Software and Systems Modeling (2022) 21:2311–2328
https://doi.org/10.1007/s10270-021-00941-7

THEME SECTION PAPER

A descriptive study of assumptions in STRIDE security threat modeling

Dimitri Van Landuyt[1] · **Wouter Joosen**[1]

Received: 16 October 2020 / Revised: 2 August 2021 / Accepted: 11 October 2021 / Published online: 17 November 2021

Abstract
Security threat modeling involves the systematic elicitation of plausible threat scenarios, and leads to the identification and articulation of the security requirements in the early stages of software development. Although they are an important source of architectural knowledge, assumptions made in this context are in practice left implicit or at best, documented informally in an unstructured textual format. As guidelines and best practices are lacking, the nature, purpose and impact of assumptions made in this context is generally not well understood. We present a descriptive study of in total 640 textual assumptions made in 96 STRIDE threat models of the same system. The study mainly focuses on the diversity in how assumptions are used in practice, in terms of (i) the role or function of these assumptions in the threat modeling process, (ii) the degree of coupling between the assumptions and the system under analysis, and (iii) the extent to which these assumptions are exclusively specific to security. We observe large differences on all three investigated aspects: practitioners use the mechanism of assumption-making for diverse purposes, but predominantly to exclude certain threats from further analysis, i.e. to scope the analysis effort by steering it away from threat scenarios that are considered less relevant up front. Based on our findings, we argue against the exclusive use of Data Flow Diagrams as the main basis for threat analysis, and in favor of integrating more expressive attacker and trust models which can co-evolve with the threat model and the system.

Keywords Threat modeling · Security architecture · Secure development life-cycle (SDLC) · STRIDE · Security assumptions · Architecture knowledge management

1 Introduction

Threat modeling is an analysis activity that takes part in the early stages of the development life cycle [13] and involves assessing the applicability and relevance (i.e. plausibility) of specific threat scenarios being enacted in the specific system under analysis or design. A systematic threat modeling methodology involves (i) establishment of an architectural model of the system under analysis, (ii) systematic and exhaustive elicitation of plausible threats, and (iii) prioritization and further documentation of the identified threats as security requirements. Many threat modeling approaches append these steps with additional support for mitigating threats. Threat modeling is an integral part of secure development life-cycles such as Microsoft's SDL [14,30], CLASP [45], OWASP [22] and OpenSAMM [6]. STRIDE [30] is one of the elicitative threat modeling approaches most widely adopted by practitioners and researchers [48].

The threat elicitation stage is systematic and exhaustive in the sense that it involves brute-force enumeration of threat scenarios that may occur in the system under design. As a consequence, this process commonly runs into the drawbacks of *threat explosion* [39,47]: due to combinatorial explosions, prohibitively large amounts of threats have to be considered, prioritized and documented. This effect hampers the overall cost-effectiveness and practical feasibility of threat modeling. In practice, many tactics are used to address this issue, ranging from raising the abstraction level by omitting lower-level details, over explicitly modeling trust boundaries in which specific threats are not to be considered, to adopting domain-specific approaches [47] in which common threats have already been instantiated for specific application domains.

One practice particularly relevant in this regard involves the formulation of *assumptions* that postulate system prop-

Communicated by J. Araujo, A. Moreira, G. Mussbacher, and P. Sánchez.

✉ Dimitri Van Landuyt
dimitri.vanlanduyt@cs.kuleuven.be

[1] imec-DistriNet, Department of Computer Science, KU Leuven, Heverlee, Belgium

erties of relevance, the implications of which are relied upon during threat documentation, prioritization and mitigation. Although threat modeling approaches advocate explicitly documenting such assumptions, very little practical guidance is provided to threat modelers and it is unclear (i) what is an appropriate way to document these assumptions, (ii) what is their intended role or function in the threat modeling process, (iii) what is their impact on the overall quality of the threat analysis and (iv) and how to properly manage them beyond the scope of a threat modeling exercise.

Regardless, assumptions made during threat modeling represent architectural knowledge on the system that may impede systematic reuse [10], hamper system evolution [17], and hinder the compatibility of threat modeling with continuous and agile development practices [20]. Particularly in the context of cyber security, proper management of assumptions is essential as in practice, security vulnerabilities are often based on invalid assumptions or deliberate breakage of such assumptions [46]. To illustrate the importance of assumption management in a cyber security context, SecureDrop is an open-source whistleblower submission system that media organizations use to securely accept documents from anonymous sources. The public design documentation of the SecureDrop package [27] includes an extensive analysis of the threats that have been anticipated and addressed during development and more crucially, lists the key assumptions under which the stringent security and privacy requirements of this software package will be met. The documentation of these assumptions as such is a key part of the architectural documentation of the SecureDrop framework, to the extent that is deliberately shared with a wide audience for public review and scrutiny.

To obtain additional insights in how threat modeling assumptions are used in practice, we conduct a descriptive study of in total 96 STRIDE threat models, applied to a realistic industrial application, a Software-as-a-Service (SaaS) offering that generates and delivers PDF documents such as invoices and payslips in a batch-oriented manner. The bulk of the investigated threat models has been created by students in a controlled context: a project-driven master course on software engineering, at a stage in which the participants had already obtained in-depth familiarity of the application case at hand, and with STRIDE and its supporting materials. This main data set of 91 student threat models is extended with a 5 additional threat models created by security experts for confirmatory purposes.

We extensively investigate the in total 640 assumptions in terms of (i) their role in the threat modeling process, (ii) the degree of system coupling and the nature of system information involved, and (iii) the degree to which these assumptions are exclusively relevant in a security context. We specifically look at the nature of the information enclosed in the assumptions, with emphasis on aspects such as trust and attacker capabilities, assumed countermeasures and other architectural elements of the system related to deployment, implementation, etc.

We adopt a qualitative approach in which the assumptions are systematically classified by means of categorical variables and metrics and text-based keyword analysis, which is mainly a manual effort conducted through expert analysis. The generated findings allow us to formulate recommendations and define a roadmap towards (i) improved treatment of architectural assumptions in the specific context of security threat modeling, and (ii) improved management of these assumptions beyond the specific threat modeling or threat analysis context, i.e. in security architecture and design, and even in the broader context of the overall secure development life-cycle (e.g. validating assumptions in a testing context).

The relevance and potential impact of implicit and poorly-managed assumptions in secure software development is broadly recognized [4,23,24,29].

Relation to prior work. The qualitative investigation involves categorical variables/metrics and text-based lexical analysis and uses an approach that is adopted from a prior study [44] about assumptions made in the context of LINDDUN privacy threat modeling. In that regard, this study is a replica study in terms of the study approach, with the following important distinctions:

- Security assumptions, created as a by-product of STRIDE are not necessarily equivalent to assumptions generated during privacy threat modeling using LINDDUN. For example, although they have been constructed and applies in an identical fashion, the security and privacy lexicons created to assess the degree of specificity are independent of each other and entirely distinct in both studies.
- We perform analysis of assumptions generated in the context of two very different applications: in this paper, we focus on assumptions that apply to a document generation SaaS application, whereas the prior study has focused on privacy assumptions that apply to the architecture of an IoT-based smart metering system [44].

Apart from the core research objectives of attaining a clearer view on security assumptions as outlined above, it is a potential side-effect to gain further insight on the commonalities and fundamental differences between security and privacy threat modeling, obtained via cross-comparison of results. From that perspective, our study builds upon and contributes to a reusable and versatile classification framework of assumptions created as a by-product of secure software development and lays the groundwork for further investigation and research on the articulation of such assumptions, for example in other requirements elicitation or software design

approaches [50] and the broader context of cyber security, threat intelligence and risk management.

Paper outline The remainder of this paper is structured as follows: Sect. 2 discusses the context and motivates the paper. Section 3 outlines the research questions and the study design. Section 4 presents the results which are subsequently interpreted and discussed in Sect. 5. Section 6 concludes the paper.

2 Context and motivation

This section discusses the background on the role of assumptions in software engineering and on threat modeling. After that, the more specific role of assumption-making in the context of threat modeling is discussed and from this context, the overall motivation for this descriptive study is provided.

2.1 Assumptions in software engineering

An architectural assumption [10,25] is a specific type of development assumption [20,35], made during the architecture design phase. Lago et al. [18] highlight the difficulty of distinguishing between constraints, requirements and assumptions, yet as they exert an impact on the architectural decision-making process, they are integral part of the architectural knowledge [1] of a system. A main difference is that constraints and requirements are in practice documented explicitly whereas assumptions are left implicit or documented in an ad-hoc fashion [8,19].

Industry survey studies [49–51] show that the inherently vague concept of development assumptions is not well known in practice, and that interpretations differ between practitioners. In the context of these studies, assumptions are defined as the reasoning behind an architectural decision and explicit distinction is made between technical, organizational and management assumptions.

In terms of their impact, assumptions are similar to architectural decisions, yet they represent uncertainty about the system and its environment. The potential impact of assumptions [50] on software development is significant. Garlan [10] refers to implicit architectural assumptions as one of the causes of architectural mismatch, impeding systematic reuse of software components. In practice, keeping track of and documenting architectural assumptions is a costly and non-trivial task, mainly because clear definitions, guidelines and tools are lacking [49].

In addition, assumptions are dynamic in nature [19, 51]: they can be valid or invalid, and this status may change over time as the system or its environment evolves. Assumptions may contradict, subsume, and replace other assumptions. They can change form over time, into requirements, design decisions, or they may become invalid or irrelevant over time. Documenting and keeping track of assumptions is called *assumption management* [19]. The mapping study of Yang et al. [50] highlights the importance of explicit assumption management throughout the software development life-cycle. Other relevant development activities include the retroactive reconstruction of architectural assumptions [25].

