behind it), as explained in Section General assumptions.

All **'X's related to processes** can (usually) be determined as not applicable to the system. Process threats related to privacy always require information disclosure of the process. When such a threat occurs, the actual disclosure of information will be more problematic than the privacy consequences that are related to this disclosure. Nevertheless, in systems that handle highly sensitive data, these threats should still be considered.

Non-repudiation is also a threat category that is less likely to occur in a general system. It can cause privacy issues for systems that involve e-voting, whistleblowing, etc. In a regular system however, non-repudiation will more likely be a (security) requirement than a (privacy) threat.

Linkability and identifiability of entities can become a privacy issue when anonymous (or pseudonymous) use of the system is required (e.g. anonymous credentials for authentication, anonymous communication to hide relationships, etc.). In general systems (especially those that require an identifiable login of users), this is however not a threat.

Combining 'X's (reduction)

In theory, each 'X' should be examined (and documented) individually. In practice however, it is advised to apply the technique of **"reduction"**. This implies that several 'X's can be combined when they apply to the same threat. This is possible for 'X's that involve DFD elements of the same DFD type (e.g. all data flows) and when the threat that corresponds to the 'X's is the same because it involves the same type of data (e.g. usernames and passwords, or non-sensitive or anonymous data, etc.) and it results in the same consequences (with the same level of priority). By combining these 'X's, the resulting threat description document will become easier to handle as documenting each individual 'X' will quickly result in an explosion of threats.

An exception to the rule that states that only 'X's of the same DFD element type should be combined, are non-compliance 'X's. Non-compliance threats are rather generic and apply to the entire system. Therefore, all **non-compliance 'X's** can be combined and handled together (again, this decision should still be documented as an assumption).

3. Eliciting privacy threat using LINDDUN

The third step can be considered as the core execution step of the LINDDUN threat modeling methodology.

As it is a quite labor-intensive task, we have divided it into three sub-activities: refining threats via threat tree patterns, documenting assumptions, and documenting threats using a threat template.

Refine threats via threat tree patterns

For each 'X' in the mapping table (see step 2), LINDDUN provides a list of concrete threats that need to be considered. To ease the navigation, the threats are organized into trees.

The privacy threat trees are inspired by the Secure Development Lifecycle (SDL) [14] and based on the state-of-art privacy developments. These threat trees reflect common attack patterns and help application designers think about privacy conditions in the system. As the threat trees are regularly updated, we refer the reader to the latest version of the trees, which is available on the LINDDUN website. For each marked intersection in the mapping template of step 1 (see Table 1), a threat tree exists showing the detailed preconditions for this specific threat category to materialize. The preconditions are hence vulnerabilities that can be exploited for a privacy attack scenario.

Document Assumptions

When eliciting threats using the LINDDUN threat trees, often certain leaf nodes or even entire branches will be considered as not relevant to the system-to-analyze and will thus not be considered during the privacy analysis. It is important to also document these decisions.

Assumptions are explicit or implicit choices to trust an element of the system (e.g., human, piece of software) to behave as expected. Clearly, the analyst needs to make assumptions that are used to reason about whether a threat (or a whole category) is relevant in the system under analysis. These assumed properties or assertions act as domain restrictions. They can for example state that a certain branch in a threat tree does not apply to the system that is being analyzed, hence limiting the scope of the LINDDUN analysis.

The analyst is incentivized to document these assumptions and argumentations as free-form text. Preferably a link to the corresponding misuse case(s) is added for traceability purposes, indicating why a certain alternative path is considered irrelevant or why a certain threat is highly likely to occur. **When one of the assumptions is altered in the process, it is important to easily track the required changes in the privacy analysis results.**

Running example: Social Network 2.0.

In the running example, we have made the assumption that re-identification is not possible in the data store (as shown in Step 3 of Figure 2), thus eliminating one branch of the threat tree[6].

General assumptions

Note that the documentation of assumptions already starts during step 2 of the LINDDUN methodology. When adding DFD elements, the mapping table, and thus the number of threats to examine, grows exponentially. This number can be limited by making more general assumptions (i.e. spanning an entire threat category or DFD element).

