Finding Security Threats That Matter: Two Industrial Case Studies

Katja Tuma[a,*], Christian Sandberg[b], Urban Thorsson[b], Mathias Widman[b], Thomas Herpel[c] and Riccardo Scandariato[d]

[a] *University of Gothenburg | Chalmers, Sweden*
[b] *Volvo, Sweden*
[c] *Zukunft Mobility, Germany*
[d] *Hamburg University of Technology, Germany*

ARTICLE INFO

Keywords:
Threat Analysis, Risk, STRIDE, Case Study, Empirical Software Engineering, Security Deskilling

ABSTRACT

In the past decade, speed has become an essential trait of software development (e.g., agile, continuous integration, DevOps) and any inefficiency is considered unaffordable time waster. Such a fast pace causes challenges for architectural threat analysis. Leading techniques for threat analysis, like STRIDE, have the advantage of being systematic. However, they are not equipped to discern between important and less critical threats, *while* the threats are being discovered. Consequently, many threats are discarded at a later time, when their risk value is assessed. An alternative technique, called eSTRIDE, promises to remove these inefficiencies by focusing the analysis on the critical parts of the architecture. Yet, no empirical evidence exists about the actual effect of trading off systematicity, for a more focused attention on high-priority threats. This paper contributes with an empirical study comparing these two approaches in the context of two industrial case studies. We found that the two approaches yield the same number of security threats during a given time frame. However, participants using eSTRIDE found twice as many high-priority threats. The underlying analysis procedures cause similarities and differences in the execution. In addition, security expertise has an effect (albeit small) on the quality of analysis outcomes and execution.

1. Introduction

Security-by-design techniques aim to avoid security pitfalls in software systems early on, starting from the design phase, when the major development effort is yet to come [22, 13]. The intent is to cut the maintenance cost induced by fixing security design flaws when it is too late. In this context, architectural threat analysis is a common activity performed by experts (often manually) to analyze the high-level design of a system for potential security issues related to its assets of interest [32]. These threat analysis techniques are routinely used in application domains where upfront design is still dominant. For instance, in safety-critical systems like automotive. Microsoft's STRIDE is a well-known threat analysis technique that is also used in the automotive domain [29, 15]. This technique has the tendency to lead to the discovery of a high volume of potential threats [33, 28]. After the discovery phase, threats are ranked according to their risk value, which is a combination of impact and likelihood. As a result, many threats are later-on discarded due to their low risk value, even though significant time went into their discovery. This is a clear element of inefficiency, which is common to these family of so-called *risk-last* approaches, where risk is considered only *after* the threats have been found.

In the past decade, speed has become an essential trait of software development (e.g., agile, continuous integration, DevOps, etc.) and any up-front inefficiency is considered unaffordable time waster. Consequently, such a fast pace causes challenges for threat analysis [8]. Further, security expertise is scarce in organizations and the time available from experts needs to be used in an optimal way. Driven by these observations, in prior work we have defined eSTRIDE, a *risk-first* threat analysis approach [34]. The core idea is to enrich the analyzed model with risk-related information, so that the analysis activity is more focused on the parts of the architecture where assets with high-priority security objectives (e.g., confidentiality) are located. This would lead to the early discovery of high-priority security threats, hence by-passing threat prioritization all together. This seems promising for organizations where development speed is key to surpassing the competition. Yet, no empirical evidence exists about the actual effect of trading off a bit of systematicity (a benefit of STRIDE) for a more focused attention on high-priority threats (the goal of eSTRIDE). Could high-priority threats go unnoticed? Are high-priority threats being discovered faster? These and similar questions beg for an answer.

The purpose of this study is to gather empirical evidence about the similarities and differences between a risk-last (STRIDE) and risk-first (eSTRIDE) threat analysis technique (see Section 2) in an industrial setting. To this aim, we conduct two case studies (see Section 3) with industrial participants (15 in total) from two automotive organizations located in different countries. Within each organization, we observe and compare two teams analyzing the same system, where each team uses one of the two mentioned techniques.

The contributions of this work are three-fold. First, we try to understand how to optimize time and effort spent on

*Corresponding author
katja.tuma@cse.gu.se (K. Tuma); christian.sandberg@volvo.com (C. Sandberg); urban.thorsson@volvo.com (U. Thorsson); mathias.widman@volvo.com (M. Widman); thomas.herpel@zf.com (T. Herpel); riccardo.scandariato@tuhh.de (R. Scandariato)
ORCID(s):

Table 1
Activities of STRIDE and eSTRIDE

Step	Activity	STRIDE	eSTRIDE
Building diagram	Scope discussion	✓	✓
	Drawing the DFD	✓	✓
	Model abstraction/refinement	✓	✓
	Asset analysis		✓
	Extending the DFD		✓
Analyzing diagram	Diagram exploration	Element by element	Scenario-based
	Threat types considered	Mapping table	Pruned mapping table
	Attack scenario development	✓	✓
	Threat feasibility discussion	✓	✓
	Threat reduction	✓	✓
	Threat consequences and prioritization	✓	

performing threat analysis. To this aim, we study the productivity levels of a risk-first analysis and the discovered high-priority threats. Second, we carry out a qualitative and quantitative comparison of how the two techniques are performed, e.g., by looking at what activities are more prevalent. This allows to identify key insights into the way teams work when they use the two techniques and, accordingly, gauge the potential for optimization. Third, we investigate the effect of security expertise on the use of both techniques. As security expertise is a scarce commodity, it is interesting to study whether less skilled teams could produce acceptable results with any of the two techniques.

The results of this study (Section 4) show no differences in productivity and timeliness of discovering high-priority security threats. But, we find differences in analysis execution. Specifically, participants using the risk-first technique found *twice as many* high-priority threats, developed detailed attack scenarios, and discussed threat feasibility in greater detail. In comparison, participants using the risk-last technique found more medium and low-priority threats. In addition, we find that security expertise has an effect (albeit small) on the quality of analysis outcomes and analysis execution. The results are further discussed in Section 5 and their validity is reproached in Section 7. We contextualize our results with respect to the related literature in Section 6, and present our conclusions in Section 8.

