The Journal of Systems & Software 144 (2018) 275–294

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Threat analysis of software systems: A systematic literature review

K. Tuma*, G. Calikli, R. Scandariato

Department of Computer Science and Engineering, University of Gothenburg and Chalmers University of Technology, Vasaparken Gothenburg, Sweden

A R T I C L E I N F O

Keywords:
Threat analysis (modeling)
Risk assessment
Security-by-design
Software systems
Systematic literature review (SLR)

A B S T R A C T

Architectural threat analysis has become an important cornerstone for organizations concerned with developing secure software. Due to the large number of existing techniques it is becoming more challenging for practitioners to select an appropriate threat analysis technique. Therefore, we conducted a systematic literature review (SLR) of the existing techniques for threat analysis. In our study we compare 26 methodologies for what concerns their applicability, characteristics of the required input for analysis, characteristics of analysis procedure, characteristics of analysis outcomes and ease of adoption. We also provide insight into the obstacles for adopting the existing approaches and discuss the current state of their adoption in software engineering trends (e.g. Agile, DevOps, etc.). As a summary of our findings we have observed that: the analysis procedure is not precisely defined, there is a lack of quality assurance of analysis outcomes and tool support and validation are limited.

1. Introduction

After decades of research the issue of integrating security early-on in the Software Development Life-cycle (SDL) has received more attention and is becoming a corner stone in software development. In this respect, architectural threat analysis plays a major role in holistically addressing security issues in software development. Threat analysis includes activities which help to identify, analyze and prioritize potential security and privacy threats to a software system and the information it handles. A threat analysis technique consists of a systematic analysis of the attacker's profile, vis-a-vis the assets of value to the organization. Such activities often take place in the design phase and are repeated later on during the product life-cycle, if necessary. The main purpose for performing threat analysis is to identify and mitigate potential risks by means of eliciting or refining security requirements. Threat analysis is particularly important, since many security vulnerabilities are caused due to architectural design flaws. A failure to consider security early-on can be a cause for so-called Architectural Technical Debt (ATD) (Li et al., 2015). Furthermore, fixing such vulnerabilities after implementation is very costly and requires work arounds which sometimes increase the attack surface.

Building Security In Maturity Model (BSIMM)[1] collects statistics from 95 companies and gauge their level of adoption with respect to several secure software development practices. According to this technical report (BSIMM, 2017), security-specific code analysis techniques have successfully found their way into the industrial practice, as two thirds of the surveyed companies routinely adopt them. However, it is a bit discouraging to find that only one third of the companies adopt architectural threat analysis. One possible explanation for that is the lack of automation support of threat analysis, since available tools require extensive human interaction for efficient use Almorsy et al. (2013); Berger et al. (2016) and Whittle et al. (2008). Another possible explanation is the lack of an industry-standard technique for threat analysis. In comparison with safety analysis techniques (failure analysis), threat analysis have yet to mature in this area (Raspotnig and Opdahl, 2013). This paper attempts to understand the potential road blocks to a wider adoption of threat analysis techniques by systematically studying the existing methods.

Recently a limited and compendiary review of threat analysis techniques has emerged (Cheung, 2016) in a form of a short technical report, yet this review only describes a handful of approaches. To the best of our knowledge, this is the first systematic and complete review of the state of the art. We have analyzed 38 primary studies for a total of 26 threat analysis techniques. With this study we aim at providing information to the practitioners about the extent the existing threats analysis techniques are applicable to their needs. Providing such information to practitioners might facilitate active usage of the aforementioned techniques and in the long-term cause techniques to mature. The contributions of this work are threefold:

(i) we systematically analyze the existing literature and identify gaps for future research,
(ii) we provide insight into the obstacles for adopting the existing approaches in practice and how these obstacles could be overcome,

* Corresponding author.
E-mail addresses: katja.tuma@cse.gu.se (K. Tuma), gul.calikli@cse.gu.se (G. Calikli), riccardo.scandariato@cse.gu.se (R. Scandariato).

[1] https://www.bsimm.com.

https://doi.org/10.1016/j.jss.2018.06.073
Received 5 February 2018; Received in revised form 7 May 2018; Accepted 22 June 2018
Available online 23 June 2018
0164-1212/

(iii) we provide insight into the adoption of the threat analysis techniques in software engineering trends (i.e. DevOps, Agile development, IoT and automotive).

The rest of the paper is organized as follows. Section 2 describes the research methodology, including the research questions and data extraction strategy. Section 3 presents the results, while Section 4 discusses them. The threats to validity are listed in Section 5. Section 6 discusses the related work and Section 7 presents the concluding remarks.

2. Research methodology

We conducted our research by adopting the systematic literature review method. By following the steps introduced by Kitchenham et al. (2007), we collected and analyzed the literature. According to the guidelines, our study consisted of three main steps: planning, conducting and documenting the review. The SLR was motivated by the need to strengthen security engineering practices in the SDL, desired both by academia and industry. We searched for similar studies in the ACM, IEEE, Google Scholar and Scopus digital libraries (November 2016), to establish whether an SLR about threat analysis techniques was previously conducted. None of the mentioned digital libraries contained an SLR about threat analysis techniques, reaching the same goals and objectives.

2.1. Research questions

The initial goal of this study is to catalog and characterize the existing threat analysis techniques. Thereafter, the second goal of our work is to provide future directions and to address how the techniques can be used by practitioners including their adoption to the latest software engineering trends. To this end, a critical analysis of the selected literature was performed answering three main research questions, which are reflected in the assessment criteria, presented in Tables 2–4.

RQ1: What are the main characteristics of the identified techniques? We have organized the first research question into four inquiries (refer to Table 2 for more details).

2.1.1. Applicability (RQ1.1)

What level of abstraction is the threat analysis technique applicable to? Threat analysis can be conducted on projects, where little is known about the actual system in the early design stages. However, systems are sometimes also analyzed for threats later-on in the SDL. For instance, integration of new units in a code-base may require a threat analysis of the effected components. Therefore, such an analysis might be performed on a low-level of abstraction (e.g. static code analysis).

2.1.2. Input (RQ1.2)

What information do the identified techniques require as input? This question refers to the information about the system that is required in order to execute the analysis. In particular, it aims at identifying the *type* and the *representation* of required information for executing the analysis. This information helps researchers and practitioners to determine which approaches can be adopted according to the available software artefacts.

2.1.3. Procedure (RQ1.30)

What kind of activities are part of the analysis procedure of the identified techniques? This research question aims at determining how the input is transformed to obtain the desired outcomes of the analysis. Most threat analysis techniques, such as STRIDE Shostack (2014); Torr (2005), CLASP (Owasp, 2018), OCTAVE (Caralli, 2007), etc. require expert knowledge for execution. However, some methods are supported by catalogs of security threats, which aid the identification of threats by providing contextual examples. This study considers such techniques as knowledge-based. Furthermore, we observe the level of precision of threat analysis procedures. A higher precision may increase the quality of the analysis and provide opportunities for security compliance. Commonly, the technique documentation includes descriptive guidelines for analysis execution, yet no clear definition is given for when the procedure ends. As part of this research question we also investigate how the proposed techniques determine when the analysis should stop (i.e. Definition of Done).

Finally, we also observe which security concerns are accounted for and to what extent is risk assessment present in the analysis procedure.

2.1.4. Outcomes (RQ1.4)

What information is gained by the outcomes of the identified techniques? This question intends to qualify the added value of adopting a technique. The main purpose for investigating the outcomes is to indicate what kind of results can be expected from the studied approaches. Among others, we assess the granularity of outcomes as well as the available quality assurance of outcomes.

RQ2: What is the ease of adoption of the identified techniques? Our second research question is motivated from a more practical perspective (see Table 3). It aims to determine the challenges of adopting the studied approaches in practice. This work refers to *ease of adoption* as a broader term compared to “usability”. First, tool availability is a strong indicator of technique maturity. Unfortunately, fully stand alone tools are less common compared to prototype tools or tool extensions. Second, practitioners benefit from a complementary guidance for execution. The guidance could provide fine- or coarse-grained instructions for using the proposed tool in combination with the theoretical concepts of the approach. Third, tools are typically accompanied by tool documentation. We also investigate whether there are other sources of technique documentation available (e.g. demonstrations). Finally, the ease of adopting the studied approaches is also dependent on the required knowledge and skill set of the analyst. For instance, approaches that require extensive education in formal methods will be difficult to use for software engineers without additional training. Likewise, manual approaches typically require domain knowledge and knowledge about security attacks and countermeasures. To this aim, the second research question aims to determine the target audience of the proposed approaches.

RQ3: What evidence exists that threat analysis techniques work in practice? The purpose of this question is to identify the extent of validation conducted for a technique. In addition to previously mentioned characteristics, providing evidence about a realistic application of an approach is very important to practitioners as well as academics. In the scientific community, validation has to be extensive and reproducible. Unfortunately validation is sometimes under-prioritized (as summarized in Table 4). First, this research question aims to determine the type of validation method used to evaluate the proposed approaches (e.g. case studies). Second, we aim to determine who performed the validation (e.g. a third party). The reader should note that we do not attempt to undermine the validation efforts contributed by the authors of the techniques. Third, this research question aims to identify the domain of validation (e.g. automotive, web based systems, etc). Validation across different domains further enables the generalizability of results. In general, extensive validation includes different validation methods across domains, preferably also performed by validators with no conflict of interest.

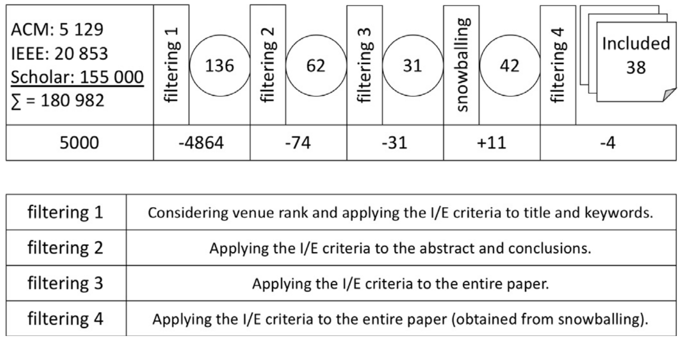

Fig. 1. Search method used in this study. The digital libraries were queried in January 2017.

2.2. Search strategy

The search strategy included an automatic search of digital libraries using a search string validated by experts. According to the SLR guidelines and lessons learned Brereton et al. (2007), the search string is comprised of keywords grouped into four categories.

```
ACM and Scholar

(1) (secur* OR privacy) AND
(2) (abuse OR misuse OR risk OR threat* OR attack* OR flaw*)
    AND
(3) (analysis OR assess* OR model* OR management OR
    elicit*) AND
(4) (system OR software OR application)

IEEE

(1) (secur* OR privacy) AND
(2) (abuse OR misuse OR risk OR threat* OR attack* OR flaw)
    AND
(3) (analysis OR assess* OR model* OR elicitation) AND
(4) (system OR software OR application)
```

We conducted pilot searches in order to refine the search string. While doing so, we excluded the keywords that did not produce additional search results. Furthermore, due to additional constraints imposed by the digital libraries (IEEE Xplore), we restricted the number of keywords and "wildcard" characters (*). To this aim, we have decided to use a second, similar search string used to search within keywords, title, abstract and full text of the publications. Keywords related to security and privacy are in the first group of terms. Keywords limiting the search results to black-hat (type) of techniques are in the second group of terms. The third group of keywords specifies the activity of the target techniques. Finally, the fourth group of keywords limits the scope to software, systems and applications.

Fig. 1 shows the adopted search method of this study. We adopt two techniques to search the existing literature in this study: (i) automatic search of digital libraries and (ii) backwards snowballing.

2.2.1. Digital libraries

We have obtained studies from the digital libraries using the search string. In January 2017, ACM returned 5129 titles, IEEE Xplore 20,853 and Google Scholar 155,000 search results. The search results were ordered by relevance and cut to top 2000 for ACM and IEEE and to top 1000 for Google Scholar, resulting in a total of 5000 search results. We then proceeded to filter the search results in several steps as shown in Fig. 1.

The CORE (Core rankings portal computing research & education, 2017) ranking portal was used for assessing the rank for both conference and journal venues. The portal provides two separate search

Table 1
Inclusion and exclusion criteria.