2.2 Threat modeling

Threat modeling involves the elicitation and mitigation of security and/or privacy threats [22,48]. It is a systematic process that entails: (i) Modeling the system under analysis, commonly in the form of Data Flow Diagrams (DFDs) which represent the system under design as a combination of data flows, entities, processes, data stores and trust boundaries; (ii) Threat analysis in turn involves instantiating threats in the context of the DFD, documenting and prioritizing threats, and (iii) Mitigation through selecting appropriate countermeasures to reduce the identified risks.

STRIDE is a threat modeling approach for security threats [14,30]. It is an acronym of: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege, which are threat categories or threat types that represent goals of potential attackers. For each of these categories, threat trees are provided. These provide hierarchical refinements of the threat categories that can be used to further concretize threats.

The relevance and applicability of each threat type is rigorously investigated in the context of the system under analysis, at the level of the elements or interactions depicted in the DFD [30,38]. Threats considered relevant are subsequently documented (instantiated as threat scenarios in accordance with specific threat specification templates) and added to the corpus of requirements that should be taken into account during design and implementation. As such, threat modeling is tightly intertwined with requirements engineering and architecture design [13,40]. Threat modeling starts with establishing initial architectural abstractions of the system (encoded in DFDs), and leads to architectural decisions that mitigate the identified threats.

2.3 Assumptions in threat modeling

Literature on threat modeling explicitly stresses the importance of assumption management and documentation. Torr et al. [37] use the term 'implementation assumptions' whereas Microsoft's SDL book [14] refers to 'security assumptions', and defines them as *"the guarantees you expect from the external dependencies"*. Haley et al. [12] discuss the role of trust assumptions in the context of security requirements. Shostack [30] advises to document assumptions during the threat modeling process. The template for recording assump-

tions suggested by Shostack strongly emphasizes on aspects of life-cycle management, such as a procedure to verify assumptions and the impact if falsified, and the appointment of a responsible to follow-up in such a case ('*Impact if wrong*', '*Who to talk to*', '*Who's following up*'). Additionally, Shostack stresses the need to continuously validate assumptions throughout later development activities and integrates these validation steps in issue tracking systems. As to the description of the assumption itself or its impact, a free-form textual notation is suggested.

When it comes to the support for assumptions in tooling, Microsoft's Threat Modeling tool [21] provides support for documenting assumptions as meta-data attributed to the DFD model via a free-from and unstructured text input field. SPARTA [31,32], our own tool implementation of STRIDE that emphasizes on automation through model-driven engineering, currently does not yet support explicitly documenting or considering assumptions in the threat elicitation process.

In threat modeling and threat analysis, assumptions play a significant and non-trivial role. For example, the goal to prevent repudiation of user actions in a system may lead to the architectural decision to keep system logs in which all actions are recorded, or alternatively, the active decision to not address the threat altogether. Both decisions rely on implicit assumptions: in the former case, that the solution of keeping logs sufficiently guarantees that the user will not be able to still repudiate its actions (e.g., the logs themselves are accurate and tamper-proof and thus provide sufficient proof), in the latter case that the user is sufficiently trustworthy and/or will not have an incentive to deny actions.

2.4 Motivation

Assumption-making is an integral part of threat modeling, and the documentation of such architectural assumptions contributes to the body of architectural knowledge [17], i.e. knowledge about the system under construction or the construction process itself.

When assumptions are tightly coupled to or highly specific to the system, they represent architectural information and have to co-evolve with the system and its design documentation. Especially when the system changes, or when assumptions are invalidated, the implications on the overall threat analysis may become non-trivial, e.g. unanticipated threats have to be mitigated, or ripple effects emerge that invalidate the findings. As such, they potentially have large impact on the effectiveness and cost-efficiency of threat modeling efforts.

In addition, assumptions play a significant role in the security architecture of a system as they affect the security properties of the ensuing system. In practice, successful attacks on systems often come down to invalidating assumptions. Many security vulnerabilities are based on invalid assumptions or deliberate breakage of assumptions [46].

However, concrete guidelines are lacking on how to document and manage assumptions in this specific context. It is particularly unclear (i) what is the intended role, purpose or function of assumptions in threat modeling, and (ii) what types of assumptions are meaningful in this specific context.

As such, we want to improve our understanding about the nature of the information commonly encoded in the assumptions made during threat elicitation. More specifically, we focus on identifying (i) to which extent they provide additional information about the decisions and rationale that drive the threat elicitation process itself (RQ1), to which extent they postulate characteristics or properties about the system under analysis and which types of characteristics (RQ2), and to which extent they provide additional information about the documented security threats, the interpretation of specific threat categories, or assumed security countermeasures and mitigations (RQ3). These broad research questions are further refined in the following section.

3 Study design

To gain insight in the nature and function of the assumptions made in practice during threat elicitation, we conduct a *controlled descriptive study*[1] of a collection of assumptions made in STRIDE. We target the following concrete research questions:

> **RQ1.** *What is the role or function of assumptions in the threat elicitation effort?*
> **RQ2.** *How tightly coupled are these assumptions to the system under analysis (its functionality, its context, its model representation)?*
> **RQ3.** *How specific are the assumptions to security?*

The Goal-Question-Metric (GQM)[2] tree [2] that underpins the descriptive study is depicted in Fig. 1. As shown, we have defined nine distinct metrics or variables. These are introduced below in Sect. 3.2.

[1] The main purpose of a descriptive study is to generate insights and increase understanding of a phenomenon through observation, data collection, categorization and characterization [11].

[2] A GQM tree is a hierarchical structure that gradually refines an overall goal into specific questions and further concretizes these questions by stating relevant metrics that allow discrimination. As such, it particularly suited to structure this descriptive study.

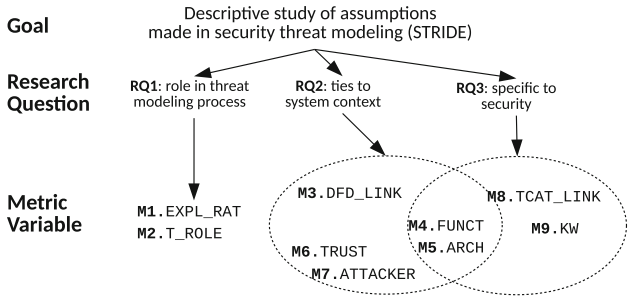


Fig. 1 Goal-question-metric structure of the descriptive study

Table 1 Overview of the participants and assumptions in the study

Total	Student participants	Expert participants
# Participants	91	5
# Threats	630	52
# Assumptions	602	38

3.1 Study data

As shown in Table 1, we have collected data in a STRIDE study involving 91 master students and 5 expert participants. These threat models are the outcome of applying per-element STRIDE[3].

Application case. The threat elicitation step was applied to an industrial document generation Software-as-a-Service (SaaS) application, for which the Data Flow Diagram (DFD) is presented in Fig. 2. This subset of the application deals with delivery of generated documents (stored in DS1 in Fig. 2) via a range of channels such as E-mail (E3 in Fig. 2), via on-line banking (E2 in Fig. 2), or via printed postal delivery service (E1 in Fig. 2). Additional access to the generated documents is offered to recipients via a personal archive, called the Personal Document Store (PDS) (P3 in Fig. 2).

An example of a relevant Information Disclosure threat in this application case involves guessing or enumerating the document identifiers to obtain access to documents that belong to different users (e.g., via data flow `DF8` depicted in Fig. 2).

Data acquisition and treatment. The data was obtained in a controlled setting: to avoid any external influence or communication between the participants, the exercise was conducted on paper in a scoped exercise that was time-boxed in a 2.5 hour session. To meet these timing requirements, participants were asked to focus exclusively on Spoofing, Tampering, Repudiation and Information Disclosure threats. In addition, only a subset of the entire application case was subjected to analysis (the document delivery part, and not the actual generation of documents) and the trust boundary (as depicted in Fig. 2 with a dashed line) allowed participants to focus exclusively on threats in the context of the system (i.e., originating on the outside of the system). Despite these limitations, this still allowed for the identification of a sufficiently diverse set of relevant application threats.

Participants were provided with this Data Flow Diagram (DFD) of the document generation application which was a system they were strongly familiarized with (as they have worked on this system in the context of an intensive software architecture project). In the exit questionnaire of the study, when asked about this, 98% of participants agreed that they were sufficient familiar with the document generation system to complete the task. When asked whether their domain knowledge of the document processing system was useful to conduct the exercise, 94% of students agreed.

Master student participants had limited prior experience with STRIDE: in the intake questionnaire, only 41% claimed to have some prior awareness, most of these claiming familiarity only with the keywords of the acronym but not the systematic threat modeling approach. To mitigate this, an introductory lecture was given to these students, and they were provided with additional printed resources during the exercise (printed slides and all resources such as threat trees). When asked in the exit questionnaire whether they found that this lecture and the supported materials were sufficient to understand the STRIDE methodology, 85% of participants agreed.

In addition to the threat models created by master students, we have obtained threat models from 5 experts, experienced practitioners of secure software engineering and STRIDE. This additional data is used to augment and enrich the data set, but is insufficient in size to draw statistical conclusions about the extent to which student assumptions would differ significantly from expert assumptions.

The main outcome per participant is a threat model, comprising of (i) a mapping table that provides an overview of the identified threats, (ii) individual threat descriptions, and (iii) a separate sheet of assumptions made during the exercise that lists textual assumptions. For illustration purposes, "Appendix A" presents an excerpt of such an assumption sheet.

Figure 3 depicts the overall data acquisition process. Three student participants were excluded as they did not manage to participate or complete the exercise in a satisfactory manner (no results). The study was conducted on paper and an extensive digitization step was first performed, which also involved pseudonymization of subjects, correcting of spelling mistakes and expanding abbreviations. In addition, (and as explained in Sect. 3.2), some text-based manipulations were performed to expand textual, non-verbatim references to DFD elements for the sake of analysis.

[3] This is the variant of STRIDE that involves considering security threats at the basis of DFD elements, as opposed to the per-interaction variant, that focuses on the identification of threats at the basis of specific interactions among DFD elements [30].