Related to the running example, as shown in the mapping table of step 2 in Figure 2, certain threat categories have been considered irrelevant (marked in grey). This limits the number of threats that need to be considered throughout the analysis. Internal DFD elements are, for example, considered trustworthy. Also non-repudiation and detectability threats are considered irrelevant for social networks (after carefully examining the corresponding threat trees). Finally, non-compliance threats should not be applied to a specific DFD element, but are applicable to the entire system (indicated by X* in the table).

Again, note that these assumptions are made to simplify the example. In practice, each assumption should be carefully thought through and should be substantiated by adequate reasoning.

These assumptions have a large impact on the privacy results as they eliminate entire threat trees from the LINDDUN analysis. It is hence imperative to explicitly document these assumptions for traceability purposes.

Document Threats using Threat Template

The result of the elicitation process should be a collection of threat scenarios that need to be documented. To this aim, LINDDUN proposes the use of misuse cases [15] [16] [17] [18].

In particular, a misuse case can be considered as a use case from the misactor's point of view. A misactor is someone who intentionally or unintentionally initiates the misuse case. We chose misuse cases because they represent a well-established technique to elicit threats.

[6] Note that the re-identification assumption has only been made for the sake of example. In practice, assumptions should only be made if there is sufficient reason to eliminate a certain threat tree branch. This reasoning should be documented as well.

Threat description template

The proposed misuse case structure is described below (*optional fields are indicated with* *):

Summary: provides a brief description of the threat.

Assets, stakeholders and threats*: describes the assets being threatened, their importance to the different stakeholders, and what the potential damage is if the misuse case succeeds.

Primary misactor: describes the type of misactor performing the misuse-case. Possible types are insiders, people with a certain technical skill, and so on. Also, some misuse cases could occur accidentally whereas other are most likely to be performed intentionally.

Basic Flow: discusses the normal flow of actions, resulting in a successful attack for the misactor.

Alternative Flows*: describes the other ways the misuse can occur.

Trigger*: describes how and when the misuse case is initiated.

Preconditions*: precondition that the system must meet for the attack to be feasible.

Leaf node(s)*: refers to the leaf node(s) of the threat tree(s) the threat corresponds to.

Root node(s)*: refers to the root node(s) of the threat tree(s) that were examined for the threat.

DFD element(s)*: lists all DFD elements to which this threat is applicable.

Remarks(*): Although optional, related assumptions should be mentioned here.

Assets, stakeholders and threats describe the impact the threat can have. It is an optional field, however it can be useful to determine the risk (step 4 of LINDDUN) as this is based on the impact and likelihood of the threat.

The *misactor* describes the profile of the attacker.

The *basic flow* discusses the normal flow of actions performed by the misactor, resulting in a successful attack. The *alternative flows* is an optional field which can be used to describe additional ways the misuse can occur.

Also *trigger* and *preconditions* are optional fields as they do not represent the core threat. They can however help reason about the threat.

Leaf nodes, root nodes, and *DFD elements* are LINDDUN-specific fields which are optional but highly recommended. The leaf and root nodes make it easy to trace the threat to its corresponding threat tree(s) and vice versa. When combining these references with the assumptions, it becomes rather straightforward to determine whether all threat trees and branches have been fully explored. The DFD elements list can be used to group threats. Often one threat is applicable to similar DFD elements (e.g. internal flows, external flows, etc.). To avoid an explosion of threats and obtain a clear, organized overview, these similar threat can be grouped into one (see Section Combining 'X's of Step 2 for more information about reduction).

> Apply "reduction" by combining similar threats and documenting them together in order to tackle the explosion of applicable threats and keep the threat documentation more easy to manage [7].

Finally the *remarks* field is also optional, although it will usually be filled in with references to the assumptions that correspond to the documented threat.

Documenting threats should not necessarily be limited to this step. On the contrary, as privacy threats can already emerge earlier on in the software development process, these threats should be documented as soon as they surface.

