Requirements Eng
DOI 10.1007/s00766-013-0195-2

ORIGINAL ARTICLE

A descriptive study of Microsoft's threat modeling technique

Riccardo Scandariato · Kim Wuyts · Wouter Joosen

Received: 28 January 2013 / Accepted: 21 November 2013

Abstract Microsoft's STRIDE is a popular threat modeling technique commonly used to discover the security weaknesses of a software system. In turn, discovered weaknesses are a major driver for incepting security requirements. Despite its successful adoption, to date no empirical study has been carried out to quantify the cost and effectiveness of STRIDE. The contribution of this paper is the evaluation of STRIDE via a descriptive study that involved 57 students in their last master year in computer science. The study addresses three research questions. First, it assesses how many valid threats per hour are produced on average. Second, it evaluates the correctness of the analysis results by looking at the average number of false positives, i.e., the incorrect threats. Finally, it determines the completeness of the analysis results by looking at the average number of false negatives, i.e., the overlooked threats.

Keywords Secure software · Empirical study · Threat modeling · STRIDE · Anti-requirements

1 Introduction

Threat modeling is recognized as one of the most important activities in software security [26]. A threat modeling technique guides the security analyst to the discovery of the actions that a malicious agent (insider or outsider) might perform in order to misuse a software system. Threats are often referred to as anti-requirements and are an important driver for the definition of the security requirements of a system [31, 42, 47].

Threat modeling is based on the identification of the system's valuable assets, such as sensitive information or the availability of certain processing facilities. Therefore, threat modeling can be applied at several levels of abstraction, depending on the type of assets considered. From a requirements engineering perspective, the assets become more tangible when some design decisions are made and an initial decomposition of the system functionality is chosen. Therefore, early threat elicitation is best performed as soon as an initial architectural model of the system is available. For instance, a solution strategy that opts for data centralization is affected by different security issues than a system where the data is fully distributed. Moreover, threats depend on the security assumptions underlying a given decomposition, as the use of public networks to interconnect the subsystems. These assumptions are only precisely understood when the high-level decomposition of the system becomes more concrete.

Threat modeling can be used to analyze the soundness of (initial) software architectures and to spot flaws early on. The discovered flaws represent an opportunity to elaborate upon the security requirements of the system and, consequently, revisit the design choices or refine the architectural model. A popular technique for threat modeling is Microsoft's STRIDE [46], which is routinely used on all of Microsoft's products [13]. It is endorsed by the most recognized secure software processes, such as Touchpoints [26], OWASP's CLASP [4] and Microsoft's SDL [19]. It is also taught in certification programs such as CSSLP (http://www.isc2.org) and used in the industry [20].

STRIDE is also the subject of ongoing research. For instance, the technique is being augmented by OWASP in order to become applicable to the domain of mobile

R. Scandariato (✉) · K. Wuyts · W. Joosen
iMinds-DistriNet, KU Leuven, 3001 Leuven, Belgium
e-mail: riccardo.scandariato@cs.kuleuven.be

Published online: 03 December 2013

applications [34]. Further, it has been extended for the purpose of privacy analysis [6]. Finally, researchers are adapting STRIDE to different types of system models [21, 38].

Despite its successful adoption, to date no empirical study has been carried out to quantify the costs and effectiveness of STRIDE. Presently, the productivity of an analyst using STRIDE is unknown, as is the average percentage of erroneous threats produced, or the share of threats that are overlooked. Without such a knowledge, it is difficult to estimate how much time has to be budgeted for the security analysis of a given software system. Similarly, it is hard to understand whether the results of the analysis are trustworthy.

The contribution of this paper is a *descriptive study* evaluating STRIDE by means of quantitative observations that have been performed in controlled, laboratory conditions in the context of a university course. The study has involved 57 students in their last year of the master in computer science. Contrary to controlled experiments, in a descriptive study a phenomenon is characterized and no attempt is made to analyze the effects of variables on the phenomenon itself. This type of study is instrumental in order to understand a technique and eventually formulate research hypotheses to be further investigated by means of comparative experiments. Incisively, Grimes and Schulz portray descriptive studies as "the first scientific toe in the water in new areas of inquiry" [10]. As STRIDE has never been studied before, the authors are indeed venturing in a new area of inquiry, and at this stage, a descriptive study appears to be the most appropriate means of investigation.

As remarked by Tichy [45], exploratory studies like ours are particularly suited for the enrollment of university students. Further, students are ideal for investigating the issues related to a technology's learning curve, as remarked by Carver et al. [3]. The authors have paid particular attention to the advice offered by Carver et al. in order to successfully embed a study in the context of a university course. In particular, the study has been harmonized with respect to the teaching goals of the course, and the students have been given adequate incentives. These are important safeguards in order to obtain realistic results. The participants of the study have been asked to perform the threat analysis of a medium-sized distributed system. The main goal of the study was to evaluate STRIDE by providing an answer to the following research questions:

- *RQ1: productivity* How many valid threats per hour are produced?
- *RQ2: correctness* Is the number of incorrect threats small?
- *RQ3: completeness* Is the number of overlooked threats small?

The study has been conducted in three subsequent installments of the above-mentioned university course. In the first 2 years, we observed the work of the participants while they were identifying the threats in an early phase of the software development life cycle, namely when the elaboration of the security requirements (and of the corresponding controls) had just begun. In the third year, we observed the participants in a scenario where the security engineering process had progressed further. They identified the threats for a system where some security controls have been put into place to satisfy the authorization and authentication requirements, which are fundamental for security. The variation in the third year is meant to provide evidence that the results obtained in the previous 2 years can be generalized and could apply to systems with different degrees of elaboration of security mechanisms.

In summary, this study concludes that STRIDE is not difficult to learn and execute, although it is relatively time-consuming. Further, many threats go undetected during the analysis.

The rest of this paper is organized as follows. Section 2 contextualizes this study in the domain of requirements engineering. Section 3 provides the necessary background information on STRIDE. Section 4 describes the planning of the study and states the test hypotheses. Section 5 describes the execution of the study and the measurement procedure. The results are presented in Sects. 6 and 7 and summarized in Sect. 8. Section 9 lists the threats to the validity of this study. Section 10 discusses the impact of STRIDE on security requirements elicitation. Finally, Sect. 11 discusses the related work, and Sect. 12 gives the concluding remarks.

2 Requirements, architecture and threat analysis

As formulated by Haley et al. [11], the essence of *security goals* is to protect assets from harm. At the beginning of the requirements engineering process, these assets are high level and often abstract, for instance, information about customers' credit cards. Harm is caused by the violation of the security concerns, such as confidentiality, integrity, availability and accountability. An example of a confidentiality concern is that the credit card information is shared with the seller only. In this example, a threat could be that the credit card information is stolen by a third-party and the corresponding security goal would be that the IT system of the seller should protect the credit card information from being stolen. The naïve example above illustrates how threat analysis is applied on high-level assets and yields to high-level security goals.

From these high-level protection goals, concrete *security requirements* are to be derived by operationalization.

As security requirements are constraints on the functional requirements, one has to determine which security goals "apply" to which functional requirements [11]. Next, one has to assess what weaknesses in the functionality should be avoided by means of the constraints. That is, the operationalization of a security goal requires that the concept of harm (i.e., the attacker's goal) is operationalized as well. Therefore, there is the need for a finer-grained round of *threat analysis*. This exercise requires the identification of how the abstract assets, such as information, translate to tangible entities in the software architecture, like data and software components, and how the functionality meshes with those entities by means of coordination and communication flows. This can be realistically carried out if a decomposition of the system has emerged and cues are available about the distribution of the responsibilities as well as the layout of the information flows. Only then, in fact, it is feasible to assess the potential weaknesses that might threaten the system. In short, some design decisions need to be taken at this stage, and an initial structure of the system needs to be shaped in order to enable the elicitation of realistic threats and, consequently, sound security requirements. We are not talking about a full-blown software architecture but rather about an initial step in the solution domain that sketches the overall organization of the system. This is a pivotal idea that shoulders this study and is substantiated by the Twin Peaks model of Nuseibeh [32] and its extension for security by Heyman et al. [14]. The Twin Peaks model has its origin in the requirements engineering community and describes how no sharp separation exists between the definition (and evolution) of requirements and architecture. The creation of these two artifacts progresses in parallel and is heavily intertwined. Not only do requirements provide the rationale for design decisions, but the latter are made while requirements are elaborated and, in turn, shape the way such elaboration proceeds. This idea is nowadays accepted by the majority of the research community, both requirements engineers and software architects [1].

To recap, the security requirements are derived by means of threat modeling, which requires the manifestation of an initial sketch of the solution, which, in turn, is either implicitly or explicitly available at that point of the requirements engineering process because of the effect of the Twin Peaks. Hence, this study deals with the threat modeling of an *initial architecture* like the one pictured in Fig. 2. However, the context of the threat modeling technique is the elicitation of the security requirements that will eventually drive the refinement of the initial design solution into a fully fledged software architecture. Incidentally, threat modeling could be used once again in order to assess the soundness of the resulting, more refined software architecture, although other techniques might also be preferred, for instance, ATAM [5].