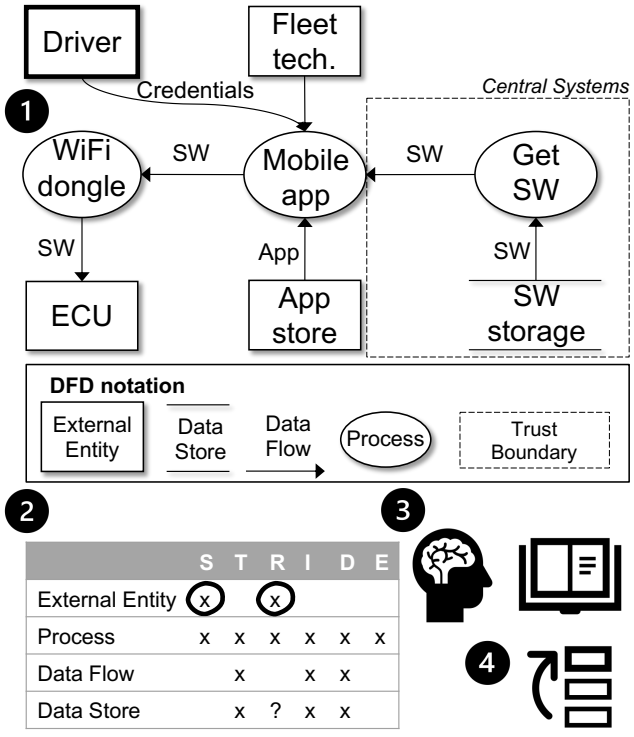


Figure 1: STRIDE in action

2. The Compared Techniques

STRIDE – STRIDE is a family of techniques developed by Microsoft to help identify threats (e.g., potential attack scenarios) that software systems are exposed to, especially because of design-level flaws. The name itself is an acronym that stands for the threat categories of Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege. For the definition of threat categories, we refer the reader to the documentation of STRIDE [29]. In particular, this study considers the 'STRIDE per element' variant (hereafter STRIDE).

Table 1 summarizes the activities performed during the analysis with STRIDE. It is a model-based technique which starts with the creation of a graphical representation of the software system under analysis, namely as a Data Flow Diagram (DFD). Figure 1 shows a simplified DFD and how it is used during STRIDE. We depict an industrial case from the automotive context (also under analysis in this work) for a firmware update of an Electronic control unit (ECU) in a truck. On a high-level, the driver (or technician) connects their mobile device to a WiFi dongle in their vehicle, logs into the Mobile app (installed on their device), gets the software from a remote software repository and installs it on the ECU of their vehicle. Creating the DFD typically involves

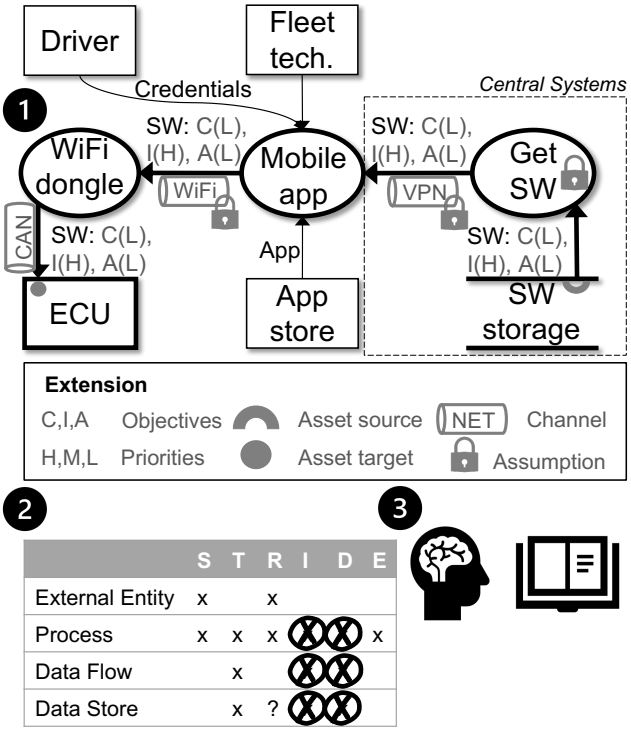


Figure 2: eSTRIDE in action

discussing the scope of the analysis (i.e., defining the breadth of analysis), drawing the diagram (e.g., on a board), abstracting or refining the model (hence defining the granularity of the analysis), and so on. A DFD represents how information moves across a software system and consists of processes (active entities, like *'Get SW'* in Figure 1), data flows (exchanged info), external entities (e.g., *'Driver'*), data stores (e.g., *'SW storage'*), and, optionally, trust boundaries (e.g., *'Central Systems'*).

The second phase in STRIDE is the systematic exploration of the DFD to identify the threats. It involves several activities (see Table 1) which can be roughly grouped into four steps (steps 1-4 in Figure 1).

First, the analysts explore the diagram one element at the time (step 1). For each element type, STRIDE suggests to look into specific threat categories, which is given by the so called threat-to-element mapping table (step 2 in Figure 1). For instance, for external entities (e.g., *'Driver'*) the analysts are suggested to only look for spoofing and repudiation threats. Next, the analysts engage in a brainstorm to develop concrete attack scenarios, during which they can use exemplary threats (of each category) for inspiration (step 3). Each discovered threat must be discussed with respect to its feasibility in order to determine its relevance. The relevant treats and discovered attack scenarios are immediately documented. Sometimes, the same threats can be present at multiple locations in the diagram (e.g., an information disclosure threat on all data flows that are not encrypted). In such situations, analysts can apply the *threat reduction* technique and continue with the diagram exploration. This technique essentially allows copying the identified attack scenarios to all the said locations in the report. After the diagram has been explored the threats are prioritized according to their risk value (step 4). As the risk assessment happens *at the end*, STRIDE is a 'risk-last' threat analysis technique. An unfortunate consequence of this approach is that the effort spent identifying low-risk threats could be a potential time-waster and a source of lower productivity.

eSTRIDE – The second technique we study is the Extended STRIDE (hereafter eSTRIDE), which is an example of a 'risk-first' technique [34]. eSTRIDE is similar to STRIDE, however the analysis focuses on parts of the architecture that are related to assets with (at least one) high-priority objective and is limited to the threat categories that are relevant for those assets. Therefore, eSTRIDE makes the implicit assumption that high-priority threats are related with the above-mentioned parts of the architecture[1]. As shown in the right-hand side of Table 1, the analysis technique is similar to STRIDE, with a few substantial differences. In eSTRIDE, an asset analysis is performed *at the beginning*, during the model building phase. This includes the identification of assets, their security objectives (e.g., confidentiality should be preserved) and their priority (e.g., high confidentiality). This security relevant information is inserted in an extended diagram, or eDFD. Figure 2 shows an eDFD (of the same system as shown before) and how it is used. For brevity, we have omitted the accountability objective (effected by the threat of repudiation) from the figure. The eDFD also carries information about the source and sink of each asset (i.e., an information scenario), the type of communication channels in the system, and domain assumptions that are relevant to the security analysis (e.g., existing security solutions). To save time, this additional information is only added to elements involved in the transfer (or storage) of high-priority assets (as observed in Figure 2).