4. Prioritizing threats

The LINDDUN methodology can result in a (large) number of documented threats[8]. Before moving forward and looking for suitable mitigation, the identified threats must be prioritized. Due to time or budget limitations, often only the most important ones will be considered for inclusion in the requirements specification and, consequently, in the design of the solution.

Risk assessment techniques provide support for this stage. **In general, risk is calculated as a function of the likelihood of the attack scenario and its impact.** The risk value is used to sort the MUCs: the higher the risk, the more important the MUC is.

> Risk = likelihood x impact

[7] See Section Combining 'X's for more information on reduction.

[8] We therefore advise to apply the concept of "reduction" during the elicitation phase, as explained in Section Combining 'X's. This implies that similar threats are combined and documented together to limit the explosion of documented threats.

The LINDDUN framework is independent from the risk assessment technique that is used. The analyst is free to pick the technique of choice, for instance the OWASP's Risk Rating Methodology [19], Microsoft's DREAD [20], NIST's Special Publication 800-30 [21], or SEI's OCTAVE [22]. These techniques leverage the information contained in the misuse cases, such as the involved assets (for the impact), and the attacker profile as well as the basic/alternative flows (for the likelihood).

5. From threats to mitigation strategies

The goal of threat modeling is to identify flaws in the system in order to resolve them. Addressing these flaws will require a number of fundamental design decisions. These decisions are often referred to as strategies (or tactics [23]). A strategy describes the means to achieve a certain objective, or, in case of LINDDUN, a means to resolve privacy threats. We will therefore refer to them as "*mitigation strategies*". They capture a high-level view of common techniques used in practice to prevent privacy threats. As these strategies can be used to classify privacy solutions, they are a suitable step in the conversion of privacy threats to appropriate privacy enhancing solutions.

Mitigation taxonomy

In essence, we leverage on a taxonomy of solution strategies that provides a structured classification of common mitigation decisions.

Obtaining privacy means "*controlling the consequences of exposing the (possibly indirect) association of individuals to some information/transaction in a given context.*" Accordingly, in order to obtain privacy, it is important to focus on two major strategies.

First, associations between users and their transactions and personal information need to be controlled in order to ensure that the user shares as little information as necessary with the system. This is the *proactive* approach. Second, the damage must be limited by controlling the associations after disclosure. To achieve this, the exposure of these associations needs to be restricted to the minimum. This is the *reactive* approach. *Concealing the association* can be divided into two sub-strategies: (1) protect the identity of the user during authentication and (2) protect the data that will be communicated to (or throughout) the system. *Guarding the association* after the data has been shared can be divided into two sub-strategies: (1) guard exposure and (2) maximize accuracy. The mitigation strategies taxonomy is available on the download page of the LINDDUN website.

Selection of mitigation strategies

In addition, we provide a means to select the appropriate strategies by mapping them to the LINDDUN threat trees (or branches) they intend to mitigate.
First, a distinction can be made based on the type of DFD element that corresponds to a specific threat. Threats related to entities and data flows correspond to the concealment of data. In particular, strategies that aim at *protecting the identity* will mitigate entity-related threats (linkability and identifiability of entity), while mitigation strategies for *data protection* before and during communication will aim at resolving data flow threats.

Second, mitigation strategies that *guard the exposure* of associations correspond to threats related to data that have already been collected and stored. These clearly map to data store related threats, which can be divided in two categories: threats that require confidentiality and threats that require minimization. The strategy that *maximizes the accuracy* by empowering the subject to update and delete collected data (either directly, or indirectly by requesting deletion) is linked to the subtree of non-repudiation that requires deniability by editing the database.

Unawareness threats can be found in both branches of the taxonomy of mitigation strategies as a user should be aware of the consequences before he shares information, as well as of what data are in fact collected about him to verify the accuracy.

Please download the mitigation strategies mapping table from the download page on the LINDDUN website.