	Inclusion criteria
1.	Primary studies
2.	Studies (i.e. papers) that address methodologies, methods or techniques for identifying, prioritizing and analyzing security threats to a system including a software component.
3.	Studies that relate to software design.
4.	Studies that relate to security or privacy of software related systems.
	Exclusion criteria
1.	Studies written in any language other than the English language.
2.	Short publications and posters (< 3 pages).
3.	Publications at venues with a CORE rank below B (explained in Section 2.2).
4.	Publications that were unavailable through the search engine.
5.	Studies that focus on concrete mitigation strategies, security solutions, taxonomies of security threats and security analysis of systems.
6.	Studies that focus on anomaly detection and intrusion detection systems.
7.	Publications about safety-hazard analysis and detection methods and studies investigating the relationships between safety and security requirements.

interfaces, one for conference venues[2] and one for journal venues .[3] Journal venues are ranked based on the ERA ranking process Australian Research Council. The ranking assigns conference and journal venues into the following categories: (i) *A** - leading venues in a discipline area, (ii) *A* - highly respected venues, (iii) *B* - good venues, (iv) *C* - venues meeting the minimum standards, and (v) *Unranked* - insufficient quality data has been provided to determine the ranking. In the first filtering step (filtering 1) the publications presented at a venue with CORE rank below B were excluded. The publications that were presented at an unranked venue required further investigation for exclusion. The inclusion and exclusion criteria (Table 1) were manually applied to the title and keywords. After this step, the amount of search results considerably decreased to 136.

In the second filtering step (filtering 2) the inclusion and exclusion criteria (Table 1) were applied to the abstract and conclusion sections of the 136 remaining publications. After this step, the amount of search results decreased to 62.

Finally, 62 papers were read entirely. In the third filtering step (filtering 3) the inclusion and exclusion criteria were applied to the entire paper, which resulted in the exclusion of 31 papers. After this step, the amount of search results decreased to 31.

2.2.2. Snowballing

We have also performed the backward snowballing search technique (Wohlin, 2014). Essentially, this involves repeating the entire search strategy on the referenced work of a final set of papers. In our

[2] http://portal.core.edu.au/conf-ranks/.
[3] http://portal.core.edu.au/jnl-ranks/.

Table 2
Assessment criteria corresponding to research question RQ1.

Characterization (RQ1)		
Applicability	Level of abstraction	Requirements level
		Architectural level
		Design level
		Implementation level
Input	Type	Goals
		Requirements
		Attacker behavior
		Security assumptions
		Architectural design
		Source code
	Representation	Textual description
		Model-based
		Other
Procedure	Knowledge based	No
		Yes
	Level of precision	None
		Based on examples
		Based on templates
		Semi-automated
		Very precise
	Security objectives	Confidentiality
		Integrity
		Availability
		Accountability
		Not applicable
	Risk	Not considered
		Internal part of technique
		Externally considered
	Stopping condition	Present
		Not present
Outcomes	Type	Mitigations
		Threats
		Security requirements
	Representation	Structured text
		Model-based
		Other
	Assurance of quality	Explicit
		Present
		Not present
	Granularity	High-level
		Low level

Table 3
Assessment criteria corresponding to research question RQ2.

Ease of adoption (RQ2)	
Tool support	None
	Prototype tool
	Tool
Guidance for execution	Coarse grained phrases
	Fine grained steps
	No structure
Documentation	Publication
	Tutorial
	Presentations
	Tool documentation
	Demonstration
Target audience	Engineer
	Security trained engineer
	Security expert
	Researcher

case, snowballing was performed on 31 papers. In the fourth filtering step (filtering 4) the inclusion and exclusion criteria were applied to the entire paper obtained by backwards snowballing. After this step, the amount of search results increased to 38, leading to the final primary studies.

Table 4
Assessment criteria corresponding to research question RQ3.

Validation (RQ3)	
Type	Case study
	Experiment
	Illustration
Validator	None
	Author
	3rd party
	both
Domain	Automotive
	IS
	SOA
	SCADA
	...

2.3. Inclusion and exclusion criteria

Table 1 presents the summarized inclusion and exclusion criteria. We restricted the review to work published at any time before January 2017 that present a contribution in the area of threat analysis throughout the Software Development Life-cycle. The first five exclusion criteria in Table 1 are self-explanatory. We have noticed that a large amount of search results focused on anomaly detection and intrusion detection systems. Furthermore, the search results contained a lot of work published on safety-hazard analysis and relationships between safety and security requirements. For these reasons we added the last two exclusion criteria (6 and 7 in Table 1).

2.4. Data extraction

Tables 2–4 depict the assessment criteria used to record the information that was needed to answer the research questions. We have extracted the information from 38 publications by building a database of the identified techniques and corresponding assessments. In this section, we provide the rationale behind some of our choices for criteria levels.

The *types of input* were determined by choosing the most commonly required information for threat analysis to start. This includes requirements (functional or non-functional), attacker behavior, security assumptions, architectural design, source code and goals. The term "goal" is often used as a general term, yet this work makes a distinction between requirements (i.e. goal refinements) and goals. Threat analysis of a system requires at least: (i) the knowledge of what the system is (architecture, source code, functional requirements) and (ii) what it should be protected from (security assumptions, attacker behavior).

Studies have shown (e.g. Yuan et al., 2015; Wang et al., 2017; Williams, 2015) that including *knowledge base* (e.g. taxonomies, catalogs of misuse and abuse cases, attack scenarios and trees, etc.) helps the analyst to identify and analyze threats. Therefore we were interested to record which existing techniques provide a knowledge-base. We have assessed the techniques as knowledge based if the they are supported by some external source of information which helps raise the quality of outcomes. For instance, some techniques provide a catalog of example threats (e.g. STRIDE Shostack, 2014; Torr, 2005), templates (e.g. misuse cases) or even use one of the existing databases (such as CAPEC[4], CWE[5], CVE[6]) to compute threat suggestions.

In addition to knowledge base, we are interested in observing the *precision* of the analysis techniques. By "precision", we refer to the repeatability or reproducibility of the obtained results. In other words, "precision" is the degree to which repeated measurements under

[4] https://capec.mitre.org/.
[5] https://cwe.mitre.org/.
[6] https://cve.mitre.org/.

unchanged conditions show the same results (Taylor, 1997). In our case, it would not be an ideal situation every time an expert makes security threat analysis on the same software artefact (e.g., software design, architecture, requirements, source code, etc.) under the same conditions and comes up with a different set of security threats as a result of this analysis. However, as it was shown by cognitive psychologists, humans are incorrigibly inconsistent in making a summary of judgments of complex information (Taylor, 1956; Osbeck and Held, 2014; Kahneman, 2011). As a result, humans frequently give different answers when asked to evaluate the same information twice and this leads to the lack of precision in expert judgments in “low validity environments”, which are environments that are not sufficiently regular to be predictable (Kahneman, 2011; Kahneman and Klein, 2009). Figuring out potential security threats for a software system by analyzing its high level design, more fine grained architectural design, its requirements documents or source code also corresponds to making predictions in a low validity environment, as there are many parameters one cannot think of in advance such as the configurations of the final software when it is deployed on site. It was shown that the existence of formulas and algorithms as a backup to expert judgment improves precision (Dawes, 1979). Therefore, we assessed the precision of each technique by observing whether the procedure of analysis is: (i) supported by a formal framework (very precise), (ii) supported by tools that semi-automate the analysis, (iii) based on templates, (iv) based on example threats or (v) not accounting for precision (none).

Since *risk assessment* plays an important role in threat prioritization, we have investigated to what extent the techniques consider risk. Namely, some studies focus on associating risk levels to identified threats, while others consider risk externally, e.g. by combining the technique with an external risk management framework.

Notice that in addition to assessing the type and representation of analysis outcomes, this work also investigates the *quality assurance of outcomes*. We assess the techniques on this criterion by observing whether the quality assurance of outcomes is explicit, present or absent (none). Analysis techniques that explicitly assure the outcomes for quality define this activity as part of the analysis procedure. For instance, if the outcomes are represented with models, the technique may perform model verification as part of the analysis procedure (as presented by Xu and Nygard, 2006). However, explicit quality assurance of outcomes is not always present in the studied approaches. If the techniques provide informal guidelines for assessing the quality of outcomes (such as a checklist of most common threats), this study still considers that a form of quality assurance is present. Finally, this work investigates the *granularity of outcomes*. We assess the granularity of outcomes with two levels: high-level and low-level outcome. For example, the analysis technique presented by Almorsy et al. (2013) projects the outcomes on models, which can be transformed into source code. Therefore, we have assessed that this technique produces a low-level outcome. On the other hand, Haley et al. (2004) present so called “threat descriptions”, which are descriptive phrases of the form: performing action “X” on/to asset “Y” could cause harm “Z”. Therefore, we have assessed that this technique produces a high-level outcome.

As per RQ2 (Table 3), this study also assesses the available support for executing a threat analysis technique. Coarse grained *guidance for execution* include high-level overview of the technique with less detailed descriptions (only using key verbs, for instance). For instance, describing the threat identification as “brainstorming threats with participants” is considered as a coarse-grained guideline. Whittle et al. (2008) provide a recommended process for developing and testing executable misuse cases. Yet, the authors do not further explain how the attack scenarios are identified or how the mitigations are supposed to be re-designed in case the simulation ends in a successful attack. On the other hand, Chen et al. (2007) exemplify how to use the supporting tool by describing one instance run. Guidelines are considered to be fine-grained if they include precise instructions for analysis execution.

The *target audience* for the techniques was assessed to understand the minimum knowledge and skills required in order to execute each analysis technique. We identified four levels of competency, three of which are aligned to the Software Assurance Competency Model presented in Hilburn et al. (2013). Table 5 shows the mapping between the competency levels and the target audience considered in our work. An engineer is considered to posses the knowledge and skills of the competency level L1. A security trained engineer is considered to possess an active knowledge of security related concepts and has an engineering degree (BSc and/or MSc), which corresponds to levels L2-L3. Finally, a security expert corresponds to the levels L4-L5. In this work we consider researches to possess an active knowledge of practical as well as theoretical concepts in the field of security in software engineering.

As per RQ3 (Table 4) we have developed the assessment criteria to understand how each technique was validated. We have assessed the validation of each technique with three levels: case study, experiment and illustration. A *case study* is sometimes a rather loosely used term in software engineering. According to Runeson and Höst (2009) the presented case studies in software engineering range from very ambitious and well organized studies in the field, to small toy examples that claim to be case studies. A case study is a research methodology used to study a real phenomena of *exploratory, descriptive, explanatory* and *improving purpose*, the later being most popular in software engineering (Runeson and Höst, 2009). It requires rigorous planning, data collection and triangulation, data analysis, a discussion on threats to validity and evidence based conclusions. This work considers all applications of the proposed approach to a real world problem as case studies, in spite of only a handful (if any) conforming to the previous definition. In addition to case studies our assessment criteria includes two other forms of validation, namely illustrations and experiments. In contrast to the lightweight illustrations, experiments measure the effects of manipulating dependent variable(s) on an independent (response) variable. Experiments identified within this study were mostly experiments in empirical software engineering (e.g. comparative experiment of two techniques).

2.5. Quality assurance in this study

The selection of primary studies and data extraction was performed by a single researcher (first author). In order to circumvent the effects of potential bias, the following quality assurance plan was implemented.

2.5.1. Random assessment of included/excluded publications

We have randomly selected 10% of search results and the second author has applied the inclusion and exclusion criteria independently (filtering 1 in Fig. 1). The outcome has been compared to the results of the first author. The few disagreements (2 papers) of plausible exclusion were discussed between the two researchers until an agreement was reached. A summary of this discussion was submitted to the third author of this study for further assurance. In summary, we are confident that the inclusion and exclusion criteria was crisp enough to minimize any selection bias.

2.5.2. Random quality check of data extraction

A second random quality check was performed by the second author with regards to the data extraction. A random sample of the included publications (5 publications, roughly 10%) was independently re-assessed. The outcomes of this quality check were again compared to the outcomes obtained by the first author. This comparison yielded to a few discrepancies due to the perceived definition of certain criterion levels (6 out of 67 criterion levels). The first and second author revisited the precise differences between: (i) tool and prototype tool, (ii) engineer with security background and security expert, (iii) goal and requirement, (iv) design and architectural level of abstraction, (v) external and internally considered risk, and (vi) case study and illustration. After a

Table 5
Target audience considered in this work in relation to the competency levels in Hilburn et al. (2013).

Target audience	Level	Major tasks	Exemplary title
Engineer	L1	Tool support, low-level implementation, testing, and maintenance	Junior software developer, Acceptance tester, Junior security engineer, Software assurance technician
Security trained engineer	L2-L3	Requirements fundamentals and analysis, architectural design, implementation, risk analysis and assessment	Security analyst, Release engineer, Information assurance analyst, Maintenance engineer, Senior software developer, Software architect
Security expert	L4-L5	Assurance assessment, assurance management, risk management across the SDL, advancing in the field by developing, modifying, and creating methods, practices, and principles at the organizational level or higher	Project manager, Senior software architect, Chief information assurance engineer, Chief software engineer
Researcher	–	Remain in touch with the current research and publish own research in the discipline of security in software engineering	PhD student, Post Doctoral candidate, Assistant Professor, Senior lecturer, etc.