3 Background on threat modeling with STRIDE

STRIDE is a model-based threat modeling technique developed by Microsoft [19]. The methodology guides the security analyst through several activities, which will be briefly discussed in the remainder of this section.

Step 1 Model the system by means of a data flow diagram (DFD). The initial activities define the scope of the analysis and produce a model of the system under review. The DFD is built during this step and is instrumental for the elicitation of the threats later on. An example of a DFD is given in Fig. 1, which is a representation of the system used in this study (see also Fig. 2). Starting from the context diagram showing the users and the third parties of the system, a more detailed decomposition, called level 1 DFD, is derived by refinement. The level 1 DFD shows the way the information travels in the system through data flows (DF) from external entities (EE) like system users to processing nodes (PN) like active software components and data storage points (DS) like database components. The hierarchical refinement can continue further down if necessary. As reported by Dhillon, this is only required in more complex systems and often a level 1 DFD is sufficient for the sake of the analysis [7].

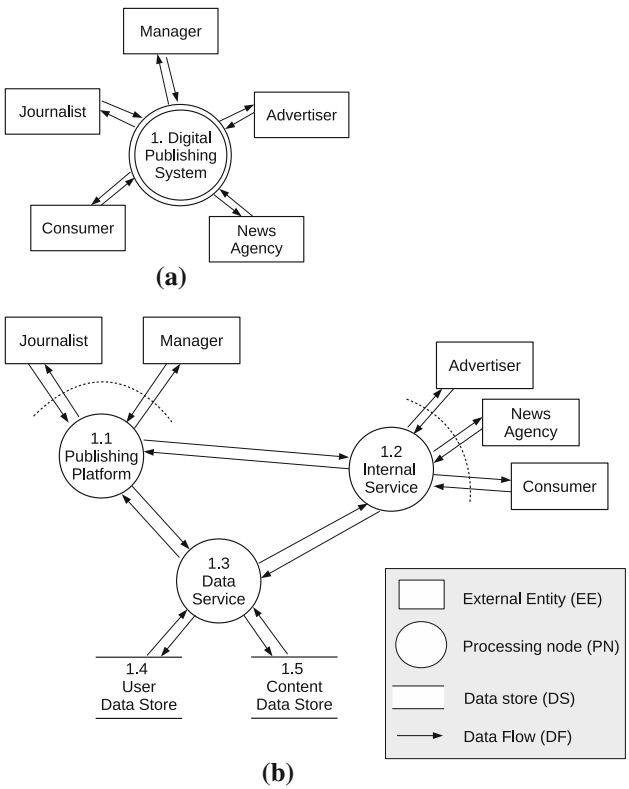

Fig. 1 A sample DFD for the digital publishing system. **a** Level 0 (context diagram). **b** Level 1

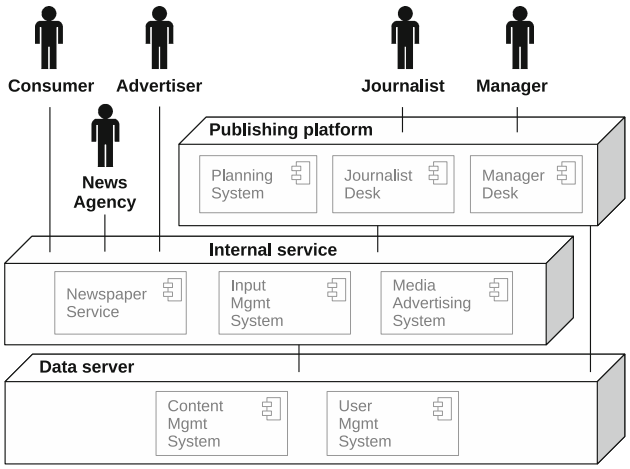

Fig. 2 The digital publishing system

Table 1 The mapping of DFD elements to threat categories

DFD elements	Applicable threat categories					
	S	T	R	I	D	E
EE	✔		✔			
DF		✔		✔	✔	
DS		✔		✔	✔	
PN	✔	✔	✔	✔	✔	✔

During the scoping and modeling step, the analyst must also list the security assumptions explicitly. Examples of assumptions are the presence of an authentication mechanism, or the possibility for a threat agent (a.k.a. attacker) to eavesdrop on the communication channels. The assumptions are used later on, during the threat elicitation step.

Step 2 Map the DFD elements to threat categories.

In STRIDE, threats are organized according to six categories:

- *Spoofing* (*S*) refers to a rogue person or program successfully impersonating another legitimate user or program.
- *Tampering* (*T*) refers to a threat agent illegitimately modifying application resources, such as in memory data.
- *Repudiation* (*R*) refers to a user (legitimate or malicious) able to deny the execution of an action within the system.
- *Information disclosure* (*I*) refers to a threat agent obtaining private information she is not supposed to access.
- *Denial of service* (*D*) refers to a threat agent making a system resource unavailable to its intended users.

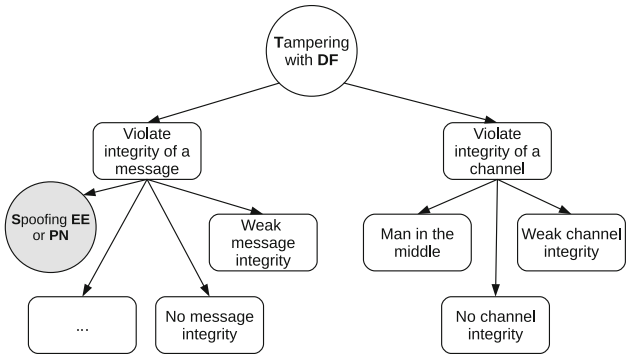

Fig. 3 A sample tree of template threats

- *Elevation of privilege* (*E*) refers to a threat agent obtaining privileged access to resources that are normally protected.

Each element type of the DFD is susceptible to one or more of the above categories, according to the schema illustrated in Table 1. For instance, the consumer (EE) in Fig. 1 is prone to spoofing and repudiation, while the internal service (PN) is susceptible to all types of threats.

Step 3 Elicit the threats. For each mapping between a generic threat category and a DFD element type, STRIDE provides a "checklist" of concrete threats that need to be considered. As shown in Fig. 3, the checklist comes in the shape of a tree containing a hierarchy of template threats that can be instantiated in the context of the system under review. In the reference book describing STRIDE [19], a catalog of 12 threat trees is provided. A tree can sometimes refer to others. In the picture, for instance, the "spoofing" tree is referenced from the "tampering with a data flow" tree. The tree-based structure is intended to ease the navigation and to provide a better overview of the rationale behind each threat.

The applicability of a threat depends on the assumptions stated in step 1. With reference to Fig. 3, for instance, the threat concerning a man-in-the-middle is relevant only if channel-level security (e.g., SSL) is used and should be discarded otherwise.

Step 4 Document the threats STRIDE does not mandate a specific format for this step. Misuse cases are commonly used in security to document threats [42]. Therefore, we have used the structured textual format that has been defined for misuse cases by Sindre and Opdahl [43]. The template is very similar to those used for use cases and includes some extra security-specific information such as the so-called capture points section. This section contains the description of how the misuse may be prevented (or detected) and represents an opportunity to identify the security requirements.

3.1 Tool support

When the study started, Microsoft was providing a prototype tool. The prototype allowed the user to draw the DFD and generated a very long list of threats (organized by the six threat categories) that the user had to prune in light of the assumptions made for the system under analysis.

Meanwhile, the prototype has been replaced by the SDL threat modeling tool, also released by Microsoft [41]. The tool still provides a graphical DFD editor but does not generate the list of threats anymore. It only generates the mappings between DFD elements and threat categories (step 2).

To avoid additional confounding factors, this work did not employ any tool. Given the evolution of the tool support, this choice proved to be successful, as our results would have been specific to the use of a prototype not available anymore.

4 Planning the study

This section presents the design of an empirical study that observed and evaluated how the participants apply the STRIDE technique to perform the security analysis of a distributed system. In order to enable replication of this study, all the experimental material mentioned here is also available online at [37].

As mentioned earlier, this study has been executed over three consecutive installments. The core part of the study has taken place during the first 2 years, when the participants (41 in total) have been asked to analyze a distributed system that is in the early phase of specification and that does not contain security mechanisms yet. In the third year, the participants (16 in total) have analyzed a similar system, which, however, contains security controls for authorization and authentication. This change has been introduced to investigate the potential for generalization of the results obtained during the previous 2 years. That is, if the results are confirmed in the third year, their applicability could go beyond the specific type of system used in the experimentation. For instance, the results could apply to systems with different degrees of elaboration of the security mechanisms.

In this and the following sections, we start describing the planning, execution and results of the core part of the study, i.e., the first 2 years. The description of the third year is deferred to Sect. 7.