The second phase of eSTRIDE is a guided exploration of the eDFD (steps 1-3 in Figure 2). Instead of visiting each element in the graph (as is done in STRIDE), the analysts consider end-to-end information scenarios (i.e., paths from source to sink, such as the path of the *'SW'* from the *'SW Storage'* all the way to the *'ECU'*) of assets with at least one high-priority objective (step 1).

Further, in step 2 the mapping table used in STRIDE is pruned before threats are identified. In particular, the categories of threats that do not relate to the high-priority objectives are discarded. For instance, in Figure 2 the only security objective with a high priority is integrity, thus information disclosure (which directly effects confidentiality) and denial of service (effecting availability) threats can be skipped. Spoofing and elevation of privilege threats can not be skipped so easily, as they can potentially threaten all the security objectives. However, the analysts may make domain assumptions about existing security solutions. For example, they may assume a mutually authenticated and end-to-end encrypted channel (*'VPN'* in Figure 2) between the

[1] This paper does not investigate this assumption. However, in our case studies, we found no evidence contradicting it.

components of the *'Central Systems'* and the *'Mobile app'*. Therefore, spoofing the process *'Get SW'* to distribute malicious updates could be skipped in the analysis. In this case, spoofing the *'ECU'* or the *'WiFi dongle'* must still be considered.

Similar to STRIDE, attack scenarios are built and discussed for feasibility with respect to the domain, and threat reduction can be applied (step 3). However, using eSTRIDE the analysts are equipped with more information to discuss the feasibility (e.g., existing security solutions). The expected benefit is a reduced effort due to bypassing threat prioritization (step 4 in STRIDE) and investigating fewer threats.

3. Design of the Study

We conduct two case studies where we compare STRIDE to the eSTRIDE. In what follows, we present the research questions, industrial cases used in this study, and the participants. We also describe the task performed by the participants, the on-site workshops, and the data collection methods.

3.1. Research Questions

This study answers research questions regarding the differences in the analysis outcomes (RQ1, RQ2, RQ4) and execution (RQ3, RQ4) for the studied techniques.

RQ1. *What are the differences between a risk-last and a risk-first analysis technique in terms of productivity?*
Risk-last threat analysis prioritizes threats at the end of the analysis procedure. In contrast, risk-first analysis aims to bypass threat prioritization by analyzing the high risks first, at the cost of a more extensive modeling phase. The purpose of the first research question is to understand whether the extra up-front effort has an impact on the productivity, measured as the amount of correctly identified threats per time unit.

RQ2. *What are the differences between a risk-last and a risk-first analysis technique with respect to the timeliness and amount of discovered high-priority threats?*
In realistic circumstances, threat analysis sessions are pressed for time. Achieving complete coverage with a manual analysis is challenging in this context. Therefore, threats are often overlooked [33, 28]. It seems reasonable to 'knowingly' overlook low-priority threats as compared to high-priority threats. The purpose of the second research question is to investigate whether risk-first analysis produces high-priority threats faster (and in a larger quantity) when compared to the risk-last analysis technique.

RQ3. *What are the differences between a risk-last and a risk-first analysis technique with respect to both the timeliness and the amount of activities as well as activity patterns?*
Apart from the activities in Table 1, important events and support activities take place during a threat analysis session. For instance, updating the diagram, or making an assumption. Support activities include pointing at the board, taking a break, documenting, referring to case documentation, etc. Due to the repetitive nature of manual threat analysis, these activities tend to re-occur. We are interested to investigate which activities appear more often or sooner, and how that differs for the two techniques. In addition, we observe combinations of activities, or activity patterns to understand which technique better facilitates constructive thinking. Therefore, the purpose of the third research question is to investigate the differences in the 'way of working' for the studied techniques.

RQ4. *What is the effect of the security expertise of the participants on the outcomes and execution of a risk-first and risk-last analysis technique?*
Previous studies paint a picture of the current security activities, skills [27] and threat analysis practices [5, 8] in agile organizations. In [5] three (out of four) interviewed companies revealed that developers are already involved in threat analysis. Further, in two organizations [5] they are even responsible for identifying threats, but still need input (i.e., from security managers, or consultants) when it comes to asset and risk analysis. Considering the fairly acceptable performance measured for STRIDE in the academic setting [28, 33], we are interested to study the effect of security expertise on the quality of analysis outcomes and technique execution for both techniques.

3.2. Industrial Partners

ORG A The study was first executed within a multinational automotive organization with over 100 000 employees worldwide. The core activity of ORG A is the production, distribution and sale of trucks, buses, and construction equipment. This multinational has established security practices in the development life-cycle and performs STRIDE-like threat analysis on a daily basis. From a security standpoint, participants from ORG A can be regarded as experts.

ORG B The study was replicated within a younger (and smaller in terms of number of employees) organization based in a different country. This organization is specializing in the development and testing of autonomous driving solutions. In comparison, development teams in ORG B are much smaller and more cross functional. In addition, threat analysis is not performed routinely within ORG B. From a security standpoint, participants from ORG B can be regarded as novices. We have gathered information about participants background and roles within the organizations with an entry questionnaire (discussed in more detail in Section 4.4). In summary, the participants of ORG A perceive themselves as security experts (or have previously undergone security training), while the participants in ORG B regard themselves as security novices.

3.3. Industrial Cases

We ask each company to identify a software system they want to analyze from a security standpoint. Each of our industrial collaborators put together a document describing their system, including textual specifications and technical diagrams. We provided feedback on the documentation in order to guarantee that it was clear and contained enough information. In what follows we briefly describe the cases but omit details due to confidentiality concerns. Both cases are

from the automotive domain, deal with (safety critical) embedded software and are comparable in size and complexity. As a term of measure, the diagrams created by the teams in ORG A contained on average 42 elements, compared to 56 in ORG B.