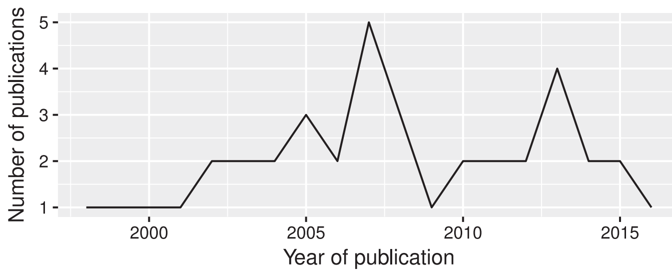

Fig. 2. Year of publication for the selected techniques.

consensus was reached, the first author of this study manually examined the rest of the publications (90%) to assure that the assessments were correct. To conclude, the authors are confident that the data was extracted correctly.

2.5.3. Continuous soundboarding

Informally, several sessions were held with all authors to maintain the quality of the review. For instance, the list of publications obtained from the initial pilot review were discussed. Further, the inclusion, exclusion and assessment criteria were refined during such sessions. These sessions were held continuously as sanity checks for the first author.

3. Results

In this section, we first overview the techniques and then present the answers to the research questions. We conclude this section with recommendations for practitioners, offering insight to the reader for when to use certain techniques.

3.1. Overview of threat analysis techniques

Fig. 2 shows a time-line of the 38 publication included in this SLR. Overall, the interest in the area of threat analysis approaches seems to be rather constant with an average of 2 publications per year. Despite a the slight progression of publications observed from early 2000 until 2007, it is difficult draw tendencies due to the small numbers.

Table 6 depicts threat analysis techniques included in this SLR. Most commonly used techniques in the presented body of knowledge were misuse cases, attack trees, problem frames and several software-centric approaches.

3.1.1. Misuse cases (MUC)

are derived from use cases in requirements engineering. In the form of templates, they are used to capture textual descriptions of threat paths, alternative paths, mitigations, triggers, preconditions, assumptions, attacker profiles, etc. The literature also mentions abuse cases, MUC maps and MUC scenarios. The difference between misuse and abuse cases is subtle and the two terms are sometimes used interchangeably. Strictly speaking, abuse is misuse with malicious intent. MUC maps and scenarios both focus on representing chained attacks, from start to the end of vulnerability exploitation.

Another way of identifying alternative paths of attack is by using *attack (or threat) trees*, where the root node is refined into leaves representing all possible attacker actions. Therefore an attack path is a single path starting at leaf node leading to the root node. Attack trees are commonly adopted in a combination with other techniques. For instance, LINDDUN Deng et al. (2011) proposes a combined analysis by first mapping the threats to (DFD) elements, using threat tree patterns and usage scenarios in order to identify MUC scenarios.

Much like threat patterns, *problem frames* are used to describe problems in software engineering. They define an intuitively identifiable problem class in terms of its context and the characteristics of its domains, interfaces and requirements (Jackson, 2001). As such problem frames are rather general in scope, therefore conceptualized security problem frames were soon introduced (Hatebur and Heisel, 2005; Beckers et al., 2013b; Lin et al., 2004).

Goal-oriented requirements engineering (GORE) perceives systems as a set of agents communicating in order to achieve goals. In GORE goals (or anti-goals) are refined until finally requirements (or anti-requirements) are achieved.

Finally, several software-centric techniques are well recognized in the software engineering community, particularly in the industrial space, such as STRIDE (Shostack, 2014; Torr, 2005), CORAS (Lund et al., 2011), P.A.S.T.A (UcedaVelez and Morana, 2015), DREAD (Owasp, 2017), Trike (Saitta et al., 2005), to name a few.

Table 7 shows the analysis techniques, their respective domains of validation and tool support. It is generally acceptable to group threat analysis techniques into risk-centric, attack-centric and software-centric techniques.

3.1.2. Risk-centric

threat analysis techniques focus on assets and their value to the organization. They aim at assessing the risk and finding the appropriate mitigations in order to minimize the residual risk. Their main objective is to estimate the financial loss for the organization in case of threat

Table 6
Threat analysis techniques. Note that, some publications were grouped by leading authors, sometimes resulting in observing separate techniques rather that fully fledged methodologies.

Methodology	Ref	Technique
Abe et al.	Abe et al. (2013)	Threat patterns, negative scenarios
Almorsy et al.	Almorsy et al. (2013)	Attack scenarios
Attack and Defense Trees	Mauw and Oostdijk (2005); Kordy et al. (2014)	Attack trees, defense trees
Beckers et al.	Beckers et al. (2013a)	MUC
Berger et al.	Berger et al. (2016)	DFDs, rule-based graph matching
CORAS	Lund et al. (2011)	Threat, risk, treatment diagrams and descriptions
Chen et al.	Chen et al. (2007)	Attack paths
Dianxiang Xu and K. E. Nygard	Xu and Nygard (2006)	Petri-nets
El Ariss and Xu	El Ariss and Xu (2011)	State charts
Encina et al.	Encina et al. (2014)	Misuse patterns
Extended i*	Elahi and Yu (2007); Mouratidis and Giorgini (2007); Elahi et al. (2010); Mouratidis et al. (2003)	Attacker agents with goals
Haley et al.	Haley et al. (2008, 2004)	Threat tuple-descriptions with rebuttals to claims
Halkidis et al.	Halkidis et al. (2008)	STRIDE, Fault tree analysis
Hatebur, Heisel et al.	Beckers et al. (2013b); Hatebur and Heisel (2005); Lin et al. (2004)	Problem frames
J. McDermott et al.	McDermott (2001); McDermott and Fox (1999)	Abuse cases
KAOS	Van Lamsweerde et al. (2007); Van Lamsweerde (2004); Van Lamsweerde and Letier (2000)	Threat graphs rooted in anti-goals, anti-models, threat trees
Karpati et al.	Karpati et al. (2010b,a)	MUC maps, MUC, attack trees
LINDDUN	Deng et al. (2011)	Threat to (DFD) element mapping, threat tree patterns, MUC scenarios
Liu et al.	Liu et al. (2003)	Attacker agents with goals
P.A.S.T.A.	UcedaVelez and Morana (2015)	Threat scenarios with associated risk and countermeasures
STRIDE	Shostack (2014); Torr (2005)	Threat to (DFD) element mapping
Sheyner et al.	Sheyner et al. (2002)	Attack graphs
Sindre and Opdahl	Sindre and Opdahl (2005)	MUC
Tong Li et al.	Li et al. (2016)	Automated generation of attack trees
Tøndel et al.	Tøndel et al. (2010)	MUC, attack trees
Whittle et al.	Whittle et al. (2008)	MUC

occurrence (e.g. CORAS Lund et al., 2011). Therefore, when risk-centric techniques are used assets dictate the priority of elicited security requirements.

On the other hand, *attack-centric* threat analysis techniques focus the analysis around the hostility of the environment. They put emphasis on identifying attacker profiles and attack complexity for exploiting any system vulnerability (e.g. Attack trees Mauw and Oostdijk, 2005). Their main objective is to achieve high threat coverage and identify appropriate threat migitations.

Finally, the literature also mentions so-called *software-centric* threat analysis techniques. This group includes techniques that focus the analysis around the software under analysis. For example, in STRIDE Shostack (2014); Torr (2005) the analysis is performed on DFDs, which provide a high-level architectural view of the software.

However, not all threat analysis techniques can be categorized in the aforementioned three groups. For instance, in GORE the main goal could be “stealing the GPS coordinates of a vehicle fleet”. In this case, the analysis would clearly evolve around that particular asset and could be therefore considered as risk-centric. Yet, the main goal could also be “malicious access to a DNS server”. In this case, the discussions and the analysis can be considered as attack-centric. For this reason, we categorize the techniques also as “GORE”, “SRE” and “Privacy”, as shown in Table 7.

Overall, a majority of techniques are attack-centric (≈ 45%) and requirements engineering approaches (GORE ≈ 20%, Security Requirements Engineering (SRE) ≈ 15%). We continue to present the results for the research questions in the subsequent sections.

3.2. RQ1: Characteristics

Table 8 shows the characteristics of the applicability and input of the selected techniques.

3.2.1. Applicability (RQ1.1)

In general, threat analysis can be performed iteratively at several stages of the software development. In this study, we differentiate between abstraction levels according to the input information required for analysis execution. We have assessed each technique for applicability at the level of requirements, architecture, design and implementation. For instance, in order to create and manually analyze attack trees the analyst only needs high level goals (or anti-goals). Therefore, the most basic form of attack trees are applicable at the level of requirements and architecture. In this study we make a distinction between the design and architectural level of abstraction.

On the *architectural level of abstraction* requirements are used in order to construct the architecture. Software architecture is a set of principal design decisions made about the system (as defined by Medvidovic and Taylor (2010)). The outcomes of this level of abstraction are high-levels diagrams (such as DFDs), sequence diagrams, flowcharts etc. The word “principal” here indicates that not all design decisions are architectural. In fact many design decisions are related to the domain, algorithms, programming languages or are based on preference.

Therefore, *designing* the intended architecture includes committing to a set of architectural styles and patterns, which are further refined until a detailed design is evaluated against the system requirements. The outcomes of this level of abstraction include most (or all) the design decisions made about the system (e.g. component diagrams, connector types and interfaces, deployment diagrams, etc).

Only two techniques are applicable at the level of implementation, where a concrete system is taken into account. First, Almorsy et al. (2013) describe a semi-automated Model-Driven Engineering (MDE) approach for a partial architecture reconstruction, followed by a risk-centric threat analysis. Second, Chen et al. (2007) presents a quantitative threat analysis approach based on attack-path analysis of COTS systems. Predominantly, the techniques are applicable at the level of requirements (14 techniques), architecture (14

Table 7
The selected analysis techniques.

Methodology	Ref	Approach	Domain	Tool	Validation
Abe et al.	Abe et al. (2013)	Attack-centric	IS	None	CS
Almorsy et al.	Almorsy et al. (2013)	Attack-centric	ERP, Web, E-commerce	None	EXP
Attack and Defense Trees	Mauw and Oostdijk (2005); Kordy et al. (2014)	Attack-centric	IS, other, ATM	Tool	CS, EXP, ILU
Beckers et al.	Beckers et al. (2013a)	Privacy	Cloud computing, E-bank	None	ILU
Berger et al.	Berger et al. (2016)	Attack-centric	Logistic application	Tool	CS
CORAS	Lund et al. (2011)	Risk-centric	Telecom, SCADE, IS	Tool	CS, EXP
Chen et al.	Chen et al. (2007)	Attack-centric	IT, COST	Prototype	CS
Dianxiang Xu and K. E. Nygard	Xu and Nygard (2006)	Attack-centric	Web store	None	CS
Encina et al.	Encina et al. (2014)	Attack-centric	Cloud services	None	CS
Extended i*	Elahi and Yu (2007); Mouratidis and Giorgini (2007); Elahi et al. (2010); Mouratidis et al. (2003)	GORE	Web-IS	Tool(Elahi et al. (2010))	ILU(Elahi et al. (2010))
Haley et al.	Haley et al. (2008, 2004)	SRE	Air traffic, HR, IS	None	CS, ILU(Haley et al. (2004))
Halkidis et al.	Halkidis et al. (2008)	Risk/Attack-centric	E-commerce	Tool	EXP
Hatebur, Heisel et al.	Beckers et al. (2013b); Hatebur and Heisel (2005); Lin et al. (2004)	Risk/Attack-centric	E-commerce	Tool	ILU(Hatebur and Heisel (2005); Lin et al. (2004)), CS (Beckers et al. (2013b))
J. McDermott et al.	McDermott (2001); McDermott and Fox (1999)	SRE	IS	None	CS (McDermott (2001))
KAOS	Van Lamsweerde et al. (2007); Van Lamsweerde (2004); Van Lamsweerde and Letier (2000)	GORE	E-commerce, Web store, Ambulance system	Tool	CS
Karpati et al.	Karpati et al. (2010b,a)	Attack-centric	Banking system, IS, Web-based IS	Tool (Karpati et al. (2010b))	CS, EXP
LINDDUN	Deng et al. (2011)	Privacy, GORE	Social network, E-health application	None	CS, EXP
Liu et al.	Liu et al. (2003)	SRE,GORE	IS	Tool	CS
P.A.S.T.A.	UcedaVelez and Morana (2015)	Risk-centric	Web-bankig application	None	ILU
STRIDE	Shostack (2014); Torr (2005)	Software-centric	IS, automotive, other	Tool	CS, ILU, EXP
Sheyner et al.	Sheyner et al. (2002)	Attack-centric	System Network	Tool	ILU
Sindre and Opdahl	Sindre and Opdahl (2005)	SRE	E-store, Telemedicine	Tool	CS, EXP
Tong Li et al.	Li et al. (2016)	Attack-centric	Smart grid	Prototype	CS
Tøndel et al.	Tøndel et al. (2010)	Attack-centric	IS	None	ILU
El Ariss and Xu	El Ariss and Xu (2011)	Attack-centric	Web store	None	CS
Whittle et al.	Whittle et al. (2008)	SRE	E-voting, CPS	Tool	CS

Table 8
The characteristics of the applicability and input of the selected techniques.