4.1 Experimental object

The digital publishing system is a medium-scale distributed system developed in the context of an industry-oriented research project carried out in partnership with major news publishers in European countries [24]. The complete description of the system is available in a technical report [48]. The main stakeholder of the system is a news publisher (like a company publishing newspapers) transitioning to Internet-based distribution channels. The main purpose of the system is (1) to support the journalists and the editors during the creation and the updating of a digital edition and (2) to provide subscribed consumers online access to the editions, e.g., via e-paper terminals. A major source of revenue is represented by the ads inserted by advertisers into each edition. In order to be competitive, the system has to be flexible with respect to the needs of the advertisers. Therefore, the system integrates the advertisers into its own core business processes and provides means to personalize the ads to the consumer's interests.

A pictorial representation of the system is given in Fig. 2. At the bottom of the figure, the Data Server node is the storage for the published edition and for the raw material used in the creation process (see the content management system), like news articles created by the desk journalists and the field reporters, as well as the ads provided by the advertisers. This node also contains information concerning the subscribed consumers (see the user management system). The internal service node is the hub of the system and provides a front-end to the consumers via the newspaper service, to the stakeholders involved in the production of the editions via the input management system and to the advertisers via the media advertising system. The core business of producing the editions is orchestrated by the publishing platform node (top part of the figure), which provides access to both journalists and managers.

We have provided the participants with sufficient documentation for understanding the system. The system documentation is a document of about 50 pages containing (1) the description of the business goals behind the system, (2) the domain model describing the main concepts, (3) the description of the stakeholders and actors (i.e., users) of the system, (4) the functional requirements as a list of 30 use cases and (5) the UML diagrams of the software architecture. The documentation is available online [37].

4.2 Participants

The participants of this study are the students of a course on "design and analysis of software systems" taught by the experimenters, which is positioned in the first semester of the second year of the master in computer science at KU Leuven in Belgium.

In the first 2 years, we have observed 41 students organized in ten teams of about four students. The students have been allowed to assemble the teams themselves, which is the state of practice in most project-oriented

Table 2 Teams and selected subsets of the system

Team	Size	Subset
1	4	Journalist and manager
2	4	Journalist and manager
3	4	Whole system
4	4	Whole system
5	3	Consumer and advertiser
6	4	Consumer and advertiser
7	4	Whole system
8	5	Journalist and manager
9	5	Consumer and advertiser
10	4	Journalist and manager

courses at our university. The team sizes are given in the second column of Table 2. In the first year, we had 23 students divided into six teams. In the second year, we had 18 students divided into four teams.

The course is optional in the curriculum, and hence, the students tend to be very motivated, as they selected the course because of a specific interest in the subject. This is also indicated by the average grade of 14 out of 20 points obtained by the teams that participated in this study in the first 2 years.

Concerning the background of the participants, they can be regarded as a rather homogeneous group. Although we have not administered a specific questionnaire to validate this assumption, we know from the demographics that the participants have the same age, nationality and study background. Regarding the latter, the course that hosted the study is available for selection to one master program only. This program requires the students to acquire 120 credits in order to graduate, and they are free to choose specializing courses for a maximum of 19 credits only, which is a negligible variation. Prior to the enrollment in the study, the students have followed a course on software requirements (focusing on domain models and use cases) and a prerequisite course on the design of software architectures. Also, the students have followed an introductory course on system and network security. Some have followed a course on software security.

4.3 Task

The participants have been requested to perform the STRIDE analysis of the digital publishing system according to the four steps described in Sect. 3. They had to use the catalog of threat trees contained in [19]. They had to consider both insider threats (e.g., from legitimate users) and outsider threats (e.g., from a competitor). As part of the process of documenting the threats as misuse cases, the participants have been asked to define some "capture points," i.e., security requirements that are meant to prevent (or monitor) the occurrence of a threat. The identification of these requirements was instrumental for a later stage of the course. As this study focuses on STRIDE and the identification of anti-requirements, the analysis of the security requirements is not in scope. However, a description of the identified security requirements is given in Sect. 10.

The digital publishing system is used throughout the course, also outside the scope of the study reported here. Therefore, its size is intentionally meant to be challenging for the students. However, the size of the STRIDE analysis grows rapidly with the size of the system under analysis. Therefore, we advised the participants to select a smaller subset of the whole system to focus on. As a guideline, they have been told to take the viewpoint of a group of system users and pick a coherent subset containing a sufficient number of key use cases that are relevant to that group. The individual choices of the teams are reported in the third column of Table 2 and fall into one of the following options:

- *Consumer and advertiser* This is a slice of the system containing the use cases that are relevant to the external users.
- *Journalist and manager* This is the slice representing the perspective of the internal users.
- *Whole system* Some participants opted for a broader analysis.

As the system under analysis in the first 2 years does not contain security controls, the participants had to keep in mind the following standard *assumptions*: (a) There are no authentication and authorization mechanisms in place, (b) no logging is implemented, and (c) the communication links are unencrypted. These assumptions are important to judge whether a potential threat is indeed applicable to the system under analysis. For instance, all threats related to weaknesses in the crypto-protocols become irrelevant under the above assumptions and must be discarded.

The participants were allowed to make additional assumptions, provided that these were properly documented. It happened only in two cases. One team assumed that the storage capacity of the data store is unlimited, which further reduces the number of applicable threats (e.g., for denial of service). Another team assumed that the data storage infrastructure was shared with other systems and hence more threats had to be considered (for information disclosure). As risk assessment is not a part of the course, the teams have not been judged on the nature of their assumptions as long as they could properly justify their choices by showing that they were reasonable.

Concerning the creation of the level 1 data flow diagram, during the STRIDE lecture, the participants have been taught

to use the deployment diagram (exemplified in Fig. 2) as a guideline. This diagram is central as the distribution of the system over several networked nodes is a key aspect from a security perspective. The participants have been taught to map nodes of the deployment diagram to processing nodes (PN) in the DFD and actors to external entities (EE). According to their choice, the teams marked the obtained DFD with a closed line identifying the slice of the system they intended to analyze. Some teams made a distinction between production data (news and ads) and consumer information. Consequently, they included two different data stores in the DFD (as in Fig. 1).

4.4 Design of the study

The teams carried out the same task (described above) on the same experimental object (the digital publishing system).

4.5 Hypotheses

Research question 1 Concerning the *productivity*, we do not formulate a test hypothesis as we have no a priory expectations about what the outcome should look like. We define the productivity as the number of correct results (i.e., the true positives) per unit of time. Note that we do not include incorrect results in the definition intentionally.

As mentioned in Table 3, we define a *true positive* (TP) as a discovered threat that meets the following three criteria:

a. It is relevant because it can be clearly related to a leaf of a threat tree,
b. It is applicable in light of the security assumptions, and
c. Its description, as given by the team, is realistic from a security perspective.

In summary, the true positives represent the amount of work (effort) correctly carried out by a team.

To further characterize the STRIDE technique, we also measure its difficulty as it is perceived by the participants. This has been done by means of a questionnaire.

Research question 2 Concerning the *correctness*, as alternative hypothesis we are interested in knowing whether, on average, the number of correct results is predominant (more than 80 %) w.r.t. the overall number of results produced (both correct and incorrect). As mentioned before, the true positives represent the correct results. Incorrect results are the *false positives* (FP), i.e., the threats that are reported by the team and do not meet the criteria listed above. Consequently, we defined the following null hypothesis:

$$H_0^P : \mu\{P \triangleq \mathrm{TP}/(\mathrm{TP} + \mathrm{FP})\} \leq 0.80$$

The quantity used in the null hypothesis is commonly known as *precision* in information retrieval terminology. Although the choice of a 80 % threshold is somewhat arbitrary, this is often regarded as a good reference for precision, e.g., in the information retrieval community.

Research question 3 Concerning *completeness*, as alternative hypothesis we were interested in knowing whether, on average, the number of correct results covers enough (more than 80 %) of the space of actual threats (both correct and overlooked). Overlooked threats are the *false negatives* (FN), i.e., the number of threats that the team failed to identify although they are relevant in the system under analysis. Therefore, we defined the following null hypothesis:

$$H_0^R : \mu\{R \triangleq \mathrm{TP} \,/\, (\mathrm{TP} + \mathrm{FN})\} \leq 0.80$$

The quantity used in the null hypothesis corresponds to the so-called *recall*. The 80 % threshold is chosen according to the same rationale as before.

Table 3 Terminology

Term	Meaning
True positive (TP)	Correct threat
False negative (FN)	Overlooked threat
False positive (FP)	Incorrect threat
Precision (P)	Percentage of the produced threats that are correct (the higher the better)
Recall (R)	Percentage of the existing threats that are discovered (the higher the better)

5 Operation of the study

5.1 Preparation of the participants

The participants received their training as part of the course. The course syllabus, material and structure did not change over the years. Furthermore, the course was taught by the same lecturers and teaching assistants.

The course focuses on the analysis of software architectures. It is organized into three parts: (a) specification of the system's security anti-requirements as a set of misuse cases and identification of the corresponding security requirements, (b) system refinement by selecting architecture-level security solutions and (c) validation of the overall system vis-a-vis a larger set of quality requirements by means of trade-off analysis. Each part comprises a set of theoretical lectures followed by a practical laboratory session where the students work in teams.