ORG A– ECU update. The analyzed industrial case is about the firmware update of an Electronic Control Unit (ECU) in a truck. The update can be performed by an authorized party (e.g., the driver who wants to change the speed limiter when crossing countries) via a mobile app and without visiting the workshop. In the analyzed scenario, the driver (or technician) connects their mobile device to the WiFi dongle of the vehicle to update the ECUs. Next, they can use the mobile application to (i) configure (if properly authorized) certain ECU parameters, or (ii) download the ECU software updates from a remote software repository (owned by the OEM) and install them on the ECUs of their vehicle. This case is documented as a box-and-arrow reference architecture and a handful of pages containing text.

ORG B– HIL testing. The analyzed system is a simulator with hardware in the loop (HIL). The platform allows the execution of automated tests of autonomous driving components. In the analyzed scenario, an authenticated test operator can schedule performance tests on a piece of embedded software. The component is rigged to the system, which provides simulated sensor data and collects performance measurements. The platform executes the appropriate test cases and provides the obtained measurements to the test operator (while also storing them in a private cloud).

3.4. Participants

In each organization, we divided the participants in two teams, with each performing the threat analysis of the same case with a different technique. Within each organization, the two teams had similar size and comparable expertise. A thorough discussion of the seniority level and security expertise of the participants (across organizations) can be found in Section 4.4.

ORG A The participants are industrial experts with some experience in threat analysis. We assembled two teams with 3 (STRIDE) and 4 (ESTRIDE) members. The ESTRIDE team had an additional member, a threat analysis trainee. Each team member had an assigned role (process enforcer, security expert, and domain expert) according to their expertise. The roles were assigned to reflect how threat analysis sessions were performed within the organization. The role of the process enforcer is to ensure that the team was performing the analysis in accordance with the prescribed procedure, and that the discussions (typically about attack feasibility) do not loop or stray to unrelated topics. Security experts take the lead in suggesting attack scenarios, and the role of the domain experts is to describe technical details required to contextualize the attack to the system under analysis. The trainee in ESTRIDE was assigned the role of a security expert, given the background of the participant.

ORG B The participants are industrial experts with deep knowledge of the case but with no prior threat analysis experience and little security background. We assembled two teams with 3 members each. Differently from the other company, there was no strict role separations among members of the groups. This is in line with the way of working at the company. All members played a mix of both domain and security expert. Additionally, each team had one student member (doing an industrial MSc thesis at the company) that joined as observer. The students did not contribute to the work of the teams.

Supervision of participants. The experimenters were present in all analysis sessions for two reasons. First, they monitored the participants to ensure that the analysis was indeed performed according the instructed procedure. Second, they were taking notes and making observations, which were used to support the data analysis afterwards. We remark that the experimenters strictly refrained from influencing the analysis in any respect and did not contribute to the discussion. In ORG A, the experimenters joined purely as observers. As the teams in ORG B were inexperienced with STRIDE, the experimenters also answered procedural questions.

3.5. Task

In both organizations, the two teams were presented with the same task: perform a threat analysis on the industrial case using the prescribed technique. Both teams were asked to (1) build a diagram based on the architectural documentation, and (2) analyze the diagram according to the assigned technique.

Building. The type of diagram to be built differed across teams: a DFD for the STRIDE teams and an eDFD for the ESTRIDE teams. As described in Section 2, we remind that eDFD are richer models that require, among other things, more in-depth thinking about the assets. During this phase, the participants resorted to their domain knowledge and the available system documentation.

Extending (only ESTRIDE). The ESTRIDE teams were asked to assign priorities to the security objectives of assets before analyzing the diagram.

Analyzing. The second phase required a discovery of as many threats as possible given the available time. The STRIDE teams performed a systematic, element by element exploration of the diagram. In contrast, the ESTRIDE teams performed a guided analysis of each end-to-end scenario containing (at least one) high-priority asset objective (see Section 2).

Prioritizing (only STRIDE). At the end, the STRIDE teams were asked to assign priorities (i.e., risk values) to the threats they had identified. These self-reported priorities are not directly used in the study, as the experimenters made their own assessment of the priorities. However, having this activity makes the comparison of the effort between the two techniques more fair, as this is part of STRIDE and it balances the extension activity in ESTRIDE.

Reporting. As soon as a threat was discovered, it was documented in a report, which had to be submitted electronically at the end. The reports contained a list of the iden-

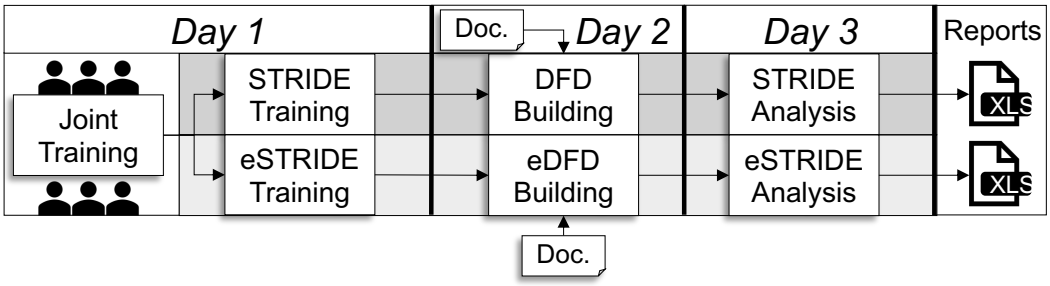


Figure 3: The execution of the study in both organizations

tified threats, their locations in the diagram, attack scenarios exemplifying the threats. The participants were given a template for the report, in the form of a spreadsheet. The purpose of the template was to simplify and standardize our analysis of the results.

3.6. Execution of the Study

Figure 3 depicts the execution of the study in both organizations. The study was split into sessions (3 hours per day in ORG A and about 5 hours per day in ORG B) taking place on site on three separate days during the same week. The authors supervised all sessions. All the material (i.e., documentation of the industrial case, training material, description of task, report template) was shared with the participants a week in advance. They were strongly encouraged to read all the material (about 20 pages) before the start of the study.