Running example: Social Network 2.0.
Linkability of the data store threats correspond, according the mitigation mapping table in Figure 3 to "guarding the exposure" mitigation strategies. As the threat tree leaf node describes "too much data", the most obvious mitigation is "minimization". In this example, generalization was selected as most suitable minimization strategy.

Similarly, information disclosure of the data store (the other leaf node of the linkability threat), can be mitigated with a confidentiality strategy. In this example, access control was selected as most suitable mitigation strategy.

6. From strategies to …

This sixth step of the LINDDUN methodology is mainly used to translate the selected mitigation strategies (from step 5) to appropriate privacy enhancing solutions. Nevertheless, this step can also be used to translate the selected mitigation strategies into privacy requirements that can be incorporated in a new iteration of the development lifecycle.

… Privacy enhancing solutions

A threat can be mitigated by several types of solutions, each with their own benefits and drawbacks. The selection of a fitting solution is thus not straightforward. We therefore leverage on mitigation strategies to enable an easier and more focused selection of the appropriate privacy enhancing technologies. By first determining a proper strategy to mitigate the threat, we narrow the scope of relevant solutions.

LINDDUN provides a table of solutions that indicates to which privacy strategies each solution contributes. The table represents the hierarchical taxonomy of mitigation strategies and links each leaf node of the taxonomy tree to the relevant solutions. Note that, although some solutions correspond to multiple strategies, we only assigned them to their key strategy (e.g. onion routing does not only remove contextual data, but also hides transactional data). In addition, some strategies refer to solutions of related strategies to avoid duplication (e.g. generalization of transactional data refers to encryption solutions in the guard exposure branch).

Once the mitigation strategy has been decided, a limited set with designated privacy enhancing techniques can be extracted from the solution table. The more detailed the proposed mitigation strategy is, the more focused the selection of appropriate solutions will be. **The table thus facilitates an easy selection of proper privacy enhancing technologies to mitigate a specific threat.**

The solutions table can be found on the download page of the LINDDUN website.

Running example: Social Network 2.0.

As shown in Figure 3, the generalization strategy was selected in step 5. This corresponds to two possible PETs in the solution table. In this example, k-anonymity was chosen as most suitable solution.

… Privacy requirements

The mitigation strategies can also be used to elicit privacy requirements that can be incorporated into another iteration of the development lifecycle. This is a rather straight-forward process. It is based on the threat trees that led to the threat and their corresponding mitigation strategies. By negating the threat tree nodes that correspond to the threat and making them more explicit by including the selected mitigation strategy, a privacy requirement is defined.

Running example: Social Network 2.0.

In the solution space example of Figure 3, the "linkability of entity" threat tree node for which the mitigation strategy of "minimization by means of generalization" was selected is translated into *"ensure unlinkability of data entries within the social network database by means of generalization"*.

Links

This section includes additional references to LINDDUN work.

LINDDUN

For more information about LINDDUN, we refer the reader to the official LINDDUN website.

You can also consult the original LINDDUN research paper [24], the paper describing the empirical evaluation of LINDDUN [25], and the PhD thesis [26] that describes the LINDDUN methodology, the empirical evaluation, and the proposed changes of LINDDUN 2.0.

Examples of applying LINDDUN

A number of example cases are available on the LINDDUN website.

Social network 2.0 – running example

The full running example is available in the LINDDUN paper [24], and on the LINDDUN website. Note that in this example has been worked out using the original LINDDUN and thus not takes into account the updated LINDDUN trees or the updated fifth step (mitigation strategies) of the methodology.

Patient communities example

A more extensive execution of LINDDUN can be found online, where a step-by-step application of the first 4 LINDDUN steps to a patient community system is provided. This document also illustrates the easy transition from the client-server diagram to the DFD required by LINDDUN.

Smart grid example

Another example involves a smart grid system that is being evaluated for privacy threats using LINDDUN. The resulting document can be found online.

LINDDUN in the Related Work

LINDDUN also has been referenced and applied by a number of independent researchers. We refer to the LINDDUN website for the latest links and testimonials.