The study is located in the context of the first part of the course, which comprises three lectures of 2.5 h and one laboratory session of 4 h.

At the beginning of the three lectures, the students get a brief recap about use cases and software architecture.

Afterward, the lectures move on to the definition of security requirements and the security analysis of a software system. The STRIDE methodology is covered in depth and extensively illustrated by means of a complete and realistic example from the e-health application domain. In the lectures, the experimental object used in the laboratory session is also introduced. The class material is available online [37]. The amount of training is higher than the standard at Microsoft, where it is reported that "most practitioners have at most 2 h of threat modeling training" [40].

The students are very motivated to execute the projects duly, as their final grade is heavily based on these (about 50 %). The amount of work required to complete the project is intentionally more than what the teams can achieve during the laboratory time. Indeed, the assigned task is of a rather realistic size and certainly not a "toy example". Therefore, the students are expected to get started during the four laboratory hours and then complete the assignment (and compile the report) as homework. As the teams also work outside the supervised laboratory hours, we have asked them to track the time they spend on the project. The authors are confident that the teams reported the time earnestly, which is confirmed by the congruence between the declared time and the work described in their reports.

5.2 Execution of the study

The teams started working on the task during the first laboratory of the course and then had about ten days to complete the task at home and to turn in their report by a given deadline.

The laboratory hours have been supervised by a teaching assistant and a lecturer. During the laboratory hours, each team had access to an Internet-connected personal computer with a graphical modeling tool to draw the data flow diagrams and an office suite for editing the misuse cases. However, the teams were allowed to work on paper if they preferred. Indeed, the teams mostly worked on paper during the brainstorming in the laboratory and produced a digital report later on.

As mentioned earlier, the digital publishing system has been previously introduced during the lectures. The detailed documentation has been made available a few days before the laboratory session, so that the participants could further familiarize themselves with the experimental object. The participants have been advised to download and read the documentation before coming to the laboratory. The assignment, containing the detailed description of the task was made available at the beginning of the laboratory session. It is now available online at [37].

Each report contained a picture representing the analyzed level 1 DFD (and the slice they selected), a list containing the assumptions made (with comments about the rationale), a table for the mappings between DFD elements and threat categories, and the misuse cases (organized by threat category) documenting the identified threats.

The teams also had to turn in a sheet reporting the effort time (in h) spent on each step of the task. The time sheets have been kept on a teamwise basis, i.e., each group reported the person hours jointly spent by the team on each step.

Further, at the end of the study, all participants had to fill in a short questionnaire containing four questions. Each question asks to rate the difficulty of one of the four steps mentioned in Sect. 3, on a scale from 1 (very easy) to 5 (very hard). The questionnaire does not identify the participant except for her team membership.

5.3 Measurement procedure

The effort time is collected from the time sheets, which have been informally checked (sometimes also during the examination) for congruence with respect to the amount of work visible in the reports. From the questionnaires, we collected the opinion of the participants about the perceived difficulty of the activities.

The reports turned in by the teams have been assessed by two security experts (the first and the second author) independently. First, the experts met to decide which threats (of the catalog) were applicable given the nature of the system under analysis and the assumptions mentioned in Sect. 4.3. Then they processed the reports. In particular, the experts counted the number of correct threats (true positives) and incorrect threats (false positives) according to the definitions given in Sect. 4.5. The experts compared their results, and in case of mismatches, they discussed until a consensus was reached. This happened in a very small number of cases (<4 % of the 260 threats reviewed).

The experts also measured the size of the DFD used by each team by looking at the corresponding picture included in the report. They counted the number of instances in each element type (process nodes, data flows and so on).

Given the set of threats in the catalog (as decided earlier) and the DFD, the experts computed the total number of applicable threats per each team. This number (called the *baseline*) represents an idealistic result that could be obtained by exhaustively analyzing each element in the DFD. In this sense, it represents an upper bound. However, in order to avoid a repetitive listing, all teams leveraged a technique called *reduction*, which is suggested in the documentation of STRIDE:

> You can apply a process called reduction to reduce the number of entities you will analyze. In short, if you have two or more DFD elements of the same type […] you can model the elements as one entity […]. In other words, when you analyze the threats to one of the elements, that same analysis applies to the other element also [19].

For instance, if a team mentioned that a certain data flow was subject to an eavesdropping threat, the same threat was not repeated anymore for the rest of the data flows. In a nutshell, via reduction the teams set out to exemplify all the different threats that were applicable in the system under analysis, rather than providing an exhaustive list. This attitude has been endorsed by the instructors of the course, and it is also accounted for in the measurement procedure.

In fact, in light of reduction, the baseline is not a faithful representative of the size of the task that the teams set out to accomplish. Consequently, it cannot be used as a yardstick to measure the overlooked threats. Therefore, the experts counted the false negatives as the number of applicable leaf nodes in the catalog of threat trees that have not been exemplified in each report.[1] This is more in line with the way the teams executed the task. As before, the counting of the false negatives has been done independently by the experts and the few issues resolved by consensus.

6 Results

In this section, we first present some descriptive statistics and then address the three research questions. All measurements used in this paper are available online at [37].

6.1 Descriptive statistics

6.1.1 DFD

The DFD produced by the teams was correct. Hence, no influence in the results can be traced back to any erroneous or incomplete DFD modeling. The size of the DFD used by each team is presented in Table 4. The total number of DFD elements is given, together with a more detailed breakdown of the number of processes (PN), data stores (DS), external entities (EE) and data flows (DF). It can be noted that the teams can be divided into two groups: DFDs of about 24 elements (models for the whole system) and of about 10 elements (other subsets). The average size of the DFDs is 14.3 elements (standard deviation is 6.90), and the 95 % confidence interval is [10, 18.5] elements (one-sample Wilcoxon test). Note that this DFD size falls within the range of real-world systems analyzed by practitioners as reported by Dhillon [7].

Table 4 Size of the DFD

Team	PN	DS	EE	DF	Total
1	2	1	1	7	11
2	2	1	1	6	10
3	2	2	5	17	26
4	2	1	5	15	23
5	1	1	2	6	10
6	2	1	1	6	10
7	3	1	5	14	23
8	3	1	1	8	13
9	1	1	1	4	7
10	2	1	1	6	10
μ	2.0	1.1	2.3	8.9	14.3
σ	0.67	0.32	1.89	4.61	6.90

The data flows are the biggest contributors to the overall DFD size (see the means in Table 4). Indeed, there is a strong correlation between the size of the DFD and the number of data flows (Spearman $\rho = 0.997$, $p < 0.05$).

6.1.2 Baseline

We recall that the baseline represents the upper bound to the number of threats that can be discovered. The average size of the baseline is 214 threats (standard deviation is 92), which is extremely large and possibly unmanageable. The confidence interval is [152.5, 274.5] threats (one-sample Wilcoxon test).

Figure 4 shows the composition of the baseline over the six threat categories. The tampering (*T*), information disclosure (*I*) and denial of service (*D*) threats are, on average, much higher than the others, and these differences are statistically significant (paired-samples Wilcoxon test). Further, the analysis of the correlation (Spearman, $p < 0.05$) between the overall count of threats in the baseline and its individual components reveals that the strongest values are obtained for *T* ($\rho = 0.988$), *I* (1.0) and *D* (0.986). This is in line with the fact that the T–I–D categories are mapped more often to the DFD elements than the other categories, as visible in Table 1. More mappings mean more checklists to apply. Additionally, the checklists for these categories contain more threats.

In conclusion, it appears that STRIDE has a tendency to produce a potentially overwhelming number of threats (unless reduction is used) and to deepen the analysis in three threat categories.

[1] Of course, the extra assumptions made by the teams have been considered.

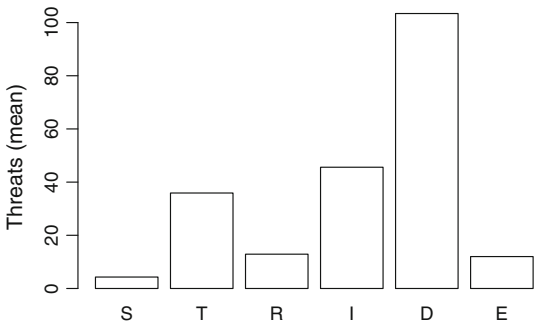

Fig. 4 Size of the baseline by threat category

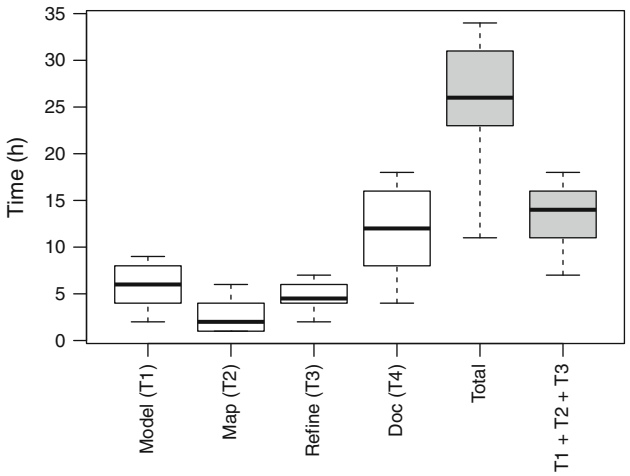

Fig. 5 *Boxplot* of time per STRIDE step

6.2 Research question 1: productivity and difficulty

6.2.1 Effort time

The average time spent by the teams on the task is 25.1 h (standard deviation is 6.6), and the confidence interval is [19, 29] hours (one-sample Wilcoxon test).