Day 1: Training. The teams were separated and specifically trained to accomplish their task. The training consisted of brushing up basic security concepts (Joint Training in Figure 3), understand how to build DFDs (or eDFDs), and learning how to perform STRIDE (or eSTRIDE) [2]. The training sessions contained several hands on exercises for the participants. Due to their limited security expertise, in ORG B we planed a longer training session (ORG B: 5h vs ORG A: 3h), in which we elaborated more on the security concepts, including security threats and countermeasures.

Day 2: Building diagram. On the second day, the teams were given printed copies of all the material and started worked on the task (see Section 3.5). Once the participants finished building the diagram, they were allowed to continue with the diagram analysis. In proportion to the complexity of the system under analysis, we allotted only 3 hours per day to simulate realistic time constraints in ORG A. We intentionally relaxed this constraint in ORG B (5 hours) to avoid overloading the less security experienced group of participants and putting them under stress.

Day 3: Analyzing the diagram. On the third day, the teams were given the same printed material, and the diagram they had created the previous day. They continued where they left off until they finished with the entire task. All the teams finished in the allotted time, and some even finished early (namely, the STRIDE team in ORG A and the ESTRIDE team in ORG B).

[2] In ORG A, we had a separate, short training session with the process enforcers to remind them to monitor the progress and speed up the discussion, if necessary.

Table 2
Codes used to mark threat analysis activities and events. Codes for events are marked by the † symbol

Activity groups	Coded activities and events
Building diagram	Drawing on the board
	Architecture abstraction/refinement
	Asset analysis
	Extending the diagram
	Focusing on assets with high-priority objectives
	Scope discussion
	Making an assumption†
Analyzing diagram	**Attack scenario development**
	Domain discussion
	Threat feasibility
	Threat consequence
	Threat prioritization
	Threat reduction
	Using assumption†
	Updating diagram†
	High-priority threat found†
	Low or Medium-priority threat found†
Support activities	Pointing at board†
	Referring to task description†
	Referring to assumptions†
	Referring to case document†
	Referring to training material†
	Break†
	Unsure
	Documenting
Detour	Chatting
	Difference in opinion
	Terminology

3.7. Qualitative Measures

We have collected voice recordings of analysis sessions in ORG A and took detailed hand notes of those sessions in ORG B. This material has been analyzed to answer RQ3.

ORG A The tape recordings have been manually tran-

scribed by the first author using dedicated software[3]. The manual transcription process helped the experimenters gain a deeper understanding of the recorded material. After having a thorough understanding of the recordings, the first author *coded* the transcriptions. Coding is a technique for systematically marking chunks of transcriptions. The analysis of code occurrences reveals trends and supports a qualitative analysis. Table 2 depicts the hierarchy of codes we used. We coded activities and events related to *diagram building*, *analysis of diagram*, *support activities*, and *detours*.

Activities are durable actions of participants, such as drawing on the board and architecture abstraction and refinement, which occurred during the diagram building phase. Regarding diagram analysis, we coded the activities of attack scenario development, threat feasibility discussion, threat consequence, and the like. We also coded detour activities (e.g., terminology discussion) and support activities (e.g., pointing on the board) to better understand activity patterns.

On the other hand, events (marked with † in Table 2) are instantaneous participant actions. For instance, while discussing about the scope of the analysis the participants may have made an assumption about trusting an external entity, which they documented immediately. Certain events were only possible to code after having assessed the reports. In particular, the event of correctly discovering a high-priority threat could be coded in the transcriptions after having assessed the threats and their priorities. Therefore, we revisited the transcriptions to manually insert such codes.

ORG B Recording of the analysis sessions was not allowed by ORG B. Therefore, the experimenters took detailed notes about the activities occurring during the sessions, including timestamps of such activities and events. These notes have been labeled, using a subset of codes in Table 2 (see codes in bold). We could not use all the codes because it would be impossible to for the experimenters to reliably keep track of that many activities in their note. Nevertheless, we believe that the subset of the codes is representative and did not have an impact on the quality of the analysis.

Interviews. To answer RQ4, we have organized short interviews (20 minutes for each team) at the end of the third day. The semi-structured interviews contained a prepared list of questions (6), which helped us better understand their background knowledge and experience. Namely, we inquired about their perceived progress, the challenges they experienced, and asked them to explain how they approached the task at hand (e.g., What was your strategy for visiting the diagram and why?). The collected information (in the form of notes) was used as a complement to the questionnaire (see below) to better understand their background knowledge and experience.

3.8. Quantitative Measures

Questionnaire. To answer RQ4, we designed a brief entry questionnaire. We used the entry questionnaire to collect the background of participants, in terms of the years of professional experience, their familiarity with security (on a scale with 4 levels), and their prior experience with threat analysis (on a scale with 4 levels).

Reports. To answer RQ1 and RQ2, we assessed the reports handed in by our participants. The hand-ins included pictures of the created diagrams (DFD or eDFD), a list of security assumptions made during the analysis, and a list of identified threats (documented according to the provided template). The reported threats have been assessed and marked as correct (true positives) or incorrect (false positives) by the first author. For the cases where the assessor was not feeling fully confident (5 to 10 threats in each organization), we had a discussion with the industrial experts (in both organizations) in order to come to an agreement. This also acted as a form of quality control for the assessment made.

A true positive (TP) is a correctly identified threat. This means that: (a) the participants found the threat in the correct diagram location, (b) the participants found a realistic attack scenario for the security threat or the participants found a security vulnerability, and (c) the threat is correct with respect to the assumptions the participants made.

A false positive (FP) is an incorrectly identified security threat. This means that the participants found the security threat in the wrong location or the threat is not correct with respect to the assumptions the participants have made. Additionally, some reported threats had insufficient information for us to assess them. In these cases, we followed a strict approach and marked them as false positive as well.

With the above measures, we compute the aggregate indicators of precision ($P = \frac{TP}{TP+FP}$) as the ratio between correctly identified threats and all reported threats.

We measured the time it took for participants to complete the task. In particular, we measured the time it took for participants to complete the phases of the task. For STRIDE these were (1) building the diagram, (2) analyzing the diagram, and (3) prioritizing the threats. Similar, for ESTRIDE the phases were (1) building the diagram, (2) extending the diagram, and (3) analyzing the diagram. We do not include coffee breaks in the measured time. Accordingly, we compute productivity ($Prod = \frac{TP}{TotalTime}$) as the amount of correctly identified threats per hour.