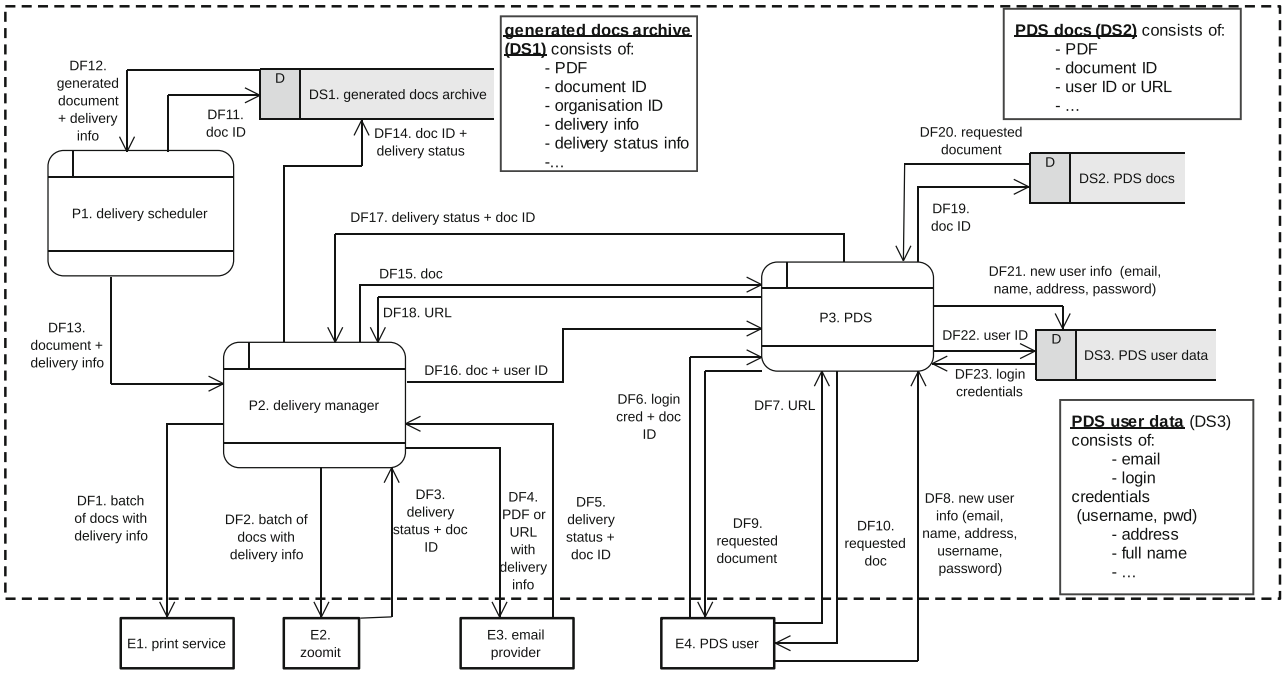


Fig. 2 The data flow diagram (DFD) of the document processing applications, as used in the study

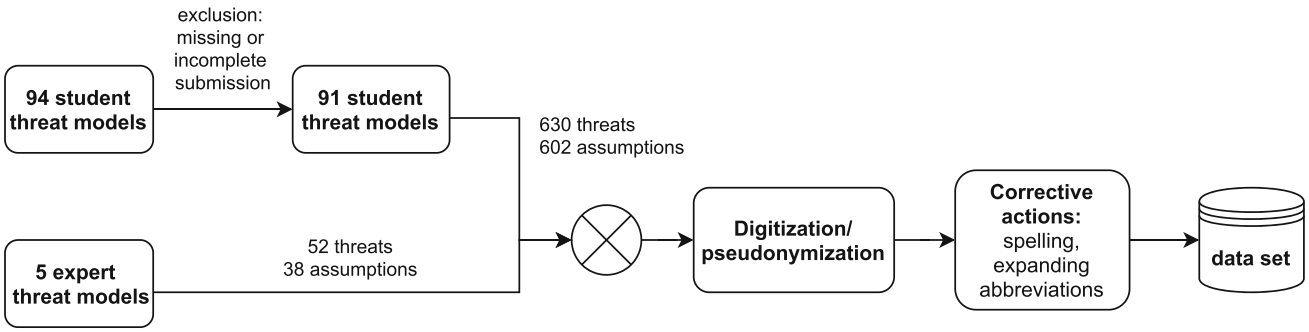


Fig. 3 The data acquisition and treatment approach that involves combining threat models obtained from master students and experts in a controlled and time-boxed session

Data quality Apart from the exclusion of missing and incomplete student submissions (3 in total), we have refrained from a-priori removal of data points at the basis of quality (e.g., the validity of the assumptions). Assumptions that are considered invalid or the consequence of mistakes, for example against the system or in the execution of the threat modeling approach, are considered equally interesting from the perspective of the study goals which are of a broad, descriptive nature. Section 5.1 provides a further view on the quality of the data set in terms of the notions of plausibility and redundancy of each assumption.

Obtained data. Figure 4 plots the obtained threat models in terms of the amount of threats and the amount of assumptions included, variables that are both normally distributed. Applying the Pearson Chi-squared test indicates that the null-hypothesis of independence between both variables cannot be rejected ($p = 0.2099$, $\alpha = 0.05$). There is in this data no discernible correlation between the amount of threats and the amount of assumptions in the threat models obtained for this system.

Figure 5 plots the distribution of the respective textual length (word count) of the collected assumptions, which is a positively skewed distribution.

3.2 Variables and metrics

We have defined 9 metrics/variables to answer the research questions[4]. Some of these variables are categorical (nominal)

[4] As explained in Sect. 1, these are identical to the metrics/variables used in our earlier study on LINDDUN assumptions [44] as we adopt the same classification methodology.

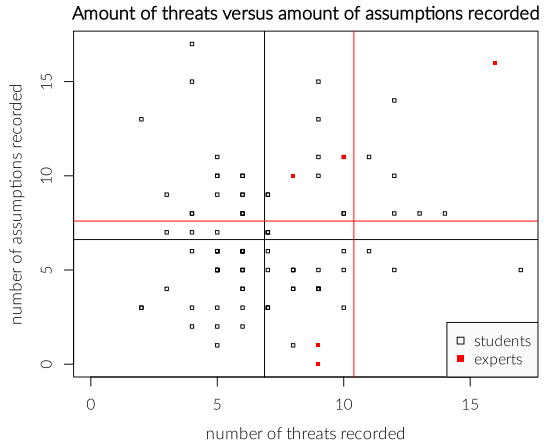


Fig. 4 Overview of the data set of threat models, in terms of the amount of threats and assumptions. The horizontal and vertical marker axes represent the mean values of the respective data sets

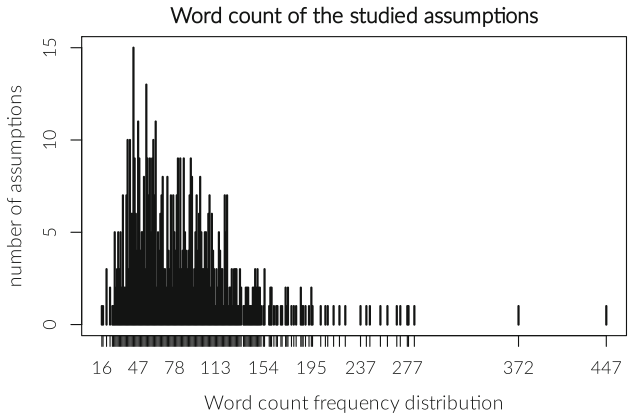


Fig. 5 Overview of the investigated assumptions, in terms of word count *(median:* 85 *words, sdt. dev.:* 48 *words)*

in nature and obtained through expert assessment, whereas other variables are count-based and calculated with text-based analysis techniques.

M1 `EXPL_RAT`. Assumptions made during threat elicitation play a certain role in the context of that activity (i.e., it exerts an impact on threats). This first metric quantifies the degree to which this role is explicitly documented as part of the assumptions.

To accomplish this, we impose the format '*condition* → *role*' on the assumptions, in which *condition* stipulates a certain property of interest and motivates the assumption (i.e., a rationale is given), and the *role* documents the impact on potential threats: the assumption either motivates the inclusion of specific threats (include), the exclusion (exclude), or further explains the priority attributed to a threat (prioritize). The `EXPL_RAT` categorical variable is defined as follows:

$$\texttt{EXPL\_RAT}(a) = \begin{cases} \texttt{condition} : a \text{ only describes a property of interest} \\ \texttt{role} : a \text{ only describes its role} \\ \texttt{both} : a \text{ describes both} \end{cases}$$

M2 `T_ROLE`. Further focusing on the actual role of the assumptions, we make distinction between assumptions whose role it is to *exclude* threats, *include* threats, or have an impact on the *prioritization* of threats. `T_ROLE` is defined as a nominal variable that classifies assumptions as follows:

$$\texttt{T\_ROLE}(a, tm) = \begin{cases} \texttt{exclude} : a \text{ explains threat absence in } tm \\ \texttt{include} : a \text{ explains threat occurrence in } tm \\ \texttt{prioritize} : a \text{ explains threat priority in } tm \\ \texttt{undetermined} \end{cases}$$

For those assumptions in which the role is not documented explicitly, the assessment is performed manually. The role of an assumption depends highly on the specific context of the threat model (tm) in which the assumption was made. Therefore, in such cases, this exercise involved manual interpretation of the documented threats, priorities given and the mapping tables.

M3 `DFD_LINK`. This metric involves counting the explicit references to DFD elements in the textual description of the assumptions, and is defined as follows:

$$\texttt{DFD\_LINK}(a) = n, \text{ the amount of references in } a \text{ to } \textit{distinct} \text{ DFD elements}$$

We have implemented this metric with the pattern matching libraries of the R statistical data processor. This involved constructing regular expressions that match with the specific identifiers given to the elements in the DFD (e.g., "`DF11`" or "`delivery scheduler`").