Methodology	Ref	Applicability				Input								
		Abstraction				Type						Representation		
		Requirements	Architectural	Design	Implementation	Requirements	Attacker behavior	Security assumptions	Architectural design	Source code	Goals	Textual description	Model-based	Other
Abe et al.	Abe et al. (2013)		•			•							•	
Almorsy et al.	Almorsy et al. (2013)		•	•	•	•		•	•	•			•	•
Attack and Defense Trees	Mauw and Oostdijk (2005); Kordy et al. (2014)	•	•								•			•
Beckers et al.	Beckers et al. (2013a)	•							•				•	
Berger et al.	Berger et al. (2016)		•	•					•				•	
CORAS	Lund et al. (2011)		•						•			•		
Chen et al.	Chen et al. (2007)			•	•		•	•				•		
Dianxiang Xu and K. E. Nygard	Xu and Nygard (2006)			•		•	•						•	
El Ariss and Xu	El Ariss and Xu (2011)		•								•	•		
Encina et al.	Encina et al. (2014)	•					•					•		
Extended i*	Elahi and Yu (2007); Mouratidis and Giorgini (2007); Elahi et al. (2010); Mouratidis et al. (2003)	•									•	•		
Haley et al.	Haley et al. (2008, 2004)	•	•			•		•	•			•	•	
Halkidis et al.	Halkidis et al. (2008)		•	•					•			•	•	
Hatebur, Heisel et al.	Beckers et al. (2013b); Hatebur and Heisel (2005); Lin et al. (2004)	•	•	•		•	•	•	•			•	•	
J. McDermott et al.	McDermott (2001); McDermott and Fox (1999)	•				•						•		
KAOS	Van Lamsweerde et al. (2007); Van Lamsweerde (2004); Van Lamsweerde and Letier (2000)	•									•	•		
Karpati et al.	Karpati et al. (2010b,a)		•	•			•		•			•		•
LINDDUN	Deng et al. (2011)	•	•					•	•			•	•	
Liu et al.	Liu et al. (2003)	•									•	•		
P.A.S.T.A.	UcedaVelez and Morana (2015)		•	•		•		•	•			•	•	•
STRIDE	Shostack (2014); Torr (2005)		•	•				•	•				•	
Sheyner et al.	Sheyner et al. (2002)			•					•				•	
Sindre and Opdahl	Sindre and Opdahl (2005)	•				•						•	•	
Tong Li et al.	Li et al. (2016)	•				•	•					•		•
Tøndel et al.	Tøndel et al. (2010)	•	•	•		•							•	
Whittle et al.	Whittle et al. (2008)	•				•							•	

Table 9
The characteristics of threat analysis procedure. KB = Knowledge Base.

		KB		Precision				
Methodology	Ref	Yes	No	None	Based on examples	Based on templates	Semi-automated	Very precise
Abe et al.	Abe et al. (2013)	•				•		
Almorsy et al.	Almorsy et al. (2013)	•			•		•	
Attack and Defense Trees	Mauw and Oostdijk (2005); Kordy et al. (2014)	•						•
Beckers et al.	Beckers et al. (2013a)	•				•		
Berger et al.	Berger et al. (2016)	•			•		•	
CORAS	Lund et al. (2011)		•	•				
Chen et al.	Chen et al. (2007)	•				•	•	
Dianxiang Xu and K. E. Nygard	Xu and Nygard (2006)		•					•
El Ariss and Xu	El Ariss and Xu (2011)		•	•				
Encina et al.	Encina et al. (2014)		•	•				
Extended i*	Elahi and Yu (2007); Mouratidis and Giorgini (2007); Elahi et al. (2010); Mouratidis et al. (2003)		•			•		
Haley et al.	Haley et al. (2008, 2004)		•					•
Halkidis et al.	Halkidis et al. (2008)	•				•		
Hatebur, Heisel et al.	Beckers et al. (2013b); Hatebur and Heisel (2005); Lin et al. (2004)		•			•		
J. McDermott et al.	McDermott (2001); McDermott and Fox (1999)		•	•				
KAOS	Van Lamsweerde et al. (2007); Van Lamsweerde (2004); Van Lamsweerde and Letier (2000)	•					•	•
Karpati et al.	Karpati et al. (2010b,a)		•	•				
LINDDUN	Deng et al. (2011)	•				•		
Liu et al.	Liu et al. (2003)		•	•				
P.A.S.T.A.	UcedaVelez and Morana (2015)		•	•				
STRIDE	Shostack (2014); Torr (2005)	•			•			
Sheyner et al.	Sheyner et al. (2002)		•				•	
Sindre and Opdahl	Sindre and Opdahl (2005)		•			•		
Tong Li et al.	Li et al. (2016)	•			•	•		
Tøndel et al.	Tøndel et al. (2010)	•					•	
Whittle et al.	Whittle et al. (2008)		•				•	

	Objectives					Risk				
Methodology	Confidentiality	Integrity	Availability	Accountability	Not applicable	Not considered	Internal part	Externally considered	Present	Not present
Abe et al.	•	•	•	•		•				•
Almorsy et al.	•	•	•	•			•			•
Attack and Defense Trees					•	•				•
Beckers et al.	•	•	•	•				•		•
Berger et al.	•	•	•	•		•				•
CORAS	•	•	•	•			•			•
Chen et al.	•	•	•				•			•
Dianxiang Xu and K. E. Nygard	•	•	•	•		•				•
El Ariss and Xu					•	•				•
Encina et al.	•	•	•	•		•				•

(continued on next page)

Table 9 (*continued*)

Methodology	Objectives					Risk				
	Confidentiality	Integrity	Availability	Accountability	Not applicable	Not considered	Internal part	Externally considered	Present	Not present
Extended i*	•	•	•	•				•		•
Haley et al.	•	•	•	•		•				•
Halkidis et al.					•		•			•
Hatebur, Heisel et al.	•	•	•	•		•				•
J. McDermott et al.					•	•				•
KAOS	•	•	•	•				•		•
Karpati et al.					•	•				•
LINDDUN	•			•				•		•
Liu et al.					•	•				•
P.A.S.T.A.	•	•	•	•			•			•
STRIDE	•	•	•	•				•		•
Sheyner et al.					•		•			•
Sindre and Opdahl					•			•		•
Tong Li et al.	•	•	•	•				•		•
Tøndel et al.					•		•			•
Whittle et al.					•	•				•

techniques) and design (11 techniques).

3.2.2. Input (RQ1.2)

The input of a threat analysis technique is all the information required in order to begin with threat identification. In order to understand the input information for each threat analysis technique, we have observed input type and representation. The *input type* can vary from high-level goals, requirements, attacker behavior, security assumptions, architectural design to source code of the system under analysis. For instance, the root node in an attack tree, typically referred to as an anti-goal, is decomposed into hierarchical leaves of possible attacker actions. Despite the domain knowledge and security expertise needed to find anti-goals and possible attack actions, the analyst does not require more than a high-level description of the system (e.g. in terms of its business functionality). The *input representation* was assessed with three levels: textual description, model-based and other.

One third of the analyzed techniques require as input architectural design (12 techniques) and one third requirements (11 techniques). Some techniques (6) consider the attacker behavior as input. Security assumptions are required for analysis in less than 25% of techniques (7). Only one technique takes source code into account as input to the analysis. Almorsy et al. (2013) present a technique where source code is an optional input to the analysis. Finally, in some techniques (5) high-level goals were used as input to the analysis. The required input is commonly represented either with textual descriptions (16 techniques), models (15 techniques) or both (6 techniques).

3.2.3. Procedure (RQ1.3)

Threat analysis procedure includes all required actions and tasks that the analyst needs to perform in order to obtain the desired outcomes. As depicted in Table 9, we assess the characteristics of the procedure that takes place during each analysis technique. To this aim we observe traces of knowledge base, precision, security objectives and risk in the procedure of each analysis technique.

On average about half of the techniques include a knowledge base of some kind. As previously mentioned, knowledge base (domain or security knowledge) helps the analyst to identify threats in the context of the system in question. Yet, we have found that most approaches take advantage of the existing knowledge base, rather than contribute with innovative examples (e.g. Hatebur and Heisel, 2005). For instance, Almorsy et al. (2013); Berger et al. (2016); Chen et al. (2007) and Tøndel et al. (2010) present formalized rules to extract knowledge from public repositories of threats and vulnerabilities namely Common Weakness Enumeration (CWE) (Martin, 2007) , Common Attack Pattern Enumeration and Classification (CAPEC) (Barnum, 2008), Open Web Application Security Project (OWASP) (Category:attack - owasp, 2017; Category:vulnerability - owasp, 2017).

In general, the precision of the technique procedures is on the level of templates and examples (about half of the publications). Four techniques (namely, Attack and defense trees Mauw and Oostdijk (2005); Kordy et al. (2014); Xu and Nygard (2006); Haley et al. (2004, 2008) and KAOS (Van Lamsweerde et al., 2007; Van Lamsweerde, 2004; Van Lamsweerde and Letier, 2000)) formally approach the analysis and are therefore very precise. Finally, six techniques (Almorsy et al., 2013; Berger et al., 2016; Chen et al., 2007, KAOS Van Lamsweerde et al., 2007; Van Lamsweerde, 2004; Van Lamsweerde and Letier, 2000; Sheyner et al., 2002; Sheyner and Wing, 2003; Tøndel et al., 2010; Whittle et al., 2008) introduce a semi-automated approach using tools (or prototype tools). According to our assessment, about a quarter of techniques (7) describe the analysis procedure with no regards towards the precision of the analysis.

A majority of techniques address security objectives explicitly in the presented approach. Some studies specifically mention only one security objective, yet in our assessment we include other security objectives that could be directly applied in the proposed approach. For instance, Hatebur, Heisel et al. Beckers et al. (2013b); Hatebur and

Heisel (2005); Lin et al. (2004) describe problem frames by introducing the authentication frame, therefore they consider confidentiality and integrity. However, the authors do not initialize possible problem frames for all security objectives. Ultimately, we do not see significant obstacles to introduce problem frames for other security objectives.

About half of the techniques (12) do not include risk assessment as part of the threat analysis technique. The rest of the studies are either risk-centric (7 techniques) or consider risk as an external activity (7 techniques).

3.2.4. Outcomes (RQ1.4)

As previously mentioned, we have observed the type and representation of outcomes. Table 10 shows the outcome characteristics of the selected techniques. We have monitored three *types of outcomes*: threats, mitigations and security requirements. All techniques present approaches that produce threats as main outcomes. Threat mitigations are security countermeasures planned for lowering the residual risk. Design-level security countermeasures are further on refined into implementable security requirements. Beyond threats as main outcomes, about half of the techniques also produce threat mitigations (15) and security requirements (12) as outcomes. In fact about a third of the techniques (8) produce all three types of outcomes (namely Xu and Nygard, 2006, Extended i* Elahi and Yu, 2007; Mouratidis and Giorgini, 2007; Elahi et al., 2010; Mouratidis et al., 2003; Haley et al., 2008; Haley et al., 2004; McDermott and Fox, 1999; McDermott, 2001, KAOS Van Lamsweerde et al., 2007; Van Lamsweerde, 2004; Van Lamsweerde and Letier, 2000, LINDDUN Deng et al., 2011; Sindre and Opdahl, 2005; Whittle et al., 2008).

In addition, we have observed the *representation of outcomes*. Most techniques result in outcomes represented with either a structured text (16), model-based form (16), or both (6). For instance, Whittle et al. (2008) introduce an aspect-oriented approach that results in finite state machines (model-based), misuse cases (model-based) and elicited security requirements (structured text).

Next to the type and representation, we have observed the *quality assurance of outcomes* for each analysis technique. Only a handful of techniques (3) have an explicit way of assuring the quality of outcomes (namely, Xu and Nygard, 2006; Haley et al., 2008; Haley et al., 2004, KAOS Van Lamsweerde et al., 2007; Van Lamsweerde, 2004; Van Lamsweerde and Letier, 2000). For example, Haley et al. (2008, 2004) include an activity for constructing satisfaction arguments as part of the procedure. The satisfaction arguments are used in order to verify whether the primary and secondary goals are satisfied with the resulting security requirements. The rest of the techniques do not have an explicit activity for quality assurance of outcomes. E.g., Beckers et al. (2013a) present a method for information security management system for cloud IS that includes threat analysis based on patterns. Their structured approach is aligned with ISO 271001 security standard and includes guidelines for assuring the quality of outcomes. Therefore, some form of quality assurance of outcomes is present, yet not explicitly defined. We have observed a presence of some kind of quality assurance of outcomes in 6 techniques. Still, the quality assurance of outcomes is predominantly not present in the techniques. For instance, Abe et al. (2013) propose an interesting approach for threat pattern detection and negative scenario generation, using transformation rules on sequence diagrams. However, the authors do not evaluate or measure the quality of the generated negative scenarios.