In Fig. 5, the time is broken down into different steps of STRIDE mentioned in Sect. 3: (1) creating the DFD, (2) mapping the DFD elements to the threat categories, (3) eliciting the threats and (4) documenting the identified threats.

It is interesting to note that, on average, the same amount of time is spent on finding the threats (steps 1–2–3 together) and documenting them as misuse cases (paired-samples Wilcoxon test).

It is not possible to link the effort time to the size of the DFD and of the baseline. Therefore, there is no proportion between the time put into the exercise and the real size of the chosen subset. This possibly relates to the way of working of the participants, which used the reduction technique, as explained in Sect. 5.3.

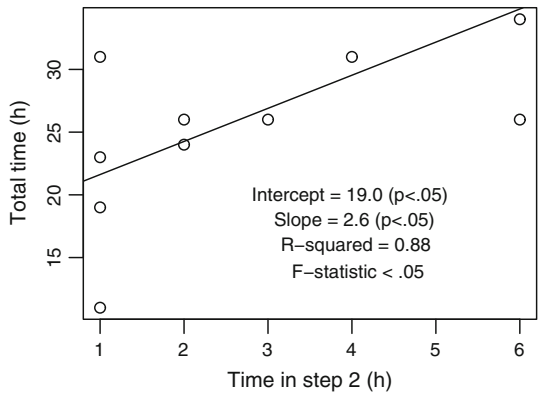

Fig. 6 Prediction model for the overall time

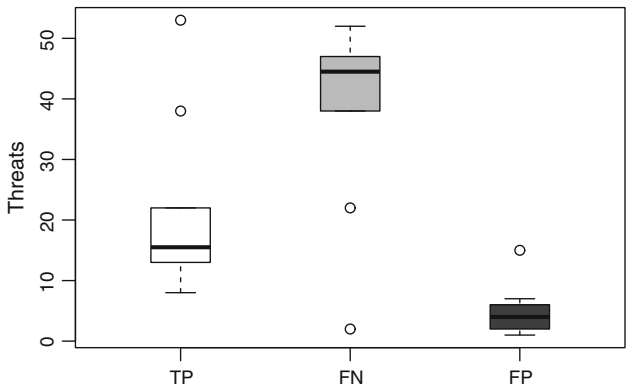

Fig. 7 *Boxplot* of true positives, false negatives and false positives

As shown in Fig. 6, an accurate model can be built ($R^2 = 0.88$, F statistic = <0.05) to predict the overall time spent on analyzing the system (steps 1–4 together) from the observation of the preliminary, smaller work done in the mapping step (step 2). The model has been built by means of the LTS robust regression technique. Replica studies should consider investigating this point further, e.g., by means of specific questions to the participants.

6.2.2 Effort

In theory, the effort (work done) would be the sum of both the true positives and the false positives. However, we want to assess the productivity with respect to the work *correctly* done. Therefore, we count only the true positives in the measure of the effort.

Figure 7 (white box) presents the boxplot for the TPs. Table 5 reports the raw numbers used to build Fig. 7. The average number of TPs is 21.1 threats (standard deviation of 13.8), and the confidence interval is [14.0, 64.5] threats (one-sample Wilcoxon test). Figure 8 presents the TPs arranged by threat categories (white bars). The statistical analysis (paired-samples Wilcoxon test) reveals that there

Table 5 Threats by team: true positives, false negatives and false positives

Team	TP	FN	FP
1	18	44	3
2	13	47	2
3	15	45	4
4	22	38	6
5	15	45	2
6	38	22	5
7	13	48	4
8	16	44	1
9	53	2	15
10	8	52	7
μ	21.1	38.7	4.9
σ	13.8	15.3	4.0
CI	[14.0, 64.5]	[23.5, 48]	[2.5, 8.5]

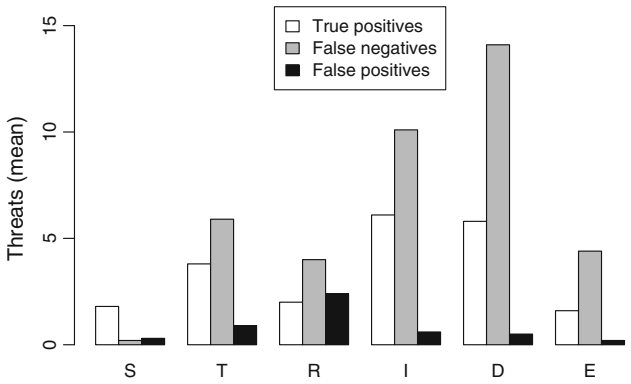

Fig. 8 True positives, false negatives and false positives by threat category

are no differences within the categories in the T–I–D group as well as within the S–R–E group. This can be appreciated also visually by looking at the white bars in Fig. 8. Further, there is a statistically significant location shift between the average number of TPs of any of the T–I–D categories and any of the S–R–E categories. Therefore, there is more focus on T-I-D threats and less on S–R–E threats. This is in line with the findings for the baseline.

The TPs are not correlated with the size of the DFD (nor of the baseline). More interestingly, the TPs cannot be linked to the time. This may imply that, for our sample, investing more time on the analysis does not guarantee an improvement.

6.2.3 Productivity

Having characterized the effort time and the true positives, we can eventually provide an answer to the first research question. Considering only the time spent in identifying (and not documenting) the threats, an average productivity of 1.8 threats/h (standard deviation of 1.5) is to be expected for a system similar to the analyzed one. The confidence interval is [0.94, 3.25] threats/h (one-sample Wilcoxon test). That is, it takes an average of 33 min to identify a correct threat.

Considering the overall time (i.e., including the time spent documenting the threats as misuse cases), the average productivity drops to 0.9 threats/h with a confidence interval of [0.48, 1.33].

Disregarding the time spent building the DFD (which is a one-time, upfront exercise), and including the time spent documenting the threats, the average productivity is of 1.2 threats/h with a confidence interval of [0.64, 1.81].

Considering the baselines as an upper bound, at this rate and with a single analyst, it would take up to seven working days to complete the analysis of the smallest DFD in our study, and up to 26 working days for the largest. These values are obtained by multiplying the average productivity by the size of the baselines.

6.2.4 Perceived difficulty

The perceived difficulty is obtained by analyzing the answers to the questionnaire, which contains one question per step of the STRIDE methodology. The respondent is asked to rate the difficulty of each step on an ordinal scale from one (very easy) to five (very difficult). Table 6 provides the number of respondents, the medians of the answers and the confidence intervals (Wilcoxon).

On average, the first two steps of the methodology are perceived as "easy" (2) by the participants, while the last two are perceived as being of "normal" (3) difficulty. The location shift between the first two steps and the last two is statistically significant (paired-samples Wilcoxon test, significance level of 0.0125 with the Bonferroni correction). Overall, it is quite remarkable that participants that have no prior practical experience with applying the STRIDE methodology find that it is not hard to work with it. This confidence testifies in favor of the very structured

Table 6 Reported difficulty per step

Step	Answers	Median difficulty	CI
1. Model the system and list the assumptions	39	2	[2, 2.5]
2. Map DFD elements to threat categories	39	2	[1.5, 2]
3. Elicit threats via threat tree patterns	39	3	[2.5, 3]
4. Document the threats as misuse cases	40	3	[3, 3.5]

approach adopted by STRIDE and the good level of guidance provided by the existing documentation. In summary, the participants are not overwhelmed by the material, following the steps is not difficult, and the threat trees simplify the life of the novice analysts. Possibly, this is a common trait of checklist-based threat modeling approaches. However, more experimentation is needed to support this point.

On the other hand, the questionnaire suggests that the participants do not have a correct perception of the large amount of threats that, on average, they let go missing (FPs). This impression has been confirmed during the discussion of the results with the students at the final examinations. However, a replica of this study should consider asking more direct questions on this matter, e.g., by letting the participants rate their perceived level of task completion. Nevertheless, the results of this study suggest that the training should particularly stress the issue of the false negatives (overlooked threats) in order to avoid overconfidence in the analysis.

We also computed the teamwise median difficulty per step and investigated the relationships with the other measures. We found a significantly strong correlation (Spearman $\rho = 0.81$, $p < 0.05$) between the median difficulty in step 3 (threats elicitation) and the number of correct threats (TP) elicited. This could mean that a greater awareness of the real difficulty of the task leads to better results.