To increase accuracy, we have manually pre-processed the assumptions to identify existing textual references to DFD elements and augment them with the explicit DFD element identifiers (between brackets). In some cases, this entailed straightforward extension, e.g. changing the occurrence of '`PDS User`' to '`PDS User (E4)`'. In other cases, this involved inserting a more extensive listing of DFD elements: e.g., extending the textual reference to '`the data flows mentioned in Assumption #2`' with '`DF1, DF2, [..], DF9`', which is an explicit enumeration of the referenced data flows. Indirect or vague references left open for interpretation were not counted, e.g. a

reference to '*data stores*' without further naming or concretizing them is not taken into account.

M4 `FUNCT`. This variable involves identifying whether the assumption refers to or relies upon system functionality, and makes distinction between functionality that contributes exclusively to security (i.e. countermeasures or other mitigations assumed to be in place) and broader types of system behavior. This classification is performed manually.

$$\texttt{FUNCT}(a) = \begin{cases} \texttt{n/a} : a \text{ does not refer to functionality} \\ \texttt{functionality} : a \text{ refers to system functionality} \\ \texttt{security} : a \text{ refers to functionality that } \textit{exclusively} \\ \quad \text{contributes to security} \end{cases}$$

M5 `ARCH` expresses the extent to which the assumptions refer to architectural information about the system under analysis that is not present in the DFD. We more specifically distinguish between (i) assumptions that further extend or refine the data flows depicted in the DFD (e.g. assumptions about the nature of the involved data types), (ii) assumptions related to the trust boundary explicitly modeled in the DFD, and (iii) on architectural information that could have been provided in complementary architectural views (e.g. in an client-server view, an implementation view, a deployment view, a decomposition or module view, a process view, etc).

Again, this involves interpreting the assumptions and thus this classification is performed manually.

$$\texttt{ARCH}(a) = \begin{cases} \texttt{n/a} : a \text{ does not refer to system architecture} \\ \texttt{DFD} : a \text{ clarifies or extends information about data flows} \\ \quad \text{in the DFD} \\ \texttt{trust boundary} : a \text{ clarifies/refines the trust boundary} \\ \quad \text{in the DFD} \\ \texttt{other} : a \text{ refers to information of different architectural} \\ \quad \text{viewpoints} \end{cases}$$

M6 `TRUST`. The level of trust bestowed upon an external entity or party (an organisation, a technology, a service) influences the decision to document a threat and the determination of its priority. The mechanism of trust is one way to externalize the responsibility of threat mitigation, e.g. by assuming party '`The E1, E2 and E3 are trustworthy and will not disclose documents`', we do not have to take explicit countermeasures to mitigate potential document leakage to third parties. Therefore, we expect assumptions to refer to this notion of 'trust' (either in external entities as depicted in the DFD, or in external parties not depicted). Implicit references to trust include mentioning of service-level agreements (SLAs), or refer to expected countermeasures or best practices from external entities. Explicit references literally refer to the terminology of 'trust' or 'trustworthiness'. The `TRUST` variable is defined as follows:

$$\texttt{TRUST}(a) = \begin{cases} \texttt{n/a} : a \text{ does not refer to trust} \\ \texttt{implicit} : a \text{ implicitly refers to trust} \\ \texttt{explicit} : a \text{ explicitly refers to trust} \end{cases}$$

M7 `ATTACKER`. Threat modeling is inherently attacker-centric. Similar to the above variable, the decision to document or prioritize threats is influenced by the assumptions about potential attackers. This variable, `ATTACKER`, therefore identifies the extent to which an assumption refers to potential attacker(s), attacker goals, incentives or capabilities.

$$\texttt{ATTACKER}(a) = \begin{cases} \texttt{n/a} : a \text{ does not refer to the attacker} \\ \texttt{implicit} : a \text{ implicitly refers to attacker} \\ \texttt{explicit} : a \text{ explicitly refers to attacker} \end{cases}$$

M8 `TCAT_LINK` determines the link between assumptions and threat categories, by counting the amount of explicit references to the STRIDE threat categories. We more specifically look for threat categories explicitly referenced by name, or the use of a derivative keyword. For example, we count '`repudiation`', '`repudiate`', but not '`deny`'.

We define the metric as follows:

$$\texttt{TCAT\_LINK}(a) = n, \text{ amount of references in } a \text{ to } \textit{distinct} \text{ threat categories}$$

The metric is calculated in an automated fashion, but relies on a manual pre-processing step that deals with aspects of disambiguation and interpretation. Multiple references to the same threat category are counted only once.

M9 `KW` The final variable, `KW`, quantifies the amount of explicit references to security-related concepts and keywords. The metric depends on the establishment of a security *lexicon* L. Based on the involved terminology, this lexicon makes distinction between attack-related, countermeasure-related, goal-related and other terminology. A keyword-based classification of an assumption a is performed as follows:

$$\texttt{KW}(\text{a, L}) = \begin{cases} \texttt{attack} : a \text{ includes } \texttt{attack} \text{ keywords from } L \\ \texttt{counter measure} : a \text{ uses} \\ \quad \textit{counter measure} \text{ keywords from } L \\ \texttt{goal} : a \text{ uses } \textit{goal} \text{ keywords from } L \\ \texttt{other} : a \text{ uses other keywords from } L \\ \texttt{none} : a \text{ includes no terms from } L \end{cases}$$

Again, this metric is applied in an automated fashion using pattern matching.

Table 2 Excerpt of the security lexicon L used in the study

Keyword	Category
Access control	Countermeasure
Backdoor	Attack
Certificate	Countermeasure
Confidential	Goal
Credential	Countermeasure
SQL injection	Attack
Unregistered	Other

Table 3 The results of applying `EXPL_RAT`

`EXPL_RAT`	Student assumptions	Expert assumptions
Condition only	422 (70.1%)	22 (57.89%)
Role only	28 (4.65%)	2 (5.26%)
Both condition and role	152 (25.2%)	14 (36.84%)

The security lexicon was established by manually highlighting security-related keywords in the assumptions (using text frequency analysis). In an additional step, we asked an independent expert on secure software engineering to verify the keywords and to annotate them with the most appropriate category (attack, countermeasure, goals, and other). Table 2 shows an excerpt of the resulting lexicon L (which consists of in total 71 keywords and is provided in the supporting materials [42].

4 Results

This section presents the results obtained by applying the variables and metrics defined in the previous section to the corpus of threat models. In all cases, the results for both data sets (students and experts) are presented separately.

4.1 RQ1: role of assumptions

We first examine the extent to which the *role* or the rationale behind the assumptions is explicitly documented in their textual description (the `EXPL_RAT` variable). The `EXPL_RAT` results are presented in Table 3.

In both data sets, the majority of assumptions did not provide an explicit explanation about the role of the assumption in the context of the threat model (70.1% of the student and 57.89% of the expert assumptions). An example of such assumption is "`passwords are encrypted`", which does not directly discuss the role or implications of that assumption in the threat model.

Most rare were assumptions stating an impact on threats without any explicit assumption or condition of interest (4.65% of the student and 5.26% of the expert assumptions). An example of such an assumption is "`Repudiation by the print service poses no threat`" which lacks any explanation why such threats are excluded.

Table 4 The results of applying `T_ROLE`

`T_ROLE`	Student assumptions	Expert assumptions
Exclude	351 (58.31%)	28 (73.68%)
Include	144 (23.92%)	7 (18.42%)
Prioritize	95 (15.78%)	1 (2.63%)
Undetermined	12 (1.99%)	2 (5.26%)

25.25% of the student and 36.89% of the expert assumptions explicitly state *both* a condition and the role, i.e. they complement the assumption (condition) with a discussion of the implications in terms of the security threats. An example is "`Due to the encryption of data flows [..], P2 is immune to tampering by passing malicious parameters. [..]`".

We have manually categorized the assumptions based on their impact on threats (`T_ROLE`). The results are summarized in Table 4.

The majority of assumptions were meant to exclude threats, i.e. to remove them from further consideration: in the student data set, this amounts to 58%, and in the expert data set to 73.68%. An example of such an assumption is "`Encryption keys are stored safely`', which eradicates a number of threats caused by encryption keys obtained by an adversary.

In the student data set, 23.92% of assumptions were found to explain or motivate the occurrence of specific threats, similarly in the expert data set this is 18.42%. An example of such an assumption is "`the data flow DF4 channel not encrypted`", which allows for a Tampering threat scenario in which unprotected data is changed in transit by a malicious adversary.

15.78% of the student assumptions were used to prioritize threats, and this was less common in the expert data set (2.63%). An example of such an assumption is the assumption that "`The P3 and the data stores are each on another machine`", which implies that if an attacker can take control of the machine that hosts the P3 process, he or she can not directly access the data stores which lowers the impact and thus the priority of such threats.

For the assumptions classified as 'undetermined', we were unable to retro-actively identify their role in the context of the threat model. For example, we were unable to trace back the role of assumption "`communication between P2 and E3 happens through sockets`" in the threat

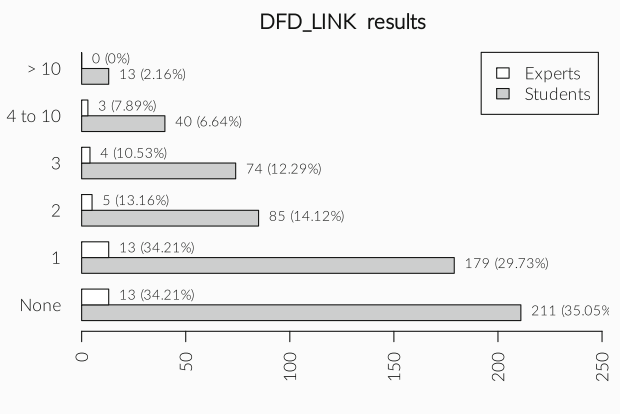


Fig. 6 Results of counting the number of references to the elements of the DFD (DFD_LINK)

model in which it was stated. It was simply not clear why this assumption was made.