Regarding the *granularity of outcomes*, only three of the techniques (namely Almorsy et al., 2013; Chen et al., 2007, and Tøndel et al., 2010) produce low-level outcomes (e.g. source code). Almost all of the techniques (25) result in outcomes of high-level abstraction, which is in line with the results obtained from observing the applicability of techniques (RQ1.1).

3.3. RQ2: Ease of adoption

As shown in Table 11, about a third of the techniques (9) do not include any tool support. The rest are supported by tools (13 techniques) or present a prototype tool (4 techniques). In general, the majority of studies include coarse-grained guidelines for execution, which could be inferred from the publication. Six techniques provide fine-grained guidelines, yet three of them are not supported by a tool. Furthermore, most approaches together with their tools are only documented in the respective publications. Only a handful of techniques provide a tool with precise guidelines on how to use it.

The target audience of the techniques are most commonly security experts (9) and security trained engineers (10). Most of the techniques describe approaches that do not require extensive knowledge of any research field. According to our assessment, only two techniques are targeted more towards researchers, namely Xu and Nygard (2006) and Halkidis et al. (2008). In general, knowledge base, automation and tool support can decrease the level of required expertise. Despite that, important steps in the analysis are still required by experts (namely, threat identification and prioritization). Two techniques are thoroughly documented and two of them could be used by engineers without further training (STRIDE Shostack (2014); Torr (2005) and KAOS Van Lamsweerde et al. (2007); Van Lamsweerde (2004); Van Lamsweerde and Letier (2000)). They are both knowledge-based (they provide example threats) and are well documented (they provide tutorials, presentations, etc.) Evidently, the better the approach is documented, the easier it is to apply in practice.

3.4. RQ3: Validation

As shown in Table 7, the majority of techniques (19) were validated with a case study. Despite of the recently increasing quantity of empirical studies in software engineering and the long history of advocating empirical research in software engineering, there is still room for improvement (Siegmund et al., 2015). About 20% of techniques (8) were validated also with an experiment, reflecting the immaturity of empirical research in the software engineering community. In addition, extensive validation (case studies and experiments) was often applied only in the domain of Web systems (E-commerce, Web store, E-bank, Social network, E-health, Web bank, E-store, E-voting). About one third of the techniques (9) were validated by illustrations.

3.5. Recommendations for practitioners

Previously reported results have not considered any preferences between the levels of our assessment criteria (e.g. model-based is preferred over textual input). To complement the objectively reported results, we include a more subjective reflection, where we aim at providing insights to the reader about the benefits of adopting certain threat analysis techniques. To this aim we have considered the amount of resource investment needed for adopting a technique. In order to simplify the discussion we categorize the resource investments into "small" and "large".

If the planned resource investment is *"small"*, the organization is likely to prefer using a technique as is, without any improvements. Additionally, if *security is not prioritized*, the allocated budget might be sufficient only for recruiting a security trained engineer (e.g. requirements engineer). In this case, the time spent on threat analysis is limited. In addition, the target audience of the technique should be engineers with or without security training. Therefore, tool availability (and maturity), documentation, low target audience and a lightweight procedure (i.e. level of precision is none, based on examples or templates) are the most valued criteria for technique selection. According to the results of this study, techniques originating in requirements engineering fit this description. In our opinion, the tier techniques that could be adopted in this kind of organizations are CORAS (Lund et al.,

Table 10
The outcomes characteristics of the selected techniques.

Methodology	Ref	Type			Representation		Quality assurance			Granularity	
		Mitigations	Threats	Security requirements	Structured text	Model-based	Explicit	Present	Not present	High-level	Low-level
Abe et al.	Abe et al. (2013)		•			•			•	•	
Almorsy et al.	Almorsy et al. (2013)		•		•				•	•	•
Attack and Defense Trees	Mauw and Oostdijk (2005); Kordy et al. (2014)	•	•			•		•		•	
Beckers et al.	Beckers et al. (2013a)		•	•	•			•		•	
Berger et al.	Berger et al. (2016)	•	•		•				•	•	
CORAS	Lund et al. (2011)	•	•			•			•	•	
Chen et al.	Chen et al. (2007)	•	•		•				•		•
Dianxiang Xu and K. E. Nygard	Xu and Nygard (2006)	•	•	•		•	•			•	
El Ariss and Xu	El Ariss and Xu (2011)		•			•			•	•	
Encina et al.	Encina et al. (2014)	•	•		•				•	•	
Extended i*	Elahi and Yu (2007); Mouratidis and Giorgini (2007); Elahi et al. (2010); Mouratidis et al. (2003)	•	•	•		•			•	•	
Haley et al.	Haley et al. (2008, 2004)	•	•	•	•		•			•	
Halkidis et al.	Halkidis et al. (2008)		•		•				•	•	
Hatebur, Heisel et al.	Beckers et al. (2013b); Hatebur and Heisel (2005); Lin et al. (2004)	•	•		•	•			•	•	
J. McDermott et al.	McDermott (2001); McDermott and Fox (1999)	•	•	•	•	•		•		•	
KAOS	Van Lamsweerde et al. (2007); Van Lamsweerde (2004); Van Lamsweerde and Letier (2000)	•	•	•	•	•	•			•	
Karpati et al.	Karpati et al. (2010b,a)		•	•		•			•	•	
LINDDUN	Deng et al. (2011)	•	•	•	•				•	•	
Liu et al.	Liu et al. (2003)	•	•		•	•		•		•	
P.A.S.T.A.	UcedaVelez and Morana (2015)		•			•			•	•	
STRIDE	Shostack (2014); Torr (2005)		•		•				•	•	
Sheyner et al.	Sheyner et al. (2002)		•			•			•	•	
Sindre and Opdahl	Sindre and Opdahl (2005)	•	•	•	•	•			•	•	
Tong Li et al.	Li et al. (2016)		•	•	•				•	•	
Tøndel et al.	Tøndel et al. (2010)		•	•		•		•		•	•
Whittle et al.	Whittle et al. (2008)	•	•	•	•	•		•		•	

Table 11
The ease of adoption for techniques.

Methodology	Ref	Tool support			Execution			Documentation					Target audience			
		None	Tool	Prototype	No structure	Coarse-grained	Fine-grained	Publication	Tutorial	Presentations	Tool docum.	Demonstration	Engineer	Sec trained Engineer	Security expert	Researcher
Abe et al.	Abe et al. (2013)	•				•		•						•		
Almorsy et al.	Almorsy et al. (2013)			•		•		•							•	
Attack and Defense Trees	Mauw and Oostdijk (2005); Kordy et al. (2014)		•			•		•	•	•	•	•			•	
Beckers et al.	Beckers et al. (2013a)		•			•		•						•		
Berger et al.	Berger et al. (2016)		•				•	•						•		
CORAS	Lund et al. (2011)		•			•		•							•	
Chen et al.	Chen et al. (2007)			•			•	•		•			•			
Dianxiang Xu and K. E. Nygard	Xu and Nygard (2006)	•					•	•								•
El Ariss and Xu	El Ariss and Xu (2011)	•				•		•					•			
Encina et al.	Encina et al. (2014)	•			•			•					•			
Extended i*	Elahi and Yu (2007); Mouratidis and Giorgini (2007); Elahi et al. (2010); Mouratidis et al. (2003)		•			•		•					•			
Haley et al.	Haley et al. (2008, 2004)	•					•	•							•	
Halkidis et al.	Halkidis et al. (2008)	•				•		•							•	•
Hatebur, Heisel et al.	Beckers et al. (2013b); Hatebur and Heisel (2005); Lin et al. (2004)		•			•		•						•		
J. McDermott et al.	McDermott (2001); McDermott and Fox (1999)	•				•		•					•			
KAOS	Van Lamsweerde et al. (2007); Van Lamsweerde (2004); Van Lamsweerde and Letier (2000)		•				•	•	•	•	•				•	
Karpati et al.	Karpati et al. (2010b,a)		•			•		•			•			•		
LINDDUN	Deng et al. (2011)	•					•	•	•					•		
Liu et al.	Liu et al. (2003)		•			•		•						•		
P.A.S.T.A.	UcedaVelez and Morana (2015)	•				•		•		•				•		
STRIDE	Shostack (2014); Torr (2005)		•			•		•	•	•	•	•	•	•		
Sheyner et al.	Sheyner et al. (2002)		•			•		•							•	
Sindre and Opdahl	Sindre and Opdahl (2005)		•			•		•		•			•			
Tong Li et al.	Li et al. (2016)			•		•		•							•	
Tøndel et al.	Tøndel et al. (2010)			•		•		•						•		
Whittle et al.	Whittle et al. (2008)		•			•		•							•	

2011), Problem Frames (Hatebur, Heisel et al.) (Beckers et al., 2013b; Hatebur and Heisel, 2005; Lin et al., 2004), MUC (Sindre and Opdahl, 2005) and Abuse cases (J McDermott et al.) (McDermott, 2001; McDermott and Fox, 1999), Extended i* (Elahi and Yu, 2007; Mouratidis and Giorgini, 2007; Elahi et al., 2010; Mouratidis et al., 2003), KAOS (Van Lamsweerde et al., 2007; Van Lamsweerde, 2004; Van Lamsweerde and Letier, 2000). These techniques seem to require less effort to use as they are less systematic and thorough. They are more intuitive and are supported by toolkits such as RE-Tools.[7]

The aforementioned techniques lack guarantees for analysis correctness (i.e. quality assurance of outcomes). In organizations where *security is prioritized*, quality assurance of outcomes also becomes important. In this case, more effort for threat analysis is justified. Therefore, the existing budget for security is bigger, sufficient for recruiting security experts. The preferred techniques should not only have good tool support and documentation, but also be systematic, thorough, expert-based and possibly semi-automated. In our opinion STRIDE (Shostack, 2014; Torr, 2005) and LINDDUN (Deng et al., 2011) are the tier techniques that fit this description. STRIDE (similarly LINDDUN) is a systematic approach that visits each element in the DFD and is therefore subjected to the so called “threat explosion” problem. In order to counter the explosion problem, some automation (namely threat category generation) is already available by the MTM.[8]

The previously mentioned techniques require little additional effort since quality assurance of outcomes is not prioritized. However, if the planned resource investment is *“large”*, the organization is likely prepared for improving an existing technique to obtain an “in-house” adapted version. We also consider academic researchers looking for a starting point in their research to be prepared for a “large” investment of resources. These techniques should be systematic by construction (e.g. formal) but most importantly show potential for improvement (e.g. technology improvement). In our opinion, two such techniques stand out. First, the work of Berger et al. (2016) presents an interesting semi-automated technique for extracting threats from graphs based on rules matching certain CAPEC and CWE entries. The authors argue that the existing notation for DFDs needs to be extended with more security semantics. To this aim, Berger et al. extend the notation by annotating flows with assets, security objectives and type of communication (e.g. manual input). A more formal definition of security semantics might assure the quality of outcomes explicitly, which is a promising research direction. Further, querying graphs could be implemented using a different set of technologies. Therefore we believe that their approach is with some effort adaptable to the needs of the organization. Second, Almorsy et al. (2013) have used Object Constraint Language (OCL) to define attack scenarios and security metrics. The authors developed an Eclipse plug-in that is able to perform a trade-off analysis for different applications based on their signature evaluator. Minimizing the architectural design space with such a semi-automated trade-off analysis could indeed benefit organizations. For organizations that already practice MDE, this approach could be tailored to the models they use. Yet OCL constraints can only be as accurate as the model instances, therefore it might be promising to pursue this research outside the space of MDE.

4. Discussion

In this section we first discuss the applicability of threat analysis techniques in current trends in the software engineering community. We then proceed to discuss the main findings of this study. In summary, the main findings are the following:

(i) There is potential for improving threat analysis techniques in order to be applicable in the context of current trends in software engineering,
(ii) there is a lack of Definition of Done in the threat analysis procedures,
(iii) there is a lack of quality assurance of outcomes,
(iv) the use of validation by illustration is predominant which is worrisome,
(v) the tools presented in the primary studies lack maturity and are not always available.

4.1. Potential for improvement along current trends

Development and Operations (DevOps) is a software engineering practice that aims at unifying software development and operations by means of higher automation, measurement, sharing and promoting a specific culture in the organization. Commonly adopted activities, such as continuous integration and deployment cause the SDL to shorten considerably. Such organizations face significant challenges in providing the required security of the product under the rapid rate of software changes. Despite the immaturity of research in integrating security into DevOps, some efforts are summarized by Mohan and Othmane (2016). According to a recent survey performed with practitioners (Ur Rahman and Williams, 2016), the majority of participants believe that other security practices are prevalent in DevOps organizations (i.e. security policies, manual security tests, security configuration). To the best of our knowledge, threat analysis techniques have not been applied in the context of DevOps. In our opinion, there are three areas where existing techniques could be improved in order to cater to the needs of DevOps.