6.3 Research question 2: correctness of the analysis results

As shown in Fig. 7 (dark gray box), we observed many more correct results (TP) than incorrect results (FP) and the location shift is statistically significant (paired-samples Wilcoxon test, significance level of 0.025 with the Bonferroni correction). The average for FPs is 4.9 threats, and the standard deviation is 4.0. The confidence interval is [2.5, 8.5] threats (one-sample Wilcoxon test).

With reference to the null hypothesis H_0^P, the average *precision* is 0.81 with a standard deviation of 0.11. This value is slightly better than the threshold used in the hypothesis. However, the one-tailed Wilcoxon test asserts that the null hypothesis cannot be rejected ($p > 0.05$). Therefore, from a statistical perspective, we conclude that the number of incorrect results is not small with respect to the results produced. Nonetheless, we remark that the statistical test would have succeeded with a slightly lower threshold of 75 %. In summary, the amount of incorrect results is not particularly concerning.

Considering Fig. 8 (dark gray bars), we can observe that there are less FPs than TPs in all categories, except for repudiation (*R*). In this latter case, there is no statistically significant difference, and hence, we can conclude that in the *R* category there are, on average, as many correct results as incorrect threats. Further, it also appears that on average there are more errors in the *R* category, while the other categories are about the same. This is confirmed by the statistical test in most cases (paired-samples Wilcoxon test). Possibly, our study suggests that the training of STRIDE analysts should pay particular attention to creating awareness about the repudiation threats.

We have analyzed the false positives in the reports, in search for reoccurring mistakes. Our analysis focused on the repudiation (*R*) category, which totals the largest amount of FPs. The most common mistakes are due to the teams reporting unrealistic threats that are not grounded in the reality of the system under analysis and whose feasibility is not demonstrated by the course of action described in the threat scenarios. In fact, often these erroneous threats are too vague, and their scenario description is not specific enough in order to assess its correctness according to the criteria listed in Sect. 4.5. We believe this is a symptom of the lack of understanding of this threat category on the participants' side. Many other errors were due to the failing at determining that certain items in the threat catalog were not applicable to the system under analysis (e.g., the threats related to logging). This was caused by a wrong interpretation of the security assumptions.

6.4 Research question 3: completeness of the analysis results

The investigation of the false negatives brings out a different story. As shown in Fig. 7 (light gray box), we observed many more overlooked threats (FN) than correct results (TP). However, the difference is not statistically significant (paired-samples Wilcoxon test, significance level of 0.025 with the Bonferroni correction).

Significance is not achieved due to the presence of outliers, which are symbolized by small circles in the figure. In fact, if the outlier with the smallest value is removed (team 9, 2 false negatives in total), the difference becomes statistically significant. The average number of FNs is 38.7 threats, and the standard deviation is 15.3. The confidence interval is [23.5, 48] threats (one-sample Wilcoxon test).

With reference to the null hypothesis H_0^R, the average *recall* is 0.36 with a standard deviation of 0.25. This value is uncomfortably low when compared to the expectation of 80 %. The one-tailed Wilcoxon test confirms that the null hypothesis cannot be rejected ($p > 0.05$) and therefore the number of correct results does not cover adequately the space of actual threats, as the number of overlooked threats is too high.

With reference to Fig. 8 (light gray bars), we can observe that the false negatives in the repudiation (*R*) and elevation of privilege (*E*) are about the same. All the other averages are different from each other, as confirmed by the statistical test (paired-samples Wilcoxon test). The denial of service category (*D*) presents the highest number of overlooked threats, which are almost three times as many as the correct results in that category. Possibly, this suggests an attention point for the training of STRIDE analysts.

We have looked into the reports of the teams in order to identify patterns in the false negatives for two categories: information disclosure (*I*) and denial of service (*D*). These categories account for the largest number of false negatives. We have not seen any recurring threat that is forgotten by the participants. Rather, we got the impression that the teams were overwhelmed by the sheer amount of threats that are applicable in these categories. The threat trees are indeed very reach, with as many as 16 leaf nodes for information disclosure and 20 for denial of service, including the references to other trees.

7 Toward a generalization of the results

The results presented so far are obtained for a software system that includes virtually no security-specific mechanisms, which could represent, for instance, an early stage of the development where the security requirements have not been elaborated upon yet. With the aim of generalizing the results, we replicated the study for a third consecutive year. This time, the digital publishing system has been extended by the teams to include security mechanisms for authentication and authorization. Subsequently, the teams have applied the STRIDE technique as described before. This case corresponds, for instance, to a more advanced stage of the development life cycle where some important security requirements have already been elicited and consequently implemented. If confirmed, the results could then apply to systems with different degrees of elaboration of their security concerns.

We observed 16 participants organized in four teams of four students each. Clearly, we have no ambition to draw statistically significant conclusions given the limited sample size. As in the previous years, the participants have shown a good level of motivation and commitment as corroborated by the average final grade of 15.75 out of 20 obtained by the teams.

Concerning the planning and operation of the third year of the study, the same procedures described in Sects. 4 and 5 have been applied. This also includes the measurement protocol described in Sect. 5.3. However, the reports have been reviewed by only one of the two experts in the third year.

7.1 Observations

On average, the teams modeled the system with a much larger DFD compared with the previous years (47.25 elements vs. 14.3). Also, the median of the reported difficulty of building the DFD (step 1) is higher ("hard" vs. "easy"), and the difference is statistically significant. The median difficulty for step 2 (mapping) stays "easy". Conversely, the median difficulty for both the threat elicitation and documentation steps appears to be easier ("easy" vs. "normal"). The participants spent on average 73.75 h on the task (+194 %), and about two thirds of the time is spent in identifying threats (before, there was a balance between identifying and documenting threats).

We report that the productivity went down to 0.5 threats/h (from 1.8), although the difference is not statistically significant. The precision went slightly down (from 0.81 to 0.76) and recall deteriorated as well (from 0.36 to 0.31).

In summary, the replica of the study on a system having more security mechanisms confirmed the results obtained in the previous years as far as the research questions 2 (correctness) and 3 (completeness) are concerned, although the higher complexity of the analyzed system might have caused a lower productivity (research question 1).

8 Summary of the findings

In summary, this study observed that in our sample:

1. The STRIDE technique is not perceived as difficult, but with an average productivity of 1.8 threats/h at best, the time cost is relatively large.
2. The average number of incorrect threats is low and corresponds to the 19–24 % of the total amount of produced threats.
3. The average number of overlooked threats is very high and corresponds to the 64–69 % of the total amount of threats.

Despite the limitations of this study discussed in Sect. 9, these observations might be of use to software managers (to budget the effort for a security analysis), to customers of security consultants (to understand the level of trust that can be granted to the results) and to instructors (to understand the learning curve and the attention points of the training).

9 Threats to validity

Concerning the *conclusion validity*, the time each team spent on the task is reported by the participants themselves. To prevent this threat, we kept reminding the participants

to track the time they were spending on the assignment. We do not deem this threat harmful to the validity of the study. In fact, the time sheets appeared congruent with the work documented by the students in their project reports.

An additional threat is represented by the mistakes that the experts might have done during the assessments of the reports. This threat has been mitigated in the first 2 years of the study by employing two experts. The authors are confident that no major errors have been done in this respect.

Concerning the *internal validity*, maturation of the participants has to be highlighted. Possibly, the participants could have become tired because of the size of the task. Another threat to the internal validity is the participants' increasing understanding of the digital publishing system. The experimenters tried to minimize this learning effect by introducing the system during the lectures and by providing the documentation of the system before the laboratory started. Also, the participants might have shared their work between teams, also from 1 year to another. However, we did not detect any case of plagiarism, and using the same experimental object across the years was central to this study, as the sample size is not sufficient to investigate multiple factors.

Finally, three threats to the *external validity* need to be mentioned. The main issue threatening the generalization of the results concerns the use of master students instead of professionals. The matter of using students in empirical studies is still controversial. However, some studies have shown that students perform comparably to professional in certain tasks such as requirements selection [44] or lead time estimation [18]. Some studies have also observed that master students (like in our study) perform more realistically than freshmen [36] and that students working in a project (like in our study) are to be preferred over students working in a classroom [2]. In any case, the use of students is advised in exploratory studies like ours [3, 45].

Second, we used teams of three to five students. The results might not generalize to the case of a single analyst. However, we remark that letting many analysts soundboard during the threat modeling of a system is commonly advised by the state of the art [39]. Third, the results might be influenced by the experimental object used in this study. For instance, the results might not apply to other application domains or to systems of different size and complexity.

10 Discussion

As mentioned in Sect. 7, in the third-year installment of the study, the participants went through two phases. First, they identified the authentication and authorization requirements of the system. This activity has been carried out using the framework of Haley et al. [11], which is referred in Sect. 2. These requirements have been used to extend the system with basic security controls. In a second phase, the extended system has been analyzed by means of STRIDE in order to identify the threats, which have been documented as misuse cases. As mentioned in Sect. 3, the documentation of the misuse cases contains a section (called "capture points") that the teams have been instructed to use as a means to describe the security requirements that could prevent the corresponding threats.