4.2 RQ2: system context

The results of applying the DFD_LINK metric are presented in Fig. 6. In total 65% of all assumptions include a direct reference to a DFD element, most of which refer to just one DFD element. Only in rare cases do assumptions refer to more than four distinct DFD elements (8%).

An example of an assumption that applies to many DFD elements is: "Data flows between P3 and DS3 (DF21 DF22 DF23) should have higher priority than DF19 and DF20, because user data is much easier to make use. And since the attacker gets user data, it's easier for the attacker to login to P3 and get specific documents of these users".

An example of an assumption that is tightly coupled to the DFD only after the pre-processing step: "the data flows internal to the eDocs system are all happening within a local area network, so we don't consider them to be tampered with". Although the assumption does not literally refer to DFD elements, the reference to 'data flows internal to the eDocs system' implicitly refers to a large set of DFD elements and is thus counted as such.

The results of applying the metrics that further investigate the nature of the information postulated in the assumptions (FUNCT, ARCH, ATTACKER, TRUST) are percentage-wise summarized in Fig. 7.

The results of FUNCT show that of all studied assumptions, over 60% of the assumptions postulate a form of system functionality, of which the largest part is exclusively related to security (40% of all assumptions). An example of such an assumption is the assumption that "passwords are encrypted". An example of an assumption postulating system functionality not related to security or privacy is "a customer does not need to verify the creation of his account".

Applying the ARCH metric indicates that 16% of the assumptions were used to further refine data flows in the DFD (e.g. "*login credentials are sent hashed*"), and around 13% used to instill semantics upon the trust boundary (e.g., "eDocs employees can access the data stores from the outside"). 7% of the studied assumptions refer to other architectural elements, decisions or considerations that are not included in a DFD. Examples of these types of assumptions are "E-mail traffic is sent over the public internet", and "DS1 and DS2 are replicated on multiple nodes", both examples of assumptions that would be difficult to express in a DFD and for which complementary architectural views would be more suited (a deployment or a network topology view). This is information that would be presented in a complementary deployment model or view but cannot be expressed in a DFD.

The ATTACKER results show that few assumptions were found to refer to the attacker: only around 5% of assumptions do so implicitly, and only 4% explicitly.

Examples of assumptions that *implicitly* refer to attacker capabilities are:

- "The employees are trained in password management. Therefore they all have an unguessable, strong password which is secretly stored in their mind", which *implicitly* states that an attacker does not have the capability to guess the password or obtain it otherwise,
- "Attacker can use vulnerabilities/ backdoors to get access to internal data flows".

Conversely, an example assumptions that *explicitly* refers to attacker capabilities or incentives is: "There is no point for an attacker to tamper with the dataflow for registration".

Applying the TRUST metric yields similar results: 7% of the studied assumptions were found to implicitly rely on trust in external parties or entities, yet again a smaller set of assumptions was found to do so explicitly (5%).

Examples of assumptions that include an *implicit* reference to trust:

- "External delivery services are assumed to be professional [..] and have proper authentication [..]".

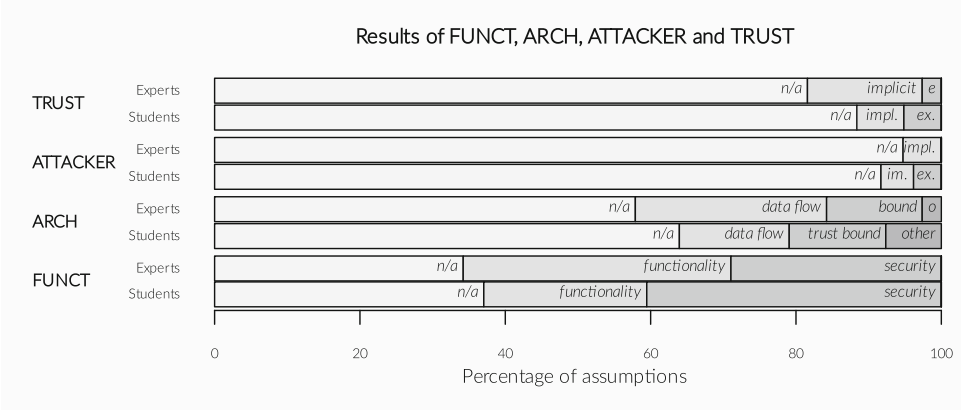


Fig. 7 Summary results of FUNCT, ARCH, ATTACKER, TRUST. (labels: *o*: other, *e/ex.*: explicit, *im.*: implicit.)

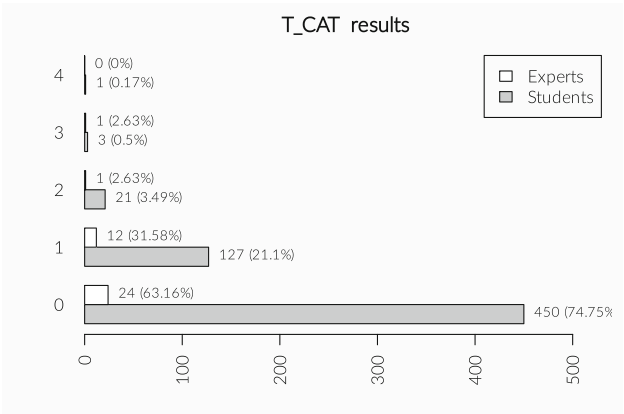


Fig. 8 Results of linking assumptions to the STRIDE threat categories (T_CAT)

– 'With each of the external service providers, an SLA has been negotiated. They will behave accordingly [..]".

Conversely, examples of assumptions that *explicitly* refer to trust are:

– "Edocs uses a trustworthy E-mail provider",
– "External services are assumed to have no malicious intent but are however still not trusted".

4.3 RQ3: relation to security

Figure 8 presents the results of the T_CAT assessment. Only around 26% of assumptions include an explicit and direct reference to a threat category or threat type.

The results of keyword analysis (KW) shown in Fig. 9 shows that 35% of assumptions lacks any reference to security terminology. Although they are documented in the specific context of a security threat modeling and analysis exercise, a majority of these assumptions are more broadly relevant in the context of architecture knowledge management.

An example of an assumption that extensively uses security terminology: "Invoices have confidentiality and integrity because they are signed with a key".

5 Discussion and findings

This section discusses the results presented in the previous section, as well as implications and recommendations for future work. Section 5.1 first discusses the most relevant threats to validity of this empirical study. Sections 5.2 to 5.4 individually discuss the findings for respectively RQ1-RQ3.

5.1 Threats to validity

A descriptive study is a form of empirical study, in which phenomena are studied through observation [11] and as such, threats to the validity of the findings and the results should not be ignored.

We discuss threats to validity related to (i) the focus on a single application case, (ii) the data acquisition strategy, (iii) involvement of master students, (iv) data treatment, (v) bias introduced by interpretation, (vi) bias introduced as a result of textual manipulation of the assumptions and finally, (vii) applicability of the findings beyond STRIDE.

(i) Single application case We have assessed the nature of the assumptions made in the context of a specific system, a document generation SaaS application, and therefore our findings about assumptions may be specific to this application (type). Possibly, in other applications, different types of threats may play a more significant role (e.g., Denial of Service threats that were not considered in this particular system) and these different threat categories might influence the nature of assumptions made during their elicitation.

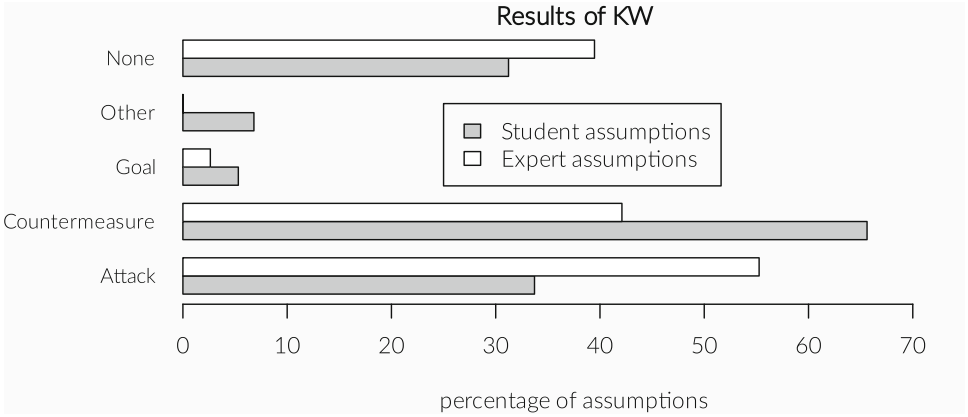


Fig. 9 Results of keyword analysis (KW)

We as such acknowledge this possible threat to validity that may imply that our findings would be biased towards the nature of application studied, and further investigation and evaluation is required towards understanding the extent to which the (i) nature of an assumption is influenced by the threat category under investigation, or (ii) the nature of the application under analysis and its inherent threat profile.

However, the adopted SaaS application is highly realistic, as it was adopted from a collaborative research project with industry, and as any cloud application, it is subject to a number of challenging and non-trivial security requirements.

(ii) Data acquisition strategy As explained in Sect. 3, the data set (consisting of threat models) taken into account in this study was obtained in a controlled, restricted and time-boxed setting: the exercise was held in a 2.5 hour time window, participants were denied access to external resources (other than the course material provided), the presented DFD was a reduced version of the entire system and students were asked to focus on Spoofing, Tampering, Repudiation and Information Disclosure threats and exclude Denial of Service and Elevation of Privilege threat categories. These measures were all taken to ensure practical feasibility and completion of the exercise in the provided time window, but in turn may have had a specific effect on the outcome: as discussed above, by not focusing on different application cases and different threat types, specific findings may not be straightforward to generalize.