In both STRIDE and ESTRIDE, the experimenters assessed the priority of the reported threats, validated the priorities with the industrial partners (for the cases that were less clear-cut), and used them as the ground truth in this study. There were 2 disagreements (in ORG B), which were resolved by consensus.

In the case of STRIDE, we have the opportunity to directly compare the reported vs assessed threat priorities. This goes beyond the objectives of this work and is not further discussed later on. However, we have observed that in ORG A 73% of the reported threat priorities were correct (i.e., they were assigned the same priority by both the participants and experimenters). In particular, there was a very good agreement on the low priorities, while in other disagreements the priorities were raised by the experimenters (20% of the reported priorities), mostly from

[3] https://www.qsrinternational.com/nvivo/home

Table 3
Correct (TP) and incorrect (FP) threats reported by the teams

	Org A			Org B		
	Stride	eStride	Common	Stride	eStride	Common
TP	12	13	6	40	32	14
FP	15	0	-	12	14	-
Precision (%)	0.4	1		0.8	0.7	

medium to high. Therefore, our participants with security background were very good at identifying low priority objectives, whereas they sometimes misidentified high priorities as medium. In contrast, in Org B only 32% of the reported threat priorities were assigned the same priority levels by experimenters. In addition, half of the reported priorities were lowered (i.e., from high to medium). This large discrepancy can be explained by the lesser security expertise of the two teams in Org B.

In case of eStride, the participants did not prioritize the threats they reported and therefore we cannot make any direct comparison.

3.9. Additional Quantitative Measures in Org A

To answer RQ3, we also analyzed the transcribed recordings. As mentioned before, coding the transcriptions enabled us to track the exact location of a particular activity or event in the transcript (e.g., position index in the text where the activity starts). Therefore, we analyzed the locations of the codes in the transcriptions and the spatial distance between them. The spatial distance between codes is a proxy measure of time distance between activities. We used the distance in the text instead of the actual time distance for convenience reasons, as the transcription software did not provide timestamps for the transcribed text. However, the spatial distance has some advantages. For instance, it is insensitive to a pause in the conversation. The *normalized average distance between codes* was measured as the average number of characters separating the (starting indexes of the) occurrences of each two codes, normalized to the total length (in characters) of the transcription.

4. Results

For each research question, this section reports the results of the objective analysis of the collected data. We further discuss and answer each research question in Section 5.

Before we dwell on the RQs, Table 3 reports the observed levels of true positives and false positives in each team (Stride vs eStride) of the two organizations (Org A vs Org B). The table also shows the threats that are in common across each pair of teams (this is further discussed in Section 5. Within organizations, the number of correctly reported threats (TPs) is similar (Org A: 12 vs 13 and Org B: 40 vs 32). In Org B the amount of mistakes is similar between treatments (Stride: 12 vs eStride: 14). Therefore,

Table 4
RQ1. Productivity

	Org A		Org B	
	Stride	eStride	Stride	eStride
Building DFD (h)	1h10	0h15	1h55	1h05
Extending DFD (h)	-	2h20	-	1h15
Analysis (h)	2h20	2h35	4h15	3h25
Prioritization (h)	0h20	-	0h50	-
Total time (h)	3h50	5h10	7h00	5h45
Productivity (TP/h)	3	2.6	5.7	5.6

the precision of the teams in Org B is similar (Stride: 0.8 vs. eStride: 0.7). However, in Org A the Stride team reported 15 incorrect (FPs) threats while the eStride team reported none. Most of the incorrect threats of this team (13 out of 15) were marked as such due to not having sufficient information for their assessment (see Section 3.8). This is reflected in the measured precision (Stride: 0.4 vs eStride: 1).

The results concerning the precision are inconclusive due to the strict assessment in Org A (i.e., threats not having sufficient information are marked as incorrect), but, otherwise, we have not observed major differences between the two techniques.

4.1. RQ1: Productivity of Teams

As shown in Table 4 (bottom line), the productivity levels are pretty similar across the two techniques. As discussed later in Section 4.4, there are some differences in productivity between the two organizations.

We have looked into the time it took for each team to accomplish the different parts of the given task. Across organizations, it took the eStride team less time to build the the initial DFD diagram (Org A: 0h15 vs 1h10 and Org B: 1h05 vs 1h55). However, the eStride teams had to put in significant extra time to extend the diagrams with the necessary security information (see Section 2). In Org A the time spent on analysing the diagrams and identifying the threats was similar across teams, while in Org B the eStride team spent about 1h less on this task. Finally, the Stride teams had to put in additional effort to prioritize the identified threats at the end (this activity is not necessary in eStride).

4.2. RQ2. Discovering High-Priority Threats

As shown in Table 5, in both organizations, the eStride teams have found more high-priority threats (Org A: 62% vs 33% and Org B 47% vs 15%). Only a part of the discovered threats were common, therefore we have observed that eStride is inclined to produce more high-priority threats compared to Stride.

Figure 4 depicts when the teams discovered high-priority threats in Org A[4]. We have manually inspected the recordings to recover the time-stamps of the discovered high-

[4]This data in not available in Org B because we were not allowed to tape the sessions.

Table 5
RQ2. The correct threats (TP) are broken down into high (H), medium (M) and low (L) priority.

	Org A			Org B		
TP	STRIDE	eSTRIDE	COMMON	STRIDE	eSTRIDE	COMMON
H	4 (33%)	8 (62%)	4	6 (15%)	15 (47%)	4
M	2 (17%)	1 (1%)	-	22 (55%)	10 (31%)	4
L	6 (50%)	4 (37%)	2	12 (30%)	7 (22%)	6
TOTAL	12	13	6	40	32	14

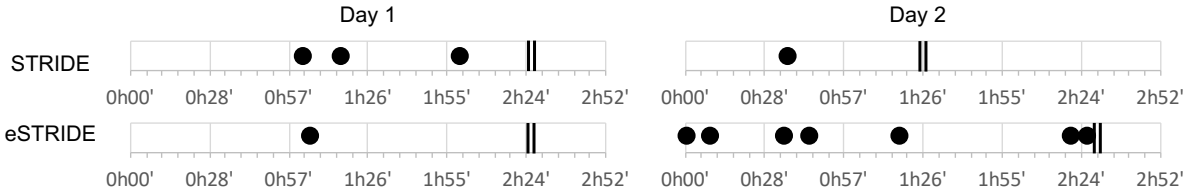


Figure 4: RQ2. High-priority threats (dots) discovered by the STRIDE (top) and eSTRIDE (bottom) team in ORG A. The end of each session is marked with a double vertical bar.

priority threats due to the importance of this code. The STRIDE team started analyzing the diagram one hour into the first day and discovered most high-priority threats during the first day. The last high-priority threat was discovered by the STRIDE team about 40 minutes into the second day. The eSTRIDE team started analyzing the diagram on the second day, hence they discovered most high-priority threats during the second day. Yet, they discovered one high-priority threat already during the first day while discussing security objectives of assets and their priorities.