First, it is important that the information that was gained from threat analysis is automatically propagated to source code level (and vice-versa). It might be beneficial to assure the *traceability* between the threats and corresponding security requirements at the level of implementation. This might facilitate a more efficient reuse of analysis outcomes in the fast changing code base. Establishing a traceable link between architectural design and implementation can be achieved with a “top-down” or “bottom-up” approach. In a “top-down” approach, the architectural design decisions need to be annotated in the source code (e.g. as presented by Abi-Antoun and Barnes, 2010). Such annotations may have to be added manually by developers themselves, which could render the technique unreliable. Therefore, there are existing approaches to extract the architecture from the code base (i.e. Software Architecture Reconstruction (SAR)) by employing dynamic and/or static reverse engineering techniques (e.g. as presented by Granchelli et al., 2017). To the best of our knowledge, the existing tools supporting SAR have limitations and are not commonly applied to practice. From a usability perspective, practices such as continuous deployment cause uncertainty in the security implications of modified code base. For instance, it would be beneficial for developers to get instant feedback on how their contribution impacts the security of the code base (e.g. one threat is mitigated).

Second, the existing techniques would benefit from guidelines of how to *compose the analysis* outcomes. In practice, the software systems under analysis are too large and complex to be analyzed at once. Therefore, organizations are forced to scope the system into sub-systems and assign the analysis to several teams of experts to be analyzed simultaneously. As a results, border elements are either analyzed multiple times, or overlooked. One possible solution could be to scope the system according to assets. In this case, elements handling certain assets would be analyzed together in an end-to-end manner. To facilitate the composability of analysis outcomes, a level of formalism could be beneficial. For example, taint analysis has been used to analyze applications in order to present potentially malicious data flows to the human analyst. The analyst (or automated malware detection tool) is able to decide whether particular flows constitute a policy violation. For instance, Arzt et al. (2014) present a flow-sensitive taint analysis

[7] http://www.utdallas.edu/~supakkul/tools/RE-Tools/.
[8] https://www.microsoft.com/en-us/download/details.aspx?id=49168.

tool for Android applications. One possible research direction could be in using hybrid taint analysis techniques on architectural models.

Third, the analysis performed for one subsystem is related to security assumptions, which may not be in line with the security assumptions of another subsystem. Further, threats with high impacts to the organization are typically prioritized. Threat prioritization is commonly still performed manually, which demands a lot of resources. Therefore, existing analysis techniques need to invest in *impact analysis* automation.

The literature states that some *Agile* practices such as Extreme Programming (XP) are not suitable for high-reliability requirements (Sharma and Bawa, 2016). Similarly to DevOps, agile development practices require highly automated threat analysis techniques due to short sprints. Incidentally, start ups and Agile organizations adopting novel software engineering practices with less supervision are facing similar challenges.

To conclude, in light of DevOps and Agile, where software development is driven by change, there are three important aspect where existing analysis techniques have yet to mature: (i) traceability of analysis in the code base, (ii) composability of analysis outcomes and (iii) threat impact analysis automation.

4.2. Definition of Done (DoD)

Threat analysis is typically performed on a certain level of abstraction. The level of abstraction is determined during the first session by system architects and security experts. However, the analysts will typically also consider threats on a lower level of abstraction, depending on their feasibility. It is up the experts to determine which parts of the system should be analyzed in detail (on a low abstraction level). Analysts are also faced with the challenge of deciding how many identified threats (and at what level of abstraction) are enough for a "good" analysis of a particular sub-system. In the Agile community, a so called Definition of Done (DoD) is used for guiding Scrum teams to program more efficiently and minimize technical debt. For example, checklists of test cases can be used to define when a planed release is finished. We borrow this term from the Agile community in order to depict the lack of similar practices in existing threat analysis techniques. Defining when threat analysis can be concluded is still an open question, which is today handled by practitioners in an ad-hoc manner. Future work in this direction could have a large impact on the IoT domain, where systems are composed of a large number of middle-ware components and devices.

4.3. Lack of precise guidelines

We have found that there is a lack of precisely defined rules for the analysis. Some techniques operationalize rules for discovering threats and apply them on a graph representing the architectural model (e.g. Almorsy et al., 2013). Yet, the guidelines provided by authors for using their plug-in tool are vague and informal. Moreover, El Ariss and Xu (2011) refer to the process of constructing attack trees as goal refinements that continue until the desired level of abstraction is reached. Such guidelines are not precise and, for instance, do not elaborate on how to identify AND/OR gates of attack trees. The lack of precise guidelines effects the techniques' ability to assure the quality of outcomes. We have rarely found that the techniques have an explicit way of determining how well the analysis was performed. While some approaches check for the number of threats found in comparison to a baseline analysis, only a handful do so systematically and automatically.

4.4. Generalization across domains

As shown in Table 7 the domains of validation vary, yet the majority are still applied to Web-based systems. However, traditional threat analysis techniques appear to be used in some form independently of the domain. In particular, we have discussed the commonly used varieties and combinations of (i) STRIDE (Shostack, 2014; Torr, 2005), (ii) attack trees (Mauw and Oostdijk, 2005), graphs and paths (Chen et al., 2007), (iii) MUCs (Sindre and Opdahl, 2005) (iv) problem frames (Hatebur and Heisel, 2005) and threat patterns (Abe et al., 2013). We argue that the aforementioned techniques are more general in nature and are therefore easily applicable across domains. Unfortunately, we have found that most approaches are poorly validated (using illustrations on toy examples) and the limited tool support typically only aids the graphical representation of threats, rather than the analysis of threats. The lack of validation across different domains questions the applicability of analysis techniques to current trends in software engineering. Internet of Things (IoT) and Cyber-Physical Systems (in particular automotive) have recently been attracting a lot of attention.

IoT systems typically consist of a large amount of relatively small devices and sensors with limited capabilities functioning as individual agents to achieve goals. These interconnected devices are commonly analyzed individually, thus their vulnerabilities are well known. Yet new vulnerabilities may arise once the devices are connected. Therefore, a knowledge-base of threats and mitigations to the known vulnerabilities could aid in automating threat analysis for IoT devices and in maintaining the quality of analysis outcomes. Recent efforts have proposed analysis approaches in the domain of IoT formalizing the cyber-physical interactions including the malicious perspective. For instance, Mohsin et al. (2017) introduced a formal risk analysis framework based on probabilistic model checking. Their framework is able to generate system threat models, which are used to formally compute the likelihood and cost of attacks. Further, Agadakos et al. (2017) have introduced an approach for modeling cyber-physical attack paths in IoT using Alloy. Their approach also ultimately generates potential threats. Non-formal approaches supporting aspects of threat analysis in IoT have also been proposed. For instance, Geneiatakis et al. (2017) have built an attacker model covering security and privacy threats in a typical IoT system. Regarding usability Mavropoulos et al. (2017) presented a tool that supports security analysis of IoT systems. Rather than aiding the analysis procedure, the tool helps to visualize assets, threats and mitigations.

In the *automotive* domain, Threat Analysis and Risk Assessment (TARA) approaches are summarized by Macher et al. (2016). TARAs summarized in this review use traditional threat analysis approaches (such as CORAS (Lund et al., 2011) and Attack trees (Mauw and Oostdijk, 2005)) as well as approaches tailored for the automotive (e.g. HEAVENS Macher et al., 2016 and SAHARA (Macher et al., 2015) are adaptations of STRIDE (Shostack, 2014), EVITA (Committee et al., 2016) is based on Attack trees (Mauw and Oostdijk, 2005), etc). Threat analysis of novel autonomous vehicles is extremely lengthy and complex due to heavy safety and security requirements and compliance to standards (e.g. ISO 26,262 ISO, 2011). The automotive industry to this day relies predominantly on threat analysis performed manually by experts. Yet there is a need to semi-automate threat analysis procedures due to scarce resources (i.e. security experts). A risk-centric lightweight threat analysis technique could facilitate the identification of the most important threats in only a few sessions. In order to ensure compliance to safety and security standards, the problematic parts of the system still need to undergo a systematic threats analysis.

In summary, significant effort has been invested in researching failure analysis in the domains of Cyber-Physical Systems (e.g. Martins et al., 2015), IoT and automotive due to the required compliance to safety standards. Therefore, mature hazard analysis (safety) techniques have already been established (e.g. failure mode and effects analysis (FMEA) Commission et al., 2006 and fault tree analysis (FTA) IEC, 2006). On the other hand, there seems to have been less focus on threat analysis techniques, particularly in Agile development and DevOps, where security is often not a business priority.

4.5. Ease of adoption

In the space of threat analysis approaches, tools have been used for three main purposes: i) partially automating the analysis procedure, ii) graphically representing threats to the system and (iii) facilitating the analysis execution (i.e. helping the analyst to follow the procedure).

Semi-automated approaches utilize tools for the purpose of *automating* a part of the analysis procedure (Such as Berger et al., 2016; Almorsy et al., 2013; Whittle et al., 2008). For instance, Whittle et al. (2008) extended an existing tool in order to automatically weave mitigation scenarios into a set of core behavior scenarios. The authors are able to then generate a new set of finite state machines including both the initial behavior and the behavior including the mitigations. Finally, they execute the attack behavior on the new set of finite state machines to determine the success of the attack.

Manual threat analysis approaches are supported by tools for the purpose of retaining the *structure of the analysis* technique. For example, MUC (and MUC maps (Karpati et al., 2010b)) are a form of templates used in the process of analysis. The tools supporting threat analysis with MUCs only provide the required elements to model the misuse, such as graphical elements to represent attackers with an empty template for defining their abilities. Meanwhile threat identification is not supported by tools and is considered a brainstorming task. Similarly, Microsoft Threat Modeling Tool[9] provides the visual elements (e.g. boxes, arrows, ellipses, etc.) needed to create DFDs. To some extent, this tool also facilitates the proper execution of the analysis, as it generates categories of threats for each DFD element. The generated categories guide the analyst through the analysis procedure of the technique. However, threat categories generated based on the threat-to-element mapping table only provide a hint of what type of threats could be identified. Similarly, the open source SeaSponge[10] threat modeling tool primarily serves as a graphical aid to represent threats on a system model. Some primary studies present tools whose purpose is both to aid automation of analysis and provide graphical representation. For instance, Sheyner et al. (2002) and Sheyner and Wing (2003) present a tool for generating and analyzing attack graphs.

Tools serving the sole purpose of *graphical representation* are fairly straight forward to use just by drag-and-dropping elements on an empty canvas. Anyone with basic computer skills could easily use them. However, such graphical tools do not support threat identification and prioritization. The correctness and completeness of the results submitted by an engineer using such tools is not assured. One could argue that more expertise is required for the proper execution of the analysis using tools that only aid the graphical representation of threats. Our assessment suggests that tools supported by knowledge-base could to some extent leverage the security (and domain) expertise required for threat analysis. Further, introducing quality assurance features is very important for a novice analyst. Finally, partial automation could help speed up the analysis to facilitate efficient training of junior analysts. Several primary studies have the potential to be extended with tool support also targeting engineers, namely Berger et al., (Almorsy et al., 2013; Chen et al., 2007), KAOS (Van Lamsweerde et al., 2007; Van Lamsweerde, 2004; Van Lamsweerde and Letier, 2000; Halkidis et al., 2008), LINDDUN (Deng et al., 2011), STRIDE (Shostack, 2014; Torr, 2005). In summary, tool support seems to be a common trend in the primary studies, yet tool proposals are preliminary with limited validation.

5. Threats to validity

We consider *internal* and *external* threats to validity, as defined in Wohlin et al. (2012). Considering that substantial work was done by a single researcher, we consider a risk of subjectivity as an internal threat to the validity of this study. The bias introduced by the first author was mitigated by including random quality controls into the review process, particularly during the selection of primary studies and data extraction.

Furthermore, in this work we restrict our search of the literature by considering only top venues available in the digital libraries mentioned in Section 2.2. Consequently, we raise the risk of considering a non-representative subset of the relevant existing literature, thus harming the validity of our conclusions. However, as per focusing the search on top venues, we are confident that the selected papers represent the most influential work done in the area of secure design in software engineering.

In general, the validity of results of systematic literature reviews depend heavily on the external validity of the selected studies. We attempted to mitigate this issue by adopting a conservative exclusion criteria, disregarding grey literature papers, position papers and short papers (< 3 pages). Finally, due to resource limitations, not all aspects could be extracted from the data. For instance, further investigations could have been made regarding the learnability in relation to tool support of the identified threat analysis techniques.