However, as the descriptive study focuses on getting a better view on the degree of variation in the nature and function of these assumptions in this specific threat modeling context, the restricted and controlled nature can also be considered beneficial as opposed to the alternative of a more open-ended, free-form exercise. In this study, the participants started with identical inputs, yet the diversity in the output assumptions is clearly shown.

(iii) Student participation Master students may not be representative of practitioners when it comes to their prior understanding of security threat modeling and thus our findings may not be applicable to assumptions made in the context of threat modeling conducted by expert or experienced practitioners.

In the study questionnaire, 41% of students claimed some familiarity with STRIDE beforehand, whereas 98% of student participants stated that they were made sufficiently familiar with STRIDE through the introductory lecture and the supporting materials. Noteworthy, 85% indicated that they found the specification of assumptions difficult.

We argue firstly that the main goal of this study was not evaluate the data in terms of the actual validity of the threats identified in the context of the case nor that of the assumptions, which are both aspects that may be heavily influenced by expertise and experience. Instead, we set out not to characterize how experienced practitioners use the mechanism of assumption-making, but to broadly explore the different functions assumptions may fulfill in this context. As such, we argue that lack of expertise or experience may play a smaller role in our findings.

Secondly, to further mitigate the problem of variation between participants, we have conducted our descriptive study in a strictly controlled context: students attended the same mandatory introduction session, were provided with the same supporting materials, and were given the same time window to conduct the assignment.

Thirdly, results of the comparison study performed by Salman et al. [26] indicate that the discrepancy between students and practitioners may be smaller than commonly expected.

Conducting a confirmatory descriptive study at a similar scale involving more experienced practitioners and industry participants would evidently be highly desirable, but this is no straightforward endeavor as it would also raise a number of practical issues (finding a large set of representative participants, maintaining comparability, controlling external

variables such a system and domain knowledge, prior experience, etc).

(iv) Data treatment Building upon the previous argumentation, we deliberately did not conduct an extensive exclusion of data points to remove those threat models that could be considered to be of low quality or invalid. Evaluation of quality or validity of assumptions would be highly difficult if not impossible: even if a specific assumption turns out to be invalid, this may either be a consequence of the threat modeler making mistakes against the system scope (which may be due to limited prior knowledge of the system under design), or the threat modeler not performing the threat modeling correctly.

In the latter case (incorrect execution of threat modeling), the assessment of validity would take on a normative stance: whether the stipulation of the assumption is to be considered correct in the context of threat modeling, would assume that such best practices exist and are well-documented, and this study is exactly a first step towards better knowledge of the function of assumptions in this context in a longer trajectory towards solidifying guidelines.

In the former case (mistakes w.r.t. the systems), even in such cases, we still consider it interesting and worthwhile to record what is the function of the invalid assumption in the broader context of the descriptive study. The main argument behind this is that invalid assumptions are considered equally interesting in the context of the study goals, which are of a broad descriptive nature.

To obtain a better view on the data quality, we have performed another iteration over the assumptions, assessing (i) whether or not the assumption is considered *plausible* in the context of the system under analysis and the threat modeling exercise, and (ii) if the assumption is considered redundant, because the information is already specified elsewhere (and the inclusion of the assumption is thus not necessary to draw the conclusion), e.g. in a different assumption, or in the DFD[5]. As with the other assessments, both variables have been scored by expert assessment, and we have generally taken a conservative stance, ranking an assumption as implausible and/or redundant only when this is objectively and clearly the case. This additional verification has led to the following insights:

- 39 out of 602 student assumptions (6.5%) was considered implausible, and 23 out of 91 student participants make at least one implausible assumption. 1 out of 38 expert assumptions (2.6%) was considered implausible, with obviously 1 out of 5 experts making at least one implausible assumption. An example of an assumption considered implausible is the statement that "`delivery status reports only contain information that is useless to attackers`" because in the system, delivery status reports provide details about recipients having received and/or read a delivered document (an implementation of 'receipt tracking') which leaks meta-information that may be valuable to attackers (e.g., to infer personal information or delivery information such as E-mail addresses) - an Information Disclosure threat. Another such example is "`Payslip/invoice does not contain very sensitive information`".
- 22 out of 602 student assumptions (3.7%) were considered redundant, and 14 out of 91 student participants make at least one redundant assumption. None of the 38 expert assumptions were considered redundant. Two examples of assumptions considered are "`the DS1 has no flow going outside the trust boundary assumption`" and "`P1 can only be accessed internally`", as in both cases, this information is entirely available in the DFD and the assumption itself provides no additional information.

The above numbers give an indication of the overall quality of the data set: while not even implying that redundant assumptions are necessarily a bad practice, they may indicate a certain degree of uncertainty of the participant in terms of their interpretation of the inputs materials. In addition, making implausible assumptions is often more an indication of making an error in judgement, rather than making fundamental mistakes against threat modeling in general.

(v) Researcher interpretation and bias As discussed in Sect. 3.2, some of the investigated variables[6] required us to manually interpret the assumptions in specific context of the threats being documented alongside, and of the system under analysis and in terms of references made to these assumptions. As this was a manual effort, this step potentially introduced interpretation issues or bias.

This validity threat was managed by first creating a clear and unambiguous classification procedure for each variable (these can be found in the supporting materials [42]). Secondly, we adopted a two-phased approach in which two involved experts classified the assumptions separately using these classification procedures, which was then followed by a phase in which discrepancies were discussed and consensus was reached.

(vi) Pre-processing and assumption manipulation As discussed in Sect. 3.2, we have manipulated the textual descrip-

[5] For example, the assumption that "`Data stores are accessed only by the internal components`" is considered redundant as this information is depicted in the DFD.

[6] `T_ROLE`, for those assumptions in which the explicit rationale was not provided explicitly in the description of the assumption, `FUNCT`, `ARCH`, `TRUST`, and `ATTACKER`.

tions of some of the assumptions (i) to sanitize the data (e.g., correct writing errors), and (ii) in preparation of automated processing (i.e., for `DFD_LINK` and `T_CAT_LINK`)). This step may have introduced error and implicit bias.

We managed this threat by adopting a conservative stance in terms of these manual changes. For example, we only replaced substrings when the reference to DFD elements was clear, unambiguous and objectively undebatable. Secondly, in those cases in which such manipulations were performed automatically, the outcome was double-checked manually afterwards.

(vii) Generalization beyond STRIDE We have studied assumptions that were generated in the specific context of per-element STRIDE threat elicitation and as such the findings may only apply to STRIDE threat elicitation. As argued above, the nature of the threat types investigated indeed may impact the nature of the ensuing assumptions, which is something that warrants further investigation.

Some of the studied assumptions are inherently trust- or attacker-centric, and this certainly is a consequence of the inherent attack-centric focus of STRIDE threat elicitation.

Our mirror study conducted over LINDDUN assumptions [44] in the context of a different system largely has similar results, with the main difference that the privacy threat categories are inherently more specific and less intuitive and thus lead to the formulation of more assumptions.

Further investigation is required to assess whether different threat modeling paradigms (e.g., goal-based threat elicitation [41]) involve the stipulation of architectural assumptions of a similar nature. Different threat modeling paradigms are considered to be complementary to scenario elicitation approaches such as LINDDUN and STRIDE [5].

5.2 Findings on RQ1: role of assumptions

The classification of the *role* of an assumption in the threat model (`T_ROLE`) allowed us to provide explanations for 98% of all assumptions studied. However, this role was only explicitly stated in 30% of assumptions (`EXPL_RAT`).

We found that the majority of the studied assumptions (59%) were made for the purpose of explaining why specific threats have not been considered nor documented, i.e. *assumptions are used as a means to scope the threat modeling effort*. Although this may also be caused by the time-boxed and time-constrained nature of the threat modeling exercise conducted by the participants, we argue that such time constraints are also at play in larger-scale exercises, for example in an industry context [7,28].

A substantial set of assumptions (19%) is furthermore used to document pre-conditions to specific threats, i.e. these assumptions contribute to the rationale behind considering these threats to be relevant. These assumptions are as such tightly coupled to the documented threats.

Recommendations/implications Especially for the types of assumptions that exclude (or reduce the priority of) threats, we strongly advocate in favor of improving the traceability and rationale documentation: the impact is substantial if such an assumption is found to be invalid afterwards, and in such case, the threats excluded at the basis of the invalid assumption should be systematically reconsidered. For such purposes, an analyst or architect should be able to retroactively query a threat model to find out why exactly a specific threat was excluded or ignored. This calls for more explicit documentation of the rationale and traceability logging of the decision process itself, not just the outcome or the assumption itself, but both the condition and the role. This in turn will force the analyst to explicitly document and motivate the impact of a specific assumption, with reference to excluded or ignored threats as a consequence.

Assumptions that clarify the existence of threats or the priority given are in turn better documented alongside the specific threat as this information is tightly related. In both cases, the information stipulated in the assumption acts as a precondition to the threat itself.

The above recommendations in turn are key enablers for more systematic impact analysis support and assumption management, for example, to assess the implications when one of the assumptions turns out to be invalid at a later stage, or assumptions are later refined for example, when the assumption that "`sufficient countermeasures are taken to present spoofing`" is replaced with the assumption that "`the system performs authentication`".

5.3 Findings on RQ2: system context

Many assumptions (65%) were found to refer directly to DFD elements (`DFD_LINK`). These assumptions are tightly coupled to the model of the system under analysis, and when that model changes (e.g. as a consequence of continued development or system evolution), these assumptions will have to co-evolve (be revised, rephrased, removed, merging or splitting assumptions, etc).

In terms of the system-related information encoded in the studied assumptions, there is large diversity. Many assumptions postulate the existence of specific security-related countermeasures or mitigations. Other functionality-related assumptions postulate system behavior, for example about the control flow of the application (e.g. "*the customer is registered before registering a destination address*"). The main use of such assumptions is to circumvent the limited expressiveness of the DFD notation in which functionality or control flow logic is not expressed as a first-class element, as these are merely represented in terms of the data flows in the system.