Compared to the STRIDE team, the eSTRIDE team did not find high-priority threats faster.

Although we do not have precise measurements, we have informally observed a similar trend regarding the discovery time of high-priority threats in ORG B. In addition, we remark that many high-priority threats were found around the trust boundaries (in both teams), therefore the strategy of visiting the diagram may have an impact on the timely discovery of high threats.

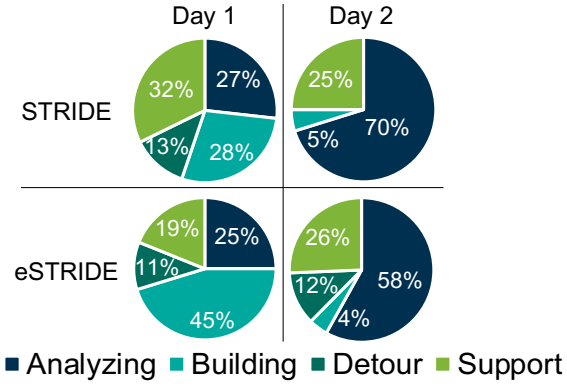
4.3. RQ3. Focus and Activity Patterns

First, we report on the activity focus in both organizations. Then we report the timelines of activities, and activity patterns for teams in ORG A.

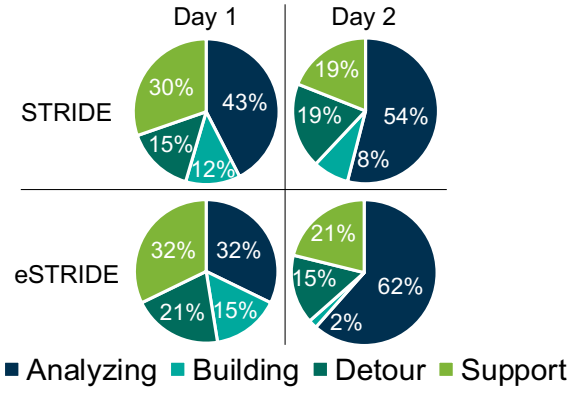
4.3.1. Focus on Activities

The focus of activities was observed by analyzing the coded transcriptions (for ORG A) and the structured notes (for ORG B). Figure 5 shows the prevalence of each of the four activity groups: building the diagram, analyzing the diagram to identify threats, performing support activities (like taking breaks, referring to task description, training material, etc.), and detouring from the task (essentially, loosing focus and wasting time). We remind the reader that we used a subset of all codes from Table 2 in ORG B.

ORG A. During the first day, the STRIDE managed



(a) ORG A



(b) ORG B

Figure 5: RQ3. Distribution of activities in both organizations.

to complete the diagram and started the identification of threats. In general, the team did not focus on one particular activity. Building the diagram covered 28% of the transcription. The diagram analysis covered 27% of the time (mainly discussing the domain and developing attack scenarios). The

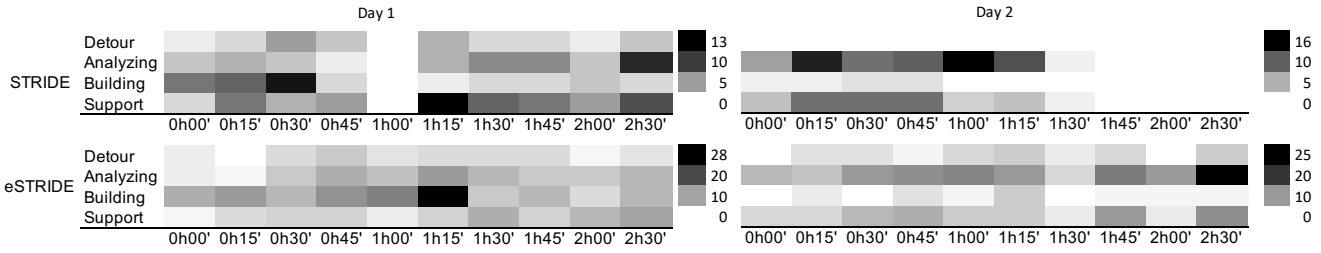


Figure 6: RQ3. Intensity of activity groups over time for the STRIDE (top) and ESTRIDE team (bottom) in ORG A

team detoured often from the prescribed procedure (13%). Finally, the STRIDE team was involved in more support activities during the first day (32%).

The ESTRIDE team invested more on the diagram building during day 1 (45%). They did not identify threats during the first day, however, they performed other analysis activities, namely a thorough asset analysis, for a similar percentage than the other team (25%). Similarly to the other team, the ESTRIDE team detoured often during the first day (11%). In contrast to STRIDE, the support activities amounted to only 19% during the first day.

During the second day, both teams made minor changes to the diagram (5% for STRIDE and 4% for ESTRIDE). The STRIDE team focused on diagram analysis more than the ESTRIDE team did (70% vs 58%), and without detouring from the task. Support activities are comparable.

ORG B. Overall, a similar trend of activity focus can be observed for the teams in the second organization. Namely, the STRIDE team started analyzing the diagram earlier. However, the ESTRIDE team did not spend significantly more time on building and extending the diagram. Both teams focused on diagram analysis during the second day (STRIDE: 54% and ESTRIDE: 62%). In contrast to ORG A, teams in ORG B detoured from the prescribed procedure more often, which is explained by a lesser familiarity of the participants with threat analysis.

Summary. Contrary to our expectations, the ESTRIDE team in ORG B (less security experienced) did not spend more time to create and extend the diagram. This is surprising, considering the additional step of asset analysis, which took more attention of the sibling team in ORG A. Similar to STRIDE, the ESTRIDE teams still analyzed the diagram on the first day, but focused on the analysis on the second day. We did not observe differences in detour and support activities across techniques.