Therefore, the teams engaged in the identification of security requirements both *before* and *after* the STRIDE analysis had been performed. This provides the opportunity to appreciate whether the use of STRIDE influenced the way the participants addressed the requirements engineering activity. Clearly, the participants had more knowledge about the system during the second round of requirements identification, which could have influenced the results as well. Note that the requirements identified before the STRIDE analysis are more limited in scope, as the participants looked into authorization and authentication only. To compensate for this, we restricted the comparison with the requirements that are clearly related to either authentication or authorization concerns. Given the small number of teams observed and the loose control of the factors, we can only draw some preliminary observations and do not imply that our intuitions have a general applicability. Nevertheless, the observations in this section are an interesting starting point and call for further investigation.

10.1 Security requirements identified before STRIDE

First and foremost, all teams neglected the authentication requirements in spite of being explicitly requested to deal with such concern. The teams simply postulated that some form of authentication was in place, without going deeper in that respect. The security requirements they produced predicate over roles like customers and advertisers, which correspond to the actors mentioned in the documentation (e.g., the use cases). However, how the individuals should be dependably mapped to the system roles is never discussed. For instance, none of the teams has defined what an "authenticated customer" is. Therefore, a number of issues related to identity management have been overlooked, including the life cycle of users' credentials. In summary, the teams have been very superficial for what concerns the authentication requirements.

Concerning authorization, the teams identified the security constraints as role-based access control rules with respect to the system assets. They focused mostly on information assets, like a news story or a newspaper edition, and much less on resources, like services and service requests, which can also be the cause of security issues.

Further, the access control rules often referred to generic operations such as create, request and update. Incidentally, the deletion of information was never present in the authorization requirements, although this is a relevant concern in the system at hand.

Furthermore, when system functionality was mentioned, the teams manifested a very idealistic attitude. They did not consider the possibility that the functionality might be exploited to circumvent the authorization constraints, for instance, because of a faulty implementation. Therefore, the teams did not foresee any "backup plan" for the case of a failing authorization requirement.

10.2 Security requirements identified after STRIDE

Concerning authentication, the teams adopted a multilateral perspective and often considered the problem of authenticating the system to the customers, e.g., to avoid scams. Previously, the teams had only considered the concern of authenticating the users to the system. The teams have also been more thorough in their requirements. They included constraints about the authentication procedures, e.g., requiring the use of hard-to-guess credentials. They systematically thought about protecting the transmission of credentials-related material by means of secure channels. They stated time constraints over the authentication sessions (so that sessions time out eventually) in order to limit the risk of identity impersonation. They even came up with the proposal of monitoring the login requests in order to identify any suspicious activity, like an attacker trying to guess a user's credential.

Concerning the authorization requirements, we observed a more balanced mix of assets, which included information as well as services and service requests. The teams also systematically applied the principle of using multiple layers of security. For instance, they required the presence of either auditing, to deter the circumvention of the authorization constraints, or input validation, to prevent the misuse of faulty functionality. Similarly, some teams required the use of encryption to protect the stored data, in order to protect the information assets from the prying eyes of an insider that bypasses the authorization constraints (e.g., by means of a side channel).

In conclusion, the teams identified more accurate security requirements after having performed the STRIDE analysis. However, this benefit came at the cost of a perspective that is too low level at times. For instance, sometimes the requirements refer to data stores rather than information. This appears as a bias due to the use of data flow diagrams in STRIDE. Similarly, some requirements are borderline to being design decisions. For instance, the use of input validation could be considered a rather solution-oriented technique.

11 Related work

11.1 Evaluation of STRIDE

McGraw has performed a study of the software security initiatives at 67 well-known companies [27]. The study resulted in the creation of the so-called BSIMM model of the most important security activities that are enforced by the software industry to date. In the author's words, the BSIMM model is a "descriptive model of software security." With reference to thread modeling, it has been observed that 35 firms perform a risk-driven architectural design review. Although the study does not comment on STRIDE directly, it provides material evidence about the relevance of this technique.

Microsoft has never published any figure related to the effectiveness of STRIDE, although the technique is often accounted for improving the quality of it products. Shostack, head of the SDL threat modeling unit, has published an experience report explaining the evolution of the technique, discussing some issues encountered over the years and highlighting the directions of future improvement [40].

Dhillon has criticized STRIDE openly and compared it to an alternative technique that has been developed in-house [7]. The main remark is about the fact that STRIDE is time-consuming because it proceeds by analyzing the DFD element-by-element. His remarks are in line with the relatively low productivity observed in this study. The author describes an alternative technique that also leverages DFD models. The DFD is annotated with extra information, like the programming language used to develop the components or the data flow type (HTTP, SQL and so on). This information is used to identify specific patterns of interaction in the DFD that might lead to weaknesses. Such identification is driven by a library of 35 DFD patterns. For instance, a pattern might be a process node that provides a Web interface. In this case, the cross-site scripting and the cross-site request forgery threats must be considered. Citing 30 industrial projects, the author reports that this technique is as effective as STRIDE concerning the identification of the threats and is more time-efficient. However, no supporting evidence is provided as the comparison is only anecdotal.

11.2 Evaluation of other threat modeling techniques

Although no empirical evaluation of STRIDE has been performed so far, some competing threat and risk modeling techniques have been put to the test of experiments.

Misuse cases are a well-known threat modeling technique that is based on the description of the system functionality via use cases and involves the brainstorming of security experts [42]. Differently from STRIDE, the analysis process does not adopt checklists of potential threats.

Via creative thinking, the experts analyze each functionality in the use case diagram and identify ways of harming the system by means of misfunctionality.

Threat modeling via misuse cases has been tested in a series of three experiments. Opdahl and Sindre compared misuse cases to attack trees, which represent another well-known technique also involving out-of-the-box thinking and based on brainstorming by security experts [33]. The experiment involved 63 students, and the results show that attack trees might lead to a higher number of identified threats. However, the study did not assess the correctness and relevance of the produced threats. The participants did not show any difference with respect to the preference of misuse cases over attack trees. Karpati et al. compared misuse cases to malactivity diagrams, a technique to model social engineering threats in business processes [23]. The study involved 75 students and found no significant differences between the two techniques, except for a preference of misactivity diagrams with respect to the ease of use. Karpati et al. compared misuse cases to misuse case maps, a technique that extends the notation of misuse cases and relates them to the architectural structure of the system [22]. The study involved 33 students and found that there is no difference in the amount of identified threats. Further misuse case maps have been perceived as more difficult to learn by the participants. However, misuse case maps helped identifying significantly more mitigations.

Diallo et al. [8] assessed common criteria, misuse cases and attack trees by means of a comparative evaluation. The authors applied the three techniques to the same object and performed an evaluation according to the following dimensions: ease of learning, usability and quality of the results. Despite the limitations of this type of setup, the authors observed that common criteria are hard to learn and use. Both misuse cases and attack tree are easier in this respect, with an advantage of attack trees when it comes to interpreting and analyzing the results.

Meland et al. [28] reported that, in their own experience, threat modeling based on misuse cases is a time-consuming and complex task. The authors suggested an improvement of the technique based on the use of libraries of reusable threat models. In practice, they proposed to structure the analysis process by adopting checklists. They set up an experiment with seven professionals to test the efficacy of two alternative libraries. One is based on the reuse of complete misuse case diagrams. The other library is based on the reuse of misuse case stubs, which are organized according to the STRIDE categories. The reader will notice that the latter case bears some similarities to the threat tree patterns used in our study. The authors observed no difference in either the amount or the type of threats discovered with the two alternatives. Also, the participants were not particularly opinioned about preferring one alternative over the other.

Hogganvik et al. [16, 17] have investigated the role of graphical models by means of a series user studies. The objective was to define an optimal representation for the risk diagrams that are produced by the CORAS risk and threat analysis technique. In particular, they focused on the comprehensibility aspects and demonstrated that the CORAS notation provides advantages over UML, because it uses more intuitive icons and is augmented with text labels.

11.3 Security requirements

Threat modeling plays an important role in the elicitation of security requirements. Therefore, the technique evaluated in this study is complementary to several other security requirements engineering methods. The most relevant ones are briefly described in this section. For a detailed comparison of these and other approaches, we refer the reader to the survey of Fabian et al. [9] and the systematic literature review of Mellado et al. [29].

KAOS is a goal-oriented methodology that has been extended to the field of security [47]. Threats are modeled as antigoals that are pursued by the attackers. High-level antigoals are obtained by negating the (security) goals of the goal model. Similar to obstacle analysis, concrete threats are obtained by refining antigoals. However, in the case of security, the domain knowledge is extended with information related to the attacker's capabilities.

In the domain of goal-oriented requirements engineering, the concept of threat is also central in the i* family of security extensions (SI* [25], Secure Tropos [30], STS [35]). For instance, in STS-ml (the i*-alike modeling language of STS), security requirements are modeled as constraints on the relationships between the actors involved in a goal model. The language supports the representation of a rich set of security requirements including confidentiality, integrity, non-repudiation, trust and so on. Threats are represented as risks associated with goals that cannot be reached or resources that become unavailable.