Bibliography

[1] A. Pfitzmann and M. Hansen, "A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management, v0.34," 2010.

[2] M. Barbaro and T. Zeller, "A Face Is Exposed for AOL Searcher No. 4417749 (The New York Times)," August 2006. [Online]. Available: http://www.nytimes.com/2006/08/09/technology/09aol.html?pagewanted=all&_r=1&.

[3] M. Roe, *Cryptography and Evidence,* PhD thesis, Clare College, University of Cambridge, 1997.

[4] E. McCalllister, T. Grance and K. Kent, "Guide to protecting the confidentiality of personally identifiable information (PII) (draft)," National Institute of Standards and Technology (NIST), 2009.

[5] M. R. Endsley, "Toward a theory of situation awareness in dynamic systems," *Human Factors: The Journal of the Human Factor and Ergonomics Society,* vol. 37, no. 1, pp. 32-64, 1995.

[6] P. Dourish and V. Bellotti, "Awareness and coordination in shared workspaces," in *Conference on Computer-supported Cooperative Work (CSCQ '92)*, 1992.

[7] M. Sohlenkamp, *Supporting Group Awareness in Multi-User Environment Through Perceptualization,* PhD thesis, Universitat Padernorn, Germany, 1998.

[8] S. Lederer, J. I. Hong, A. K. Dey and J. A. Landay, "Personal privacy through understanding and action: Five pitfalls for designers," *Personal and Ubiquitous Computing,* vol. 8, pp. 440-454, 2004.

[9] S. Patil and A. Kobsa, "Privacy Considerations in Awareness Systems: Designing with Privacy in Mind," in *Human-Computer Interaction Series*, Springer, 2009, pp. 187-206.

[10] W3C, *Platform for privacy preferences (P3P).*

[11] European Parliament, *Directive 95/45/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data,* 1995.

[12] US Federal Register - Rules and Regulations, *HIPAA administrative simplification: Enforcement; final rule.,* 2006.

[13] OECD, *Guidelines on the protection of privacy and transborder flows of personal data, organization for economic cooperation and development,* 1980.

[14] M. Howard and S. Lipner, The Security Development Lifecycle, Microsoft Press, 2006.

[15] I. Alexander, "Misuse cases: Use cases with hostile intent," *IEEE Software,* vol. 20, no. 1, pp. 58-66, 2003.

[16] J. McDermott and C. Fox, "Using abuse case models for security requirements analysis.," in *15th Annual Computer Security Application Conference (ACSAC '99)*, 1999.

[17] G. Sindre and A. L. Opdahl, "Eliciting security requirements with misuse cases," *Requirements Engineering,* vol. 10, no. 1, pp. 34-44, 2005.

[18] G. Sindre and A. L. Opdahl, "Templates for misuse case description," 2001.

[19] OWASP, "Risk rating methodology," [Online]. Available: http://owasp.org/index.php/OWASP_Risk_Rating_Methodology.

[20] Microsoft, "Improving web application security: Threats and countermeasures," [Online]. Available: https://msdn.microsoft.com/en-us/library/aa302419.aspx#c03618429_011.

[21] NIST, "Risk management guide for information technology systems, special publication 800-30," [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf.

[22] CMU Software Engineering Institute, "OCTAVE," [Online]. Available: http://www.cert.org/octave.

[23] L. Bass, P. Clements and R. Kazman, Software Architecture in Practice, Addison-Wesley Professional, 2005.

[24] M. Deng, K. Wuyts, R. Scandariato, B. Preneel and W. Joosen, "A privacy threat analysis framework: supporting the eliciation and fulfillment of privacy requirements," *Requirements Engineering,* vol. 16, no. 1, pp. 3-32, 2011.

[25] K. Wuyts, R. Scandariato and W. Joosen, "Empirical evaluation of a privacy-focussed threat modeling methodology," *The Journal of Systems and Software,* vol. 96, pp. 122-138, 2014.

[26] K. Wuyts, *Privacy Threats in Software Architectures,* PhD thesis, Department of Computer Science, KU Leuven, 2015.