Secondly, the DFD notation is largely agnostic of security countermeasures, and in the best case, these are represented in terms of their effects on data flows (e.g. credentials or tokens being shared between processes is the visible representation of an authentication mechanism being in place).

The results of the ARCH metric confirm the problem: many assumptions exist exclusively to refine or postulate specific properties of the data elements involved in the data flows depicted in the DFD, or the trust boundary itself.

Recommendations/implications Although initial steps have been taken to augment DFD notations with for example countermeasure information [33], or even to instill formal security semantics onto DFD models [9,34,36], these observations lead us to recommend altogether reconsidering the use of DFDs as the single source of system information in threat modeling.

While DFDs are easy to comprehend and create, they clearly lack the expressiveness to represent all information of relevance for efficient threat elicitation. More concretely, we advocate in favor of using architectural descriptions that support (i) explicit data modeling and classification (data types), (ii) explicit first-class modeling of countermeasures and mitigations (security architecture) and (iii) to allow the threat modeler to draw upon the information typically encoded in complementary architectural views (deployment, process views, implementation views, etc) [3,15,16].

This will fundamentally change the way in which threat elicitation methods work and calls for improved tooling and automation [31]. Specific to threat elicitation approaches, instead of iterating over all data flows or interactions, threat elicitation can be based on iterating over all threat categories and explicitly checking the individual pre-conditions for individual threat types (e.g. as model-based patterns or constraints). Continuously assessing whether these preconditions are met in the context of a more complex software architecture model will then allow for closer alignment of threat modeling with the other practices of software architecture design.

External elements The ATTACKER and TRUST variables do not focus on inherent properties of the system under analysis but on external elements such as capabilities of attackers and aspects of trust (TRUST) in entities (employees, third-parties). Although these assumptions represent 17% of all assumptions in the studied data sets, only in 7% of assumptions, reliance on these aspects was made explicit.

As mentioned, an elicitative scenario-driven threat modeling approach such as STRIDE is by definition attacker-centric: assessing the viability of specific threats implicitly involves assessing the technical feasibility of that specific attack, but also takes into account the trust bestowed upon insiders and outsiders. Many factors come into play, such as the estimated technical capabilities of adversaries, the system's attack surface, and the incentives and motives of potential attackers. Determining appropriate mitigations is contextually strongly dependent on these elements.

In security architecture (i.e., the process of securing a system against threat by actively introducing appropriate mitigations), aspects of attacker capabilities and trust relations to external parties are highly relevant, yet in practice, they are documented informally or implicitly. This is problematic as these are both inherently *dynamic* in time: trust relations will change over time, attacker capabilities improve, and what was first considered technically infeasible may become possible due to newly-discovered vulnerabilities, technological advancements, data leaks, etc.

Recommendations/implications Therefore, we advocate in favor of approaches that involve explicit modeling of attackers and trust relationships. These models, which are an integral part of the security architecture of a system, can then serve as tangible inputs to be taken into account during threat modeling.

In turn, as the system evolves over time, these models can be revised and co-evolve, and this in turn will again allow for more extensive support for change impact analysis. For example, we could leverage a threat model as a means to automatically assess the impact in case a previously trusted stakeholder were to misbehave by dynamically re-assessing how many and which security threats this would raise (threats that were excluded on the assumption of trustworthiness of said stakeholder).

5.4 Findings on RQ3: relation to security

We found that in the studied assumptions, a substantial subset (78%) was in direct reference to security-related concepts (KW). This is a strong indication that these assumptions will mainly be relevant in the threat modeling effort (i.e. they are part of the threat modeling process rationale documentation) or by extension, the security architecture of the system under design.

In terms of the types of keywords being used (KW), the results show that the assumptions mainly focus on *attack* and *countermeasure*-terminology which is not surprising given (i) the inherent attack-centric nature of the STRIDE security threat modeling approach and (ii) the observation from Sect. 5.2 that most of the assumptions are used to exclude threats from consideration, typically by stipulating existence of appropriate countermeasures in the system.

Recommendations/implications Assumptions that stipulate countermeasures are mainly caused by the lack of expressiveness in the input model. In line with the recommendations described above, we argue in favor of explicitly modeling countermeasures and security primitives in the system models (DFDs or more complicated models) on which threat elicitation is conducted.

Further investigation is required to assess whether security specific assumptions will differ fundamentally from the other types of assumptions discussed in literature, e.g. assumptions made during requirements elicitation [43], or in later stages of the development life-cycle. For example, it will be an interesting track of follow-up research to properly characterize and compare the different types of architectural assumptions encountered in practice, in extension of descriptive mapping studies such as the one of Yang et al. [50].

Although the data shows differences between students and experts—experts seem to adopt a more *attack*-centric vocabulary, whereas students focus more on countermeasure-related assumptions—, we refrain from drawing any strong conclusions due to the limited size of the expert data set. Exploring the differences between expert threat modelers and novices in terms of their focal points is definitely an interesting target for future empirical research in this context.

6 Conclusion

We have presented a descriptive study of 640 assumptions made in the context of STRIDE security threat modeling. We summarize our observations as follows: firstly, the studied assumptions are most commonly used to scope the threat modeling process, i.e. to reduce the amount of threats to be investigated. Secondly, many of the assumptions stipulate architectural information that cannot be expressed in the DFD notation that is commonly used in this context. Thirdly, some assumptions explicitly refer to other notions not typically modeled in a system architecture such as trust relations and attacker capabilities.

Based on our findings, we state the following recommendations for improvement of threat elicitation approaches such as STRIDE primarily and security architecture design practices in general:

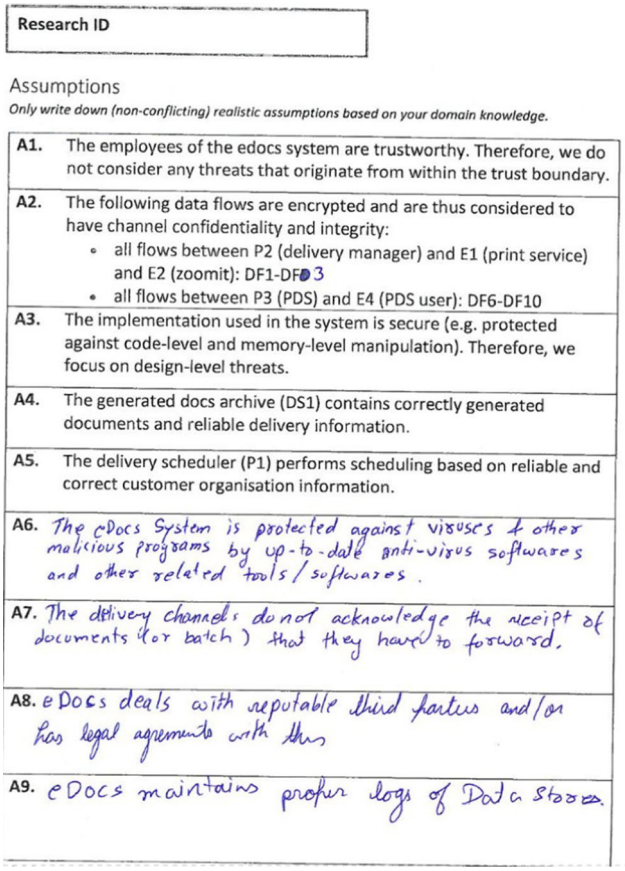
- Proper process rationale and traceability documentation is required when assumptions are used to exclude or ignore threats.
- Assumptions that motivate the relevance of a threat (or the priority given) are more ideally documented as part of the threat documentation itself (e.g., as preconditions).
- We argue in favor of rethinking the role of DFD models as the main input and see value in adopting more expressive architectural descriptions that are structured according to complementary viewpoints.
- Explicit modeling of assumptions related to attackers and their capabilities (e.g., different attacker profiles), as well as trust modeling will greatly benefit not only the threat modeling process but security architecture design in general.

Aside from providing insights and recommendations, our study also provides a systematic and reproducible set of metrics and variables for assessing security-related assumptions. This is a complementary classification tool to for example, the more generic conceptual taxonomy of architectural assumptions proposed by Yang et al. [49]. Further investigation is required to compare the specific assumptions made in this study with architectural assumptions made in a different context.

These results highlight the necessity of further research into the impact of assumptions on the effectiveness of threat modeling in general, for example by quantifying the impact of invalid assumptions in case studies, or the integration of threat modeling in iterative development practices (e.g. continuous or agile threat modeling) which requires further systematization. Especially from the perspective of threat elicitation, these assumptions are relevant inputs that exist outside of the actual models (DFDs). Further improving their documentation (e.g. by annotating them in the input DFD model as a minimal step) will be essential to increase reproducibility and automation.

A Appendix: assumption sheet

See Fig. 10.



Research ID

Assumptions

Only write down (non-conflicting) realistic assumptions based on your domain knowledge.

A1.	The employees of the edocs system are trustworthy. Therefore, we do not consider any threats that originate from within the trust boundary.
A2.	The following data flows are encrypted and are thus considered to have channel confidentiality and integrity: • all flows between P2 (delivery manager) and E1 (print service) and E2 (zoomit): DF1-DF3 • all flows between P3 (PDS) and E4 (PDS user): DF6-DF10
A3.	The implementation used in the system is secure (e.g. protected against code-level and memory-level manipulation). Therefore, we focus on design-level threats.
A4.	The generated docs archive (DS1) contains correctly generated documents and reliable delivery information.
A5.	The delivery scheduler (P1) performs scheduling based on reliable and correct customer organisation information.
A6.	The eDocs System is protected against viruses & other malicious programs by up-to-date anti-virus softwares and other related tools / softwares.
A7.	The delivery channels do not acknowledge the receipt of documents (or batch) that they have to forward.
A8.	eDocs deals with reputable third parties and/or has legal agreements with them
A9.	eDocs maintains proper logs of Data stores.