6. Related work

To the best of our knowledge this is the only systematic literature review on threat analysis techniques. However, recently Cheung (2016) has contributed with a brief literature review of 8 threat analysis techniques. The main purpose of this work was to identify the added value and impact of adopting threat analysis techniques to cyber-physical systems of public water infrastructures. The author summarizes a subset of the primary studies analyzed in this work.

Threat analysis is used for the main purpose of security requirement elicitation or refinement. Hence, security requirements engineering approaches may include aspects of threat analysis. We continue to address the related work in the areas of security requirements engineering and risk analysis and assessment.

6.1. Security requirements engineering

Mellado et al. (2010) performed a systematic literature review concerning security requirements engineering methodologies, processes, frameworks and techniques. The authors selected 51 primary studies to investigate. Some of these studies are overlapping with our selection of primary studies (namely Mellado et al., 2007; Haley et al., 2008; Sindre and Opdahl, 2005; Whittle et al., 2008). Among assessing the selected studies based on a smaller set of criteria, the authors additionally present the integration of primary studies with security standards. Our work could also be extended to include the integration of primary studies with security standards, which would further aid practitioners.

Similarly, Salini and Kanmani (2012) have published a survey on security requirements engineering approaches. The authors present and compare SRE issues and methods. Additionally, the authors stress the importance of threat analysis in the early stages of software development. Yet, this survey focused on reviewing SRE frameworks and processes (e.g. Security Quality Requirements Engineering methodology (SQUARE) and Security Requirements Engineering Process (SREP)).

Further, Fabian et al. (2010) contributed with a conceptual framework for SRE with a strong focus on security requirements elicitation and analysis. The authors use the proposed framework to compare several SRE approaches. Similar to our study, Fabian et al. also investigate problem frames and other UML-based modeling approaches. Additionally, the authors also assess the quality of outcomes for the selected studies. In contrast to this study, the authors perform an unsystematic comparison of SRE methods, as opposed to a systematic comparison of threat analysis techniques.

Muñante et al. (2014) have performed a review of SRE methods

[9] https://www.microsoft.com/en-us/download/details.aspx?id=49168.

[10] https://github.com/mozilla/seasponge.

with a focus on risk analysis and model-driven engineering (MDE). The purpose of their work was to identify which SRE methods are compatible with existing risk analysis and MDE approaches. To this aim, Munante et al. have analyzed the existing work and concluded that KAOS and Secure i* are the most compatible SRE methods with a model-driven approach. They also concluded that extending them with risk analysis concepts is feasible. Despite the overlap in primary studies of this work, Munante et al. have based their analysis on a smaller set of assessment criteria and have done so unsystematically.

Daramola et al. (2012) have published a comparative review on i*-based and use case-based security modeling approaches. Their main findings show that both categories of approaches show conceptual similarities in the modeling aspects and method process. They also found several differences between both categories of approaches (namely, representational, supported activities and techniques, quality of outcomes and tool support).

Kriaa et al. (2015) have performed a survey of approaches combining safety and security for industrial control systems. The authors contribute with highlighting the main commonalities, differences and interconnections between safety and security in industrial control systems. A subset of the reviewed approaches overlap with our primary studies (namely, CORAS, MUC). Their review also considers Failure Mode and Effects Analysis (FMEA), Failure Mode, Vulnerabilities and Effect Analysis (FMVEA), Fault Tree Analysis (FTA), which are based on attack trees. In contrast to their work, this study only investigates threat analysis techniques from the security perspective.

6.2. Risk analysis and assessment

Latif et al. (2014) present a systematic literature review in the field of cloud computing with a focus on risk assessment. The purpose of their work was to categorize the existing approaches and explore which areas need further investigation. The authors selected 31 primary studies and have looked into the existing risks in cloud computing from the perspective of a customer and a provider. Their main finding is that topics such as data security and privacy are widely investigated, whereas. physical and organizational security, have received less attention. However, their literature review is narrowly scoped only to one domain and does not assess the characteristics of the selected works.

Cherdantseva et al. (2016) present a state-of-the-art review of the literature on cyber security risk assessment methods for SCADA systems. The authors have selected and examined 24 risk assessment methods. They provide descriptions of the methods and assess them with an elaborated criteria. Among other methods, the review also includes attack trees, petri net analysis, attack and defense modeling and CORAS. Interestingly, the authors propose several challenges for future work, some of which are in line with the findings of this work, namely (i) need for improving the validation of methods (ii) overcoming the attack-failure orientation (in this work referred to as determining the stopping condition), (iii) lack of tool support. Yet, their review has a strong focus on risk assessment methods in the context of SCADA systems.

Dubois et al. (2010) have contributed with a systematic approach to define a domain model of information system, which is used to compare, select of improve security risk management methods. The authors provide a literature review as part of their study, which also include threat analysis approaches CORAS, OCTAVE and Common Criteria. Yet, this study contributes with an ontology, rather then systematically reviewing the literature.

Raspotnig and Opdahl (2013) have compared risk identification techniques for safety and security requirements. The purpose of their work was to investigate whether and how the techniques can mutually strengthen each other. Among other methods, the authors also assess attack trees, MUC and KAOS. Similar to this study, the authors also look into stakeholders (in this study target audience) of the selected methods. One of their main findings was that security techniques can be strengthened by including better stakeholders and communication descriptions, while the safety techniques can benefit from a tighter integration between the risk identification and development processes. However, this study does not look into safety or the interaction between safety and security.

7. Conclusions and future work

In this study, we have performed a systematic review of 26 threat analysis approaches for secure software design. We have developed detailed assessment criteria reflecting our research questions, presented in Section 3. Our search strategy included an automatic search of three digital libraries and snowballing. The data was extracted from the primary studies according to the assessment criteria. The main findings of this study show that the existing techniques lack in quality assurance of outcomes. Furthermore, the techniques lack maturity, validation and tool support. Finally, they lack a clear definition of when the analysis procedure is done.

As per the results discussed in Section 4, we identify three possible directions for future work. First, a connection (feedback) between the intended and actual architecture might aid in understanding the reality of analysis outcomes. The quality of outcomes might only provide insightful speculations without a clear link to the actual architecture. Further, other architectural design decisions might have lead to architectural decay (Riaz et al., 2009), causing a disconnection to the “as-planned” security. To this aim, a formal language for design-level threat analysis may aid in establishing the link to the extracted architecture (e.g. by means of adapting dynamic and/or static reverse engineering approaches). Regarding the Definition of Done, we believe that further investigations are needed to understand the effects of composing analysis outcomes of subsystems. To this aim, the assets play an important role as border elements between subsystems (e.g. middle-ware). Further, a semi-automated way of composing analysis outcomes might facilitate analysis reuse for products in different stages of the SDL. Finally, in the context of DevOps and Agile, we believe that analysis velocity is preferred over analysis systematicity. Therefore, an analysis approach focusing on most important assets might be more appropriate for such organizations. To this aim, we are evaluating a risk-first lightweight approach for finding the most important threats sooner in the analysis procedure (Tuma et al., 2017).

Acknowledgments

This research was partially supported by the Swedish VINNOVA FFI project “HoliSec: Holistic Approach to Improve Data Security”.

References

Abe, T., Hayashi, S., Saeki, M., 2013. Modeling security threat patterns to derive negative scenarios. Software Engineering Conference (APSEC), 2013 20th Asia-Pacific. 1. IEEE, pp. 58–66.

Abi-Antoun, M., Barnes, J.M., 2010. Analyzing security architectures. Proceedings of the IEEE/ACM International Conference on Automated Software Engineering. ACM, pp. 3–12.

Agadakos, I., Chen, C.-Y., Campanelli, M., Anantharaman, P., Hasan, M., Copos, B., Lepoint, T., Locasto, M., Ciocarlie, G. F., Lindqvist, U., 2017. Jumping the air gap: Modeling cyber-physical a ACK paths in the internet-of-things.

Almorsy, M., Grundy, J., Ibrahim, A.S., 2013. Automated software architecture security risk analysis using formalized signatures. Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, pp. 662–671.

Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., Le Traon, Y., Octeau, D., McDaniel, P., 2014. Flowdroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. ACM Sigplan Notices 49 (6), 259–269.

Kitchenham, B., et al., 2007. Guidelines for performing systematic literature reviews in software engineering. Technical report, Ver. 2.3 EBSE Technical Report. EBSE sn.

Barnum, S., 2008. Common attack pattern enumeration and classification (capec) schema description. 3 Cigital Inc. http://capec.mitre.org/documents/documentation/CAPEC_Schema_Description_v1

Beckers, K., Côté, I., Faßbender, S., Heisel, M., Hofbauer, S., 2013. A pattern-based method for establishing a cloud-specific information security management system. Requir. Eng. 18 (4), 343–395.

Beckers, K., Hatebur, D., Heisel, M., 2013. A problem-based threat analysis in compliance with common criteria. Availability, Reliability and Security (ARES), 2013 Eighth International Conference on. IEEE, pp. 111–120.

Berger, B.J., Sohr, K., Koschke, R., 2016. Automatically extracting threats from extended data flow diagrams. International Symposium on Engineering Secure Software and Systems. Springer, pp. 56–71.

Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M., Khalil, M., 2007. Lessons from applying the systematic literature review process within the software engineering domain. J. Syst. Softw. 80 (4), 571–583.

Bsimm7, 2017. https://www.bsimm.com/. (Accessed on 12/08/2017).

Caralli, R.A., Stevens, J.F., Young, L.R., Wilson, W.R., 2007. Introducing octave allegro: Improving the information security risk assessment process.

Category:attack - owasp, 2017. https://www.owasp.org/index.php/Category:Attack. (Accessed on 11/02/2017).

Category:vulnerability - owasp, 2017 https://www.owasp.org/index.php/Category:Vulnerability. (Accessed on 11/02/2017).

Chen, Y., Boehm, B., Sheppard, L., 2007. Value driven security threat modeling based on attack path analysis. System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on. IEEE. 280a–280a

Cherdantseva, Y., Burnap, P., Blyth, A., Eden, P., Jones, K., Soulsby, H., Stoddart, K., 2016. A review of cyber security risk assessment methods for SCADA systems. Comput. Secur. 56, 1–27.

Cheung, C.Y.C., 2016. Threat Modeling Techniques. Technical Report. Faculty of Technology, Policy and Management, Delft University of Technology.

Core rankings portal computing research & education, 2017. http://www.core.edu.au/conference-portal. (Accessed on 12/04/2017).

Commission, I.E., et al., 2006. Analysis techniques for system reliability: Procedure for failure mode and effects analysis (FMEA). Int. Electrotech. Comm.

Committee, S.V.E.S.S., et al., 2016. Sae j3061-cybersecurity guidebook for cyber-physical automotive systems. SAE-Society Automot. Eng.

Daramola, O., Pan, Y., Karpati, P., Sindre, G., 2012. A comparative review of i-based and use case-based security modelling initiatives. Research Challenges in Information Science (RCIS), 2012 Sixth International Conference on. IEEE, pp. 1–12.

Dawes, R.M., 1979. The robust beauty of improper linear models in decision making. Am. Psychol. 34 (7), 571.

Deng, M., Wuyts, K., Scandariato, R., Preneel, B., Joosen, W., 2011. A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. Requir. Eng. 16 (1), 3–32.

Dubois, É., Heymans, P., Mayer, N., Matulevičius, R., 2010. A systematic approach to define the domain of information system security risk management. Intentional Perspectives on Information Systems Engineering. Springer, pp. 289–306.

El Ariss, O., Xu, D., 2011. Modeling security attacks with statecharts. Proceedings of the Joint ACM SIGSOFT Conference–QoSA and ACM SIGSOFT Symposium–ISARCS on Quality of Software Architectures–QoSA and Architecting Critical Systems–ISARCS. ACM, pp. 123–132.

Elahi, G., Yu, E., 2007. A goal oriented approach for modeling and analyzing security trade-offs. Concept. Model.-ER 2007 375–390.

Elahi, G., Yu, E., Zannone, N., 2010. A vulnerability-centric requirements engineering framework: analyzing security attacks, countermeasures, and requirements based on vulnerabilities. Requir. Eng. 15 (1), 41–62.

Encina, C.O., Fernandez, E.B., Monge, A.R., 2014. Threat analysis and misuse patterns of federated inter-cloud systems. Proceedings of the 19th European Conference on Pattern Languages of Programs. ACM, pp. 13.

Excellence in research for australia | australian research council2018. http://www.arc.gov.au/excellence-research-australia. (Accessed on 01/25/2018).

Fabian, B., Gürses, S., Heisel, M., Santen, T., Schmidt, H., 2010. A comparison of security requirements engineering methods. Requir. Eng. 15 (1), 7–40.