Problem frames have also been adapted to the domain of security [12]. In particular, the methodology provides a catalog of patterns called security problem frames (SPF) representing recurring configurations of the problem domain where security requirements are modeled. A SPF can be refined by taking into consideration security mechanisms, such as cryptographic schemes or access control systems. To this aim, several concretized security problem frames (CSPF) are provided by the methodology.

12 Conclusion

This paper presented an evaluation of the STRIDE threat modeling technique by means of a study that spanned 3 years.

As summarized in Sect. 8, the paper has drawn conclusions on the process-oriented aspects related to applying the technique. The paper has also characterized the quality of the analysis results in terms of correctness and completeness.

The field of empirical secure software engineering is still in its infancy, and more work is needed to objectively quantify the cost and performance of many techniques in the field of software security. This paper contributes to giving some momentum to the field. However, the authors want to stress that the challenge is also of a methodological nature. It is often hard to define a measure for the security properties. Hence, it is difficult to assess whether the results of a technique are "good." We have assessed the work of the teams by means of false positives and false negatives, which is a commonplace way of reporting results in security. To this aim, we started from an operational definition of true positives, false positives and false negatives, which is both crisp and easy to apply. These definitions enable the objective and repeatable assessment of the artifacts, e.g., the analysis results in our case.

This paper also generates a number of interesting research questions that could be the subject of future experimental work. For instance, the false negatives (incorrect results) could be caused by the misinterpretation of some template threats in the checklists. Testing the comparative efficacy of alternative threat trees might unearth the root cause of the errors made by the participants of this study. Similarly, the cause of the high number of false negatives (overlooked threats) is particularly worthy of investigating. The outcome of such studies would lead to improvements in the field of threat modeling that are grounded in actual fact. Ultimately, this would contribute to the long-term mission of a sounder discipline of secure software engineering.

Acknowledgments This research is partially funded by the Research Fund KU Leuven, and by the EU FP7 project NESSoS, with financial support from the Prevention of and Fight against Crime Programme of the European Union (B-CCENTRE).

References

1. Avgeriou P, Grundy J, Hall J, Lago, P, Mistrik I (eds) (2011) Relating software requirements and architectures. Springer, Berlin
2. Berander P (2004) Using students as subjects in requirements prioritization. In: International symposium on empirical software engineering (ISESE)
3. Carver J, Jaccheri L, Morasca S (2010) A checklist for integrating student empirical studies with research and teaching goals. Empir Softw Eng 15(1):35–59
4. Chandra P, Wohleber T, Feragamo J, Williams J (2007) CLASP v1.2: comprehensive, lightweight application security process. Tech. rep., OWASP
5. Clements P, Kazman R, Klein M (2001) Evaluating software architectures: methods and case studies. Addison-Wesley, Reading
6. Deng M, Wuyts K, Scandariato R, Preneel B, Joosen W (2011) A privacy threat analysis framework. Requir Eng 16(1):3–32
7. Dhillon D (2011) Developer-driven threat modeling: lessons learned in the trenches. IEEE Secur Priv 9(4):41–47
8. Diallo M, Romero-Mariona J, Sim SE, Alspaugh T, Richardson D (2006) A comparative evaluation of three approaches to specifying security requirements. In: Working conference on requirements engineering: foundation for software quality (REFSQ)
9. Fabian B, Gürses S, Heisel M, Santen T, Schmidt H (2010) A comparison of security requirements engineering methods. Requir Eng 15(1):7–40
10. Grimes D, Schulz K (2002) Descriptive studies: what they can and cannot do. Lancet 359:145–149
11. Haley C, Laney R, Moffett J, Nuseibeh B (2008) Security requirements engineering: a framework for representation and analysis. IEEE Trans Softw Eng 34(1):133–153
12. Hatebur D, Heisel M, Schmidt H (2007) A pattern system for security requirements engineering. In: International conference on availability, reliability and security (ARES)
13. Hernan S, Lambert S, Ostwald T, Shostack A (2006) Uncover security design flaws using the STRIDE approach. MSDN Mag. http://msdn.microsoft.com/en-us/magazine/cc163519.aspx
14. Heyman T, Yskout K, Scandariato R, Schmidt H, Yu Y (2011) The security twin peaks. In: International symposium on engineering secure software and systems (ESSoS)
15. Hogganvik I, Stølen K (2005) On the comprehension of security risk scenarios. In: International workshop on program comprehension (IWPC)
16. Hogganvik I, Stølen K (2006) A graphical approach to risk identification motivated by empirical investigations. In: International conference on model driven engineering languages and systems (MoDELS)
17. Hogganvik I, Lund M, Stølen K (2009) Reducing the effort to comprehend risk models: textlabels are often preferred over graphical means. Risk Anal 51(5):916–932
18. Höst M, Regnell B, Wohlin C (2000) Using students as subjects—a comparative study of students and professionals in lead-time impact assessment. J Empir Softw Eng 5(3):201–214
19. Howard M, Lipner S (2006) The security development lifecycle. Microsoft Press, Redmond
20. Ingalsbe J, Kunimatsu L, Baeten T, Mead N (2008) Threat modeling: diving into the deep end. IEEE Softw 25(1):28–34
21. Johnstone M (2010) Threat modelling with STRIDE and UML. In: Australian information security management conference
22. Karpati P, Opdahl A, Sindre G (2011) Experimental comparison of misuse case maps with misuse cases and system architecture diagrams for eliciting security vulnerabilities and mitigations. In: International conference on availability, reliability and security (ARES)
23. Karpati P, Sindre G, Matulevicius R (2012) Comparing misuse case and mal-activity diagrams for modelling social engineering attacks. Int J Secur Softw Eng 3(2):54–73
24. KU Leuven DigiNews project. http://goo.gl/M6xkF
25. Massacci F, Mylopoulos J, Zannone N (2010) Security requirements engineering: the SI* modeling language and the secure tropos methodology. In: Ras ZW, Tsay LS (eds) Advances in intelligent information systems. Springer, New York, pp 147–174
26. McGraw G (2006) Software security: building security in. Addison-Wesley, Reading
27. McGraw G, Migues S, West J (2013) Building security in maturity model (BSIMM-V). Tech. rep., Cigital

28. Meland P, Tøndel I, Jensen J (2010) Idea: reusability of threat models—two approaches with an experimental evaluation. In: Engineering secure software and systems (ESSoS)
29. Mellado D, Blanco C, Sanchez LE, Fernandez-Medina E (2010) A systematic review of security requirements engineering. Comput Stand Interface 32(4):153–165
30. Mouratidis H, Giorgini P (2007) Secure Tropos: a security-oriented extension of the tropos methodology. Int J Softw Eng Knowl Eng 17(2):285–309
31. Myagmar S, Lee A, Yurcik W (2005) Threat modeling as a basis for security requirements. In: Symposium on requirements engineering for information security (SREIS)
32. Nuseibeh B (2001) Weaving together requirements and architectures. IEEE Comput 34(3):115–119
33. Opdahl AL, Sindre G (2009) Experimental comparison of attack trees and misuse cases for security threat identification. Inf Softw Technol 51(5):916–932
34. OWASP Mobile security project: mobile threat model. https://www.owasp.org/index.php/OWASP_Mobile_Security_Project
35. Paja E, Dalpiaz F, Poggianella M, Roberti P, Giorgini P (2012) STS-Tool: socio-technical security requirements through social commitments. In: International conference on requirements engineering (RE)
36. Runeson P (2003) Using students as experiment subjects—an analysis on graduate and freshmen student data. In: International conference on empirical assessment in software engineering (EASE)
37. Scandariato R, Wuyts K, Joosen W Experimental material. https://sites.google.com/site/descriptivestudy/
38. Schaad A, Borozdin M (2012) TAM2: automated threat analysis. In: Annual ACM symposium on applied computing (SAC)
39. Schneier B (1999) Attack trees. Dr. Dobb's J 24(12):21–29
40. Shostack A (2008) Experiences threat modeling at Microsoft. In: Workshop on modeling security (ModSec)
41. Shostack A (2009) Getting started with the SDL threat modeling tool. MSDN Mag. http://msdn.microsoft.com/en-us/magazine/dd347831.aspx
42. Sindre G, Opdahl AL (2005) Eliciting security requirements with misuse cases. Requir Eng 10(1):34–44
43. Sindre G, Opdahl, AL (2002) Templates for misuse case description. In: Workshop on requirements engineering: foundations for software quality (REFSQ)
44. Svahnberg M, Aurum A, Wohlin C (2008) Using students as subjects—an empirical evaluation. In: International symposium on empirical software engineering and measurement (ESEM)
45. Tichy W (2000) Hints for reviewing empirical work in software engineering. Empir Softw Eng 5(4):309–312
46. Torr P (2005) Demystifying the threat-modeling process. IEEE Secur Priv 3(5):66–70
47. Van Lamsweerde A (2004) Elaborating security requirements by construction of intentional anti-models. In: International conference on software engineering (ICSE)
48. Van Landuyt D, Gregoire J, Michiels S, Truyen E, Joosen W (2006) Architectural design of a digital publishing system. Tech. rep., KU Leuven