Fig. 10 Excerpt of an assumption sheet (subject id *32*)

References

1. Babar, M.A., Dingsoyr, T., Lago, P., van Vliet, H.: Software Architecture Knowledge Management: Theory and Practice. Springer, Berlin (2009)
2. Basili, V.R.: Software modeling and measurement: The goal/question/metric paradigm. Technical report, College Park, MD, USA (1992)
3. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice, 3rd edn. Addison-Wesley Professional, Boston (2012)
4. Bazarhanova, A., Smolander, K.: The review of non-technical assumptions in digital identity architectures. In: Proceedings of the 53rd Hawaii International Conference on System Sciences (2020)
5. Bulusu, S.T., Laborde, R., Wazan, A.S., Barrère, F., Benzekri, A.: Which security requirements engineering methodology should i choose?: Towards a requirements engineering-based evaluation approach. In: Proceedings of the 12th International Conference on Availability, Reliability and Security. p. 29. ACM (2017)
6. Chandra, P., et al.: Software assurance maturity model. A guide to building security into software development v1. 0 (2009)
7. Dhillon, D.: Developer-driven threat modeling: lessons learned in the trenches. IEEE Secur. Priv. **9**(4), 41–47 (2011)
8. Feilkas, M., Ratiu, D., Jurgens, E.: The loss of architectural knowledge during system evolution: An industrial case study. In: 2009 IEEE 17th International Conference on Program Comprehension. pp. 188–197 (2009)
9. France, R.B.: Semantically extended data flow diagrams: a formal specification tool. IEEE Trans. Softw. Eng. **18**(4), 329–346 (1992). https://doi.org/10.1109/32.129221
10. Garlan, D., Allen, R., Ockerbloom, J.: Architectural mismatch or why it's hard to build systems out of existing parts. In: Proceedings of the 17th International Conference on Software Engineering. pp. 179–185. ICSE '95, ACM (1995)
11. Grimes, D.A., Schulz, K.F.: Descriptive studies: what they can and cannot do. Lancet **359**(9301), 145–149 (2002)
12. Haley, C.B., Laney, R.C., Moffett, J.D., Nuseibeh, B.: Using trust assumptions with security requirements. Requir. Eng. **11**(2), 138–151 (2006)
13. Heyman, T., Yskout, K., Scandariato, R., Schmidt, H., Yu, Y.: The security twin peaks. In: International Symposium on Engineering Secure Software and Systems. pp. 167–180. Springer (2011)
14. Howard, M., Lipner, S.: The Security Development Lifecycle: SDL: A Process for Developing Demonstrably More Secure Software. Microsoft Press (2006)
15. ISO/IEC/IEEE: ISO/IEC/IEEE Systems and software engineering: Architecture description. ISOIEC/IEEE 42010:2011(E) (2011)
16. Kruchten, P.: The 4+1 View Model of Architecture. IEEE software (1995)
17. Kruchten, P., Lago, P., van Vliet, H.: Building Up and Reasoning About Architectural Knowledge, pp. 43–58. Springer, Berlin (2006)
18. Lago, P., van Vliet, H.: Explicit assumptions enrich architectural models. In: Proceedings of ICSE '05. pp. 206–214. ACM (2005)
19. Lewis, G., Mahatham, T., Wrage, L.: Assumptions management in software development. Techniacl report CMU/SEI-2004-TN-021, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA (2004), http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6941
20. Mamun, M.A.A., Hansson, J.: Review and challenges of assumptions in software development (2011)
21. Microsoft: Microsoft Threat Modeling Tool: documentation. https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool (2019)
22. OWASP: Application threat modeling. https://www.owasp.org/index.php/Application_Threat_Modeling (2017)
23. Piasecki, S., Urquhart, L., McAuley, D.: Defence against dark artefacts: An analysis of the assumptions underpinning smart home cybersecurity standards. Available at SSRN **3463799**,(2019)
24. Ramkumar, M.: Cybersecurity: It's All About the Assumptions. National Cyber Summit (NCS), Huntsville pp. 8–9 (2016)
25. Roeller, R., Lago, P., van Vliet, H.: Recovering architectural assumptions. J. Syst. Softw. **79**(4), 552–573 (2006)
26. Salman, I., Misirli, A.T., Juristo, N.: Are students representatives of professionals in software engineering experiments? In: Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on. vol. 1, pp. 666–676 (2015)
27. SecureDrop: SecureDrop 0.3 Threat Model. https://docs.securedrop.org/en/stable/threat_model/threat_model.html
28. Shevchenko, N., Chick, T.A., O'Riordan, P., Scanlon, T.P., Woody, C.: Threat modeling: a summary of available methods. Technical report (2018)
29. Shostack, A.: Experiences threat modeling at microsoft. In: Modeling Security Workshop. Department of Computing, Lancaster University, UK (2008)
30. Shostack, A.: Threat Modeling: Designing for Security. Wiley, Crosspoint (2014)
31. Sion, L.: Automated threat analysis for security and privacy (2020), https://lirias.kuleuven.be/retrieve/589409/thesis.pdf
32. Sion, L., Van Landuyt, D., Yskout, K., Joosen, W.: SPARTA: Security & privacy architecture through risk-driven threat assessment. In: 2018 IEEE International Conference on Software Architecture Companion (ICSA-C). pp. 89–92. IEEE (2018)
33. Sion, L., Yskout, K., Van Landuyt, D., Joosen, W.: Solution-aware data flow diagrams for security threat modelling. SAC2018: SA-TTA track (2018)

34. Soudani, N., Raggad, B.G., Zouari, B.: A formal design of secure information systems by using a formal secure Data Flow Diagram (FSDFD). Risks and Security of Internet and Systems (CRiSIS), 2009 Fourth International Conference on pp. 131–134 (2009). https://doi.org/10.1109/CRISIS.2009.5411965
35. Steingruebl, A., Peterson, G.: Software assumptions lead to preventable errors. Secur. Priv. IEEE **7**(4) (2009)
36. Tao, Y., Kung, C.: Formal definition and verification of data flow diagrams. J. Syst. Softw. **16**(1), 29–36 (1991). https://doi.org/10.1016/0164-1212(91)90029-6
37. Torr, P.: Demystifying the threat-modeling process. IEEE Secur. Priv. **3**(5), 66–70 (2005)
38. Tuma, K., Scandariato, R.: Two architectural threat analysis techniques compared. In: European Conference on Software Architecture. pp. 347–363. Springer (2018)
39. Tuma, K., Scandariato, R., Widman, M., Sandberg, C.: Towards security threats that matter. In: Computer Security, pp. 47–62. Springer (2017)
40. Turpe, S.: The trouble with security requirements. In: 2017 IEEE 25th International Requirements Engineering Conference (RE). pp. 122–133. IEEE (2017)
41. Van Lamsweerde, A.: Goal-oriented requirements enginering: a roundtrip from research to practice [enginering read engineering]. In: Proceedings. 12th IEEE International Requirements Engineering Conference, 2004. pp. 4–7. IEEE (2004)
42. Van Landuyt, D., Joosen, W.: Supporting materials consisting of raw data sets and data sets after pre-processing, R scripts used for assessment, and the calculated scores (CSV) per assumption for each metric
43. Van Landuyt, D., Joosen, W.: Modularizing early architectural assumptions in scenario-based requirements. In: FASE, pp. 170–184. Springer (2014)
44. Van Landuyt, D., Joosen, W.: A descriptive study of assumptions made in LINDDUN privacy threat elicitation. ACM (2019). https://people.cs.kuleuven.be/dimitri.vanlanduyt/dvanlanduyt_sosym20_r2_supporting_materials.zip
45. Viega, J.: Building security requirements with CLASP. In: ACM SIGSOFT Software Engineering Notes. vol. 30, pp. 1–7. ACM (2005)
46. Williams, L., McGraw, G., Migues, S.: Engineering security vulnerability prevention, detection, and response. IEEE Softw. **35**(5), 76–80 (2018)
47. Wuyts, K., Van Landuyt, D., Hovsepyan, A., Joosen, W.: Effective and efficient privacy threat modeling through domain refinements. In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing. pp. 1175–1178. ACM (2018)
48. Xiong, W., Lagerström, R.: Threat modeling: a systematic literature review. Comput. Secur. **84**, 53–69 (2019)
49. Yang, C., Liang, P., Avgeriou, P.: A survey on software architectural assumptions. J. Syst. Softw. **113**, 362–380 (2016)
50. Yang, C., Liang, P., Avgeriou, P.: Assumptions and their management in software development: a systematic mapping study. Inf. Softw. Technol. **94**, 82–110 (2017)
51. Yang, C., Liang, P., Avgeriou, P., Eliasson, U., Heldal, R., Pelliccione, P., Bi, T.: An industrial case study on an architectural assumption documentation framework. J. Syst. Softw. **134**, 190–210 (2017)

Dimitri Van Landuyt is a research manager at iMinds-DistriNet, the Distributed Systems research group of the Department of Computer Science of KU Leuven, Belgium. Having obtained a PhD in Computer Science in 2011, his current research efforts are diverse yet focus on the primary application of core software engineering principles and techniques to contemporary software development, with topics ranging from security and privacy by design, to cloud computing, decentralized cloud storage systems, to customization of Software-as-a-Service applications.

Wouter Joosen is full professor in distributed software systems at the Department of Computer Science of KU Leuven, Belgium. He obtained a PhD degree from KU Leuven in 1996. He has also co-founded spin-off companies of KU Leuven: Luciad, a company specializing in software components for Geographical
Information Systems, and Ubizen (now part of Verizon Business Solutions), where he has been the CTO from 1996 till 2000, and COO from 2000 till 2002. His research interests are in cloud computing, focusing on software architecture and middleware, and in security aspects of software, including security in component frameworks and security architectures.