Geneiatakis, D., Kounelis, I., Neisse, R., Nai-Fovino, I., Steri, G., Baldini, G., 2017. Security and privacy issues for an iot based smart home. Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2017 40th International Convention on. IEEE, pp. 1292–1297.

Granchelli, G., Cardarelli, M., Di Francesco, P., Malavolta, I., Iovino, L., Di Salle, A., 2017. Towards recovering the software architecture of microservice-based systems. Software Architecture Workshops (ICSAW), 2017 IEEE International Conference on. IEEE, pp. 46–53.

Haley, C., Laney, R., Moffett, J., Nuseibeh, B., 2008. Security requirements engineering: a framework for representation and analysis. IEEE Trans. Software Eng. 34 (1), 133–153.

Haley, C.B., Laney, R.C., Nuseibeh, B., 2004. Deriving security requirements from crosscutting threat descriptions. Proceedings of the 3rd International Conference on Aspect-Oriented Software Development. ACM, pp. 112–121.

Halkidis, S.T., Tsantalis, N., Chatzigeorgiou, A., Stephanides, G., 2008. Architectural risk analysis of software systems based on security patterns. IEEE Trans. Dependable Secure Comput. 5 (3), 129–142.

Hatebur, D., Heisel, M., 2005. Problem frames and architectures for security problems. SAFECOMP. Springer, pp. 390–404.

Hilburn, T., Ardis, M., Johnson, G., Kornecki, A., Mead, N.R., 2013. Software assurance competency model. Technical Report. Carnegie-Mellon Univ. Pittsburgh PA Software Engineering Inst.

IEC, 2006. Fault tree analysis (FTA).

ISO, 2011. Road vehicles – Functional safety.

Jackson, M., 2001. Problem frames: Analysing and structuring software development problems. Addison-Wesley.

Kahneman, D., 2011. Thinking, fast and slow. Macmillan.

Kahneman, D., Klein, G., 2009. Conditions for intuitive expertise: a failure to disagree. Am. Psychol. 64 (6), 515.

Karpati, P., Opdahl, A.L., Sindre, G., 2010. Harm: hacker attack representation method. International Conference on Software and Data Technologies. Springer, pp. 156–175.

Karpati, P., Sindre, G., Opdahl, A., 2010. Visualizing cyber attacks with misuse case maps. Requir. Eng. 262–275.

Kordy, B., Mauw, S., Radomirović, S., Schweitzer, P., 2014. Attack–defense trees. J. Logic Comput. 24 (1), 55–87.

Kriaa, S., Pietre-Cambacedes, L., Bouissou, M., Halgand, Y., 2015. A survey of approaches combining safety and security for industrial control systems. Reliab. Eng. Syst. Saf. 139, 156–178.

Latif, R., Abbas, H., Assar, S., Ali, Q., 2014. Cloud computing risk assessment: a systematic literature review. Future Information Technology. Springer, pp. 285–295.

Li, T., Paja, E., Mylopoulos, J., Horkoff, J., Beckers, K., 2016. Security attack analysis using attack patterns. Research Challenges in Information Science (RCIS), 2016 IEEE Tenth International Conference on. IEEE, pp. 1–13.

Li, Z., Liang, P., Avgeriou, P., 2015. Architectural technical debt identification based on architecture decisions and change scenarios. Software Architecture (WICSA), 2015 12th Working IEEE/IFIP Conference on. IEEE, pp. 65–74.

Lin, L., Nuseibeh, B., Ince, D., Jackson, M., 2004. Using abuse frames to bound the scope of security problems. Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International. IEEE, pp. 354–355.

Liu, L., Yu, E., Mylopoulos, J., 2003. Security and privacy requirements analysis within a social setting. Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International. IEEE, pp. 151–161.

Lund, M.S., Solhaug, B., Stølen, K., 2011. A guided tour of the coras method. Model-Driven Risk Analysis. Springer, pp. 23–43.

Macher, G., Armengaud, E., Brenner, E., Kreiner, C., 2016. A review of threat analysis and risk assessment methods in the automotive context. International Conference on Computer Safety, Reliability, and Security. Springer, pp. 130–141.

Macher, G., Sporer, H., Berlach, R., Armengaud, E., Kreiner, C., 2015. Sahara: a security-aware hazard and risk analysis method. Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015. IEEE, pp. 621–624.

Martin, R.A., 2007. Common weakness enumeration. MITRE Corporation.

Martins, G., Bhatia, S., Koutsoukos, X., Stouffer, K., Tang, C., Candell, R., 2015. Towards a systematic threat modeling approach for cyber-physical systems. Resilience Week (RWS), 2015. IEEE, pp. 1–6.

Mauw, S., Oostdijk, M., 2005. Foundations of attack trees. Icisc. 3935. Springer, pp. 186–198.

Mavropoulos, O., Mouratidis, H., Fish, A., Panaousis, E., 2017. Asto: a tool for security analysis of iot systems. Software Engineering Research, Management and Applications (SERA), 2017 IEEE 15th International Conference on. IEEE, pp. 395–400.

McDermott, J., 2001. Abuse-case-based assurance arguments. Computer Security Applications Conference, 2001. ACSAC 2001. Proceedings 17th Annual. IEEE, pp. 366–374.

McDermott, J., Fox, C., 1999. Using abuse case models for security requirements analysis. Computer Security Applications Conference, 1999.(ACSAC'99) Proceedings. 15th Annual. IEEE, pp. 55–64.

Medvidovic, N., Taylor, R.N., 2010. Software architecture: foundations, theory, and practice. Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering. 2. ACM, pp. 471–472.

Mellado, D., Blanco, C., Sánchez, L.E., Fernández-Medina, E., 2010. A systematic review of security requirements engineering. Comput. Stand. Interfaces 32 (4), 153–165.

Mellado, D., Fernández-Medina, E., Piattini, M., 2007. A common criteria based security requirements engineering process for the development of secure information systems. Comput. Stand. Interfaces 29 (2), 244–253.

Mohan, V., Othmane, L.B., 2016. Secdevops: Is it a marketing buzzword?-mapping research on security in devops. Availability, Reliability and Security (ARES), 2016 11th International Conference on. IEEE, pp. 542–547.

Mohsin, M., Sardar, M.U., Hasan, O., Anwar, Z., 2017. Iotriskanalyzer: a probabilistic model checking based framework for formal risk analytics of the internet of things. IEEE Access.

Mouratidis, H., Giorgini, P., 2007. Secure tropos: a security-oriented extension of the tropos methodology. Int. J. Software Eng. Knowl. Eng. 17 (02), 285–309.

Mouratidis, H., Giorgini, P., Manson, G., 2003. Modelling secure multiagent systems. Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems. ACM, pp. 859–866.

Muñante, D., Chiprianov, V., Gallon, L., Aniorté, P., 2014. A review of security requirements engineering methods with respect to risk analysis and model-driven engineering. International Conference on Availability, Reliability, and Security. Springer, pp. 79–93.

Osbeck, L.M., Held, B.S., 2014. Rational intuition: philosophical roots, scientific investigations. Cambridge University Press.

Owasp, 2018. https://www.owasp.org/index.php/Main_Page. (Accessed on 01/29/2018).

Owasp, 2017. https://www.owasp.org/index.php/Main_Page. (Accessed on 11/09/2017).

Raspotnig, C., Opdahl, A., 2013. Comparing risk identification techniques for safety and security requirements. J. Syst. Softw. 86 (4), 1124–1151.

Riaz, M., Sulayman, M., Naqvi, H., 2009. Architectural decay during continuous software evolution and impact of design for change on software architecture. International Conference on Advanced Software Engineering and Its Applications. Springer, pp. 119–126.

Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. 14 (2), 131.

Saitta, P., Larcom, B., Eddington, M., 2005. Trike v. 1 methodology document [draft]. URL: http://dymaxion.org/trike/Trike_v1_Methodology_Documentdraft.pdf.

Salini, P., Kanmani, S., 2012. Survey and analysis on security requirements engineering. Comput. Electr. Eng. 38 (6), 1785–1797.
Sharma, A., Bawa, R., 2016. A comprehensive approach for agile development method selection and security enhancement. Proc. Int. J. Innov. Eng. Technol. 6, 36–44.
Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.M., 2002. Automated generation and analysis of attack graphs. Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on. IEEE, pp. 273–284.
Sheyner, O., Wing, J., 2003. Tools for generating and analyzing attack graphs. International Symposium on Formal Methods for Components and Objects. Springer, pp. 344–371.
Shostack, A., 2014. Threat modeling: designing for security. John Wiley & Sons.
Siegmund, J., Siegmund, N., Apel, S., 2015. Views on internal and external validity in empirical software engineering. Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on. 1. IEEE, pp. 9–19.
Sindre, G., Opdahl, A.L., 2005. Eliciting security requirements with misuse cases. Requir. Eng. 10 (1), 34–44.
Taylor, F., 1956. Paul e. meehl: clinical versus statistical prediction. a theoretical analysis and a review of the evidence (book review). Int. J. Psychoanal. 37, 490.
Taylor, J., 1997. Introduction to error analysis, the study of uncertainties in physical measurements.
Tøndel, I.A., Jensen, J., Røstad, L., 2010. Combining misuse cases with attack trees and security activity models. Availability, Reliability, and Security, 2010. ARES'10 International Conference on. pp. 438–445.
Torr, P., 2005. Demystifying the threat modeling process. IEEE Secur. Priv. 3 (5), 66–70.
Tuma, K., Scandariato, R., Widman, M., Sandberg, C., 2017. Towards security threats that matter. Computer Security. Springer, pp. 47–62.
UcedaVelez, T., Morana, M.M., 2015. Risk centric threat modeling: process for attack simulation and threat analysis. John Wiley & Sons.
Ur Rahman, A.A., Williams, L., 2016. Security practices in devops. Proceedings of the Symposium and Bootcamp on the Science of Security. pp. 109–111.
Van Lamsweerde, A., 2004. Elaborating security requirements by construction of intentional anti-models. Proceedings of the 26th International Conference on Software Engineering. IEEE Computer Society, pp. 148–157.
Van Lamsweerde, A., Letier, E., 2000. Handling obstacles in goal-oriented requirements engineering. IEEE Trans. Software Eng. 26 (10), 978–1005.
Van Lamsweerde, A., et al., 2007. Engineering requirements for system reliability and security. NATO Secur. Sci. Series D-Inf. Comm. Secur. 9, 196.
Wang, P., Chao, K.-M., Lo, C.-C., Wang, Y.-S., 2017. Using ontologies to perform threat analysis and develop defensive strategies for mobile security. Inf. Technol. Manag. 18 (1), 1–25.
Whittle, J., Wijesekera, D., Hartong, M., 2008. Executable misuse cases for modeling security concerns. Software Engineering, 2008. ICSE'08. ACM/IEEE 30th International Conference on. IEEE, pp. 121–130.
Williams, I., 2015. Evaluating a Method to Develop and Rank Abuse Cases based on Threat Modeling, Attack Patterns and Common Weakness Enumeration. North Carolina Agricultural and Technical State University Ph.D. thesis.
Wohlin, C., 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. ACM, pp. 38.
Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., 2012. Experimentation in software engineering. Springer Science & Business Media.
Xu, D., Nygard, K.E., 2006. Threat-driven modeling and verification of secure software using aspect-oriented petri nets. IEEE Trans. Software Eng. 32 (4), 265–278.
Yuan, X., Nuakoh, E.B., Williams, I., Yu, H., 2015. Developing abuse cases based on threat modeling and attack patterns. JSW 10 (4), 491–498.

Katja Tuma is a Ph.D. student in software engineering at the Department of Computer Science and Engineering in Gothenburg, Sweden. Katja received her prior education in Computer and Information Science at the University of Ljubljana, Slovenia. Her main research interests are in the area of secure software design, particularly focusing on the precision and efficiency of analyzing and mitigating security threats in the early design phases.

Gul Calikli holds Ph.D. and M.Sc. degrees in Computer Engineering and a BS degree in Mechanical Engineering from Bogazici University. She is an Assistant Professor at the Department of Computer Science and Engineering, which is shared between Chalmers University of Technology and University of Gothenburg. Previously, she was a Post-Doctoral Research Fellow at The Open University, in the United Kingdom and a Post-Doctoral Research Fellow at Ryerson University, in Canada. Her research interests are empirical software engineering for privacy and human aspects in software engineering. She is a member of IEEE Computer Society and ACM.

Riccardo Scandariato received his Ph.D. in Computer Science in 2004 from Politecnico di Torino, Italy. In 2014, he moved to Gothenburg (Sweden) and joined the Department of Computer Science and Engineering, which is shared between the Chalmers University of Technology and the University of Gothenburg. Riccardo's main research interests are in the area of secure software engineering, with a particular focus on privacy & security by design and empirical methods for security. He has published over 75 papers in the area of security and software engineering. He is an Associate Editor of the International Journal of Secure Software Engineering.