4.3.2. Timeline of Activities in ORG A

Figure 6 depicts the intensity of each activity group over time. We remind the reader that a comparable timeline of activity codes can not be presented for ORG B as the organization did not allow to record the analysis sessions. The intensity (darker color means more intense) is computed by counting the number of code occurrences for each activity group per ten-minute time frame. Note that, the STRIDE transcription is almost half the size of the ESTRIDE transcription (90,612 vs 151,907 characters). This explains the different proportion of code occurrences in the timelines. In what follows, we discuss the similarities and differences in activities during the first day and the second day.

Similarities (Day 1). In the first 15 minutes both teams focused on building the diagram. In particular, both teams focused on abstracting and refining the architecture, discussing the domain, discussing the scope, and drawing on the board. Other support activities in this time window include referring to the case documentation. In the span of the entire session, both teams sometimes detoured from the instructed analysis procedure. The detours during the first day are fairly evenly distributed across teams. Both teams made the assumptions during the first day, and made one last assumption about one hour into the second day.

Differences (Day 1). About an hour into the first day (see 1h15' in left column of Figure 6), both teams focused on support activities (particularly, referring to case documentation). The STRIDE team finished building the diagram after about an hour. They read parts of the case documentation aloud to validate the diagram before they started to analyze it. On the other hand, the ESTRIDE team started extending the diagram with domain assumptions after about an hour. They verified each assumption by reading the case documentation aloud. The ESTRIDE team started looking for threats only on the second day. In contrast to STRIDE, the ESTRIDE team made, overall, less assumptions and documented them early on. The STRIDE team agreed upon some assumptions but did not document them.

Similarities (Day 2). As instructed, both teams performed activities related to diagram analysis which are accompanied by support activities (mainly, documenting threats). Figure 6 (right column) shows a strong focus on analysis activities in three time frames. This can be observed for STRIDE about fifteen minutes and one hour into the second day. ESTRIDE strongly focused on analysis activities at the end of the second day. In all three time frames, the teams managed to thoroughly analyze one threat in a span of five minutes. This entailed (1) developing attack scenario, (2) using an assumption, (3) discussing threat consequence, (4) determining feasibility, and (5) finding a correct threat. We have observed that focusing on the above-mentioned pattern (1-4) is beneficial for correctly discovering threats.

Differences (Day 2). Compared to the first day, both teams detoured less from the instructed analysis procedure. In particular, the STRIDE team did not detour at all. In fact, the STRIDE team finished about one hour earlier. Compared

Table 6
RQ3 (ORG A). The average spatial distance (in number of characters normalized by transcription length) between activity pairs in both teams. The top part contains the pairs that have the most similar distances between the two techniques (small values in the third column). The bottom part contains the pairs with the least similar distances (big values)

Activity pairs	STRIDE	eSTRIDE	Δ dist
Threat reduction & Ref. to assumptions	close	close	0.10
Terminology & Domain discussion	close	close	1.70
High-priority threat found & Attack scenario or vulnerability	close	close	1.84
Asset analysis & Updating diagram	far	close	29.0
Ref. to training material & Unsure	close	far	38.38
Scope discussion & Updating diagram	far	close	38.24

to eSTRIDE, during the second day the STRIDE team focused less on feasibility analysis and on attack scenario development. The STRIDE team often updated their diagram during the second day. Concretely, the team merged data flows and removed one external entity and three data stores. Simplifying the diagram helped the team to finish early.

Summary. During the first day the eSTRIDE team spent more time building the diagram and during the second day, the STRIDE team did not detour from the analysis procedure. We further discuss this in Section 5.

4.3.3. Distance between Activity Pairs in ORG A

We calculated the average distances between all activity pairs for both teams. We then compare the average distance of each pair across the two treatments, by computing the difference. In Table 6, we focus on the activity pairs that have the most similar distances (top) and those with the most different distances (bottom) between the two techniques. For a more complete account, Figure 7 shows this difference for all activity pairs with extreme values. Specifically, we depict the difference if the two codes appeared either close to each other, or far from each other in the transcription of one of the treatments. Note that some activities could only be observed in one of the treatments (e.g., threat prioritization was only performed in the STRIDE team) and therefore do not appear in this Figure.

In addition, we analyzed the distances between single activities in relation to all other activities for both teams. In particular, Figure 8 shows the average distance between finding a high-priority threat and each other activity.

Similarities. Both teams referred to their assumptions during threat reduction to make sure the reductions do not lead to overlooked threats ($\Delta dist = 0.10$). When the teams referred to assumptions, they read the assumption out loud. In addition, both teams engaged in a domain discussion while clarifying the terminology. Finally, both teams found high-priority threats while developing attack scenarios or identifying vulnerabilities.

Figure 8 shows, that the average distance between using assumptions and finding high-priority threats is small in the transcriptions of both teams. The teams used assumptions to justify their reasoning for a threat or vulnerability existence. Therefore, the average distance between referring to assumptions and a finding high-priority threat is small in the eSTRIDE transcriptions.

Differences. In contrast to STRIDE, the eSTRIDE team performed an asset analysis and iteratively updated the diagram with the extra security information ($\Delta dist = 29.0$). In addition, the eSTRIDE team discussed the scope of the analysis while updating the diagram. For instance, they discussed which parts of the system can be left out of the analysis (assumed as trusted). This was not discussed at length in the STRIDE team. During the first day, STRIDE team referred to the training material when unsure.

Compared to STRIDE, the average distance between finding high-priority threats and discussing threat feasibility (and consequence) is smaller in the eSTRIDE transcription (see Figure 8). Further, the eSTRIDE team found the first high-priority threat when analyzing the assets and extending the diagram in the first day. Compared to eSTRIDE, the average distance between finding high-priority threats and referring to training and case documentation is smaller in the STRIDE transcription. In fact, the STRIDE team relied more on the support material, whereas the eSTRIDE team relied more on the domain expert. This may be due to factors of team dynamics, rather then the differences in the techniques. Finally, the STRIDE team made several assumptions during diagram analysis, therefore the average distance between making assumptions are finding high-priority threats is smaller, compared to the eSTRIDE transcription.

Summary. For both teams, assumptions played an important role in finding high-priority threats and in reducing threats. In addition, developing attack scenarios and discussing threat feasibility supported finding high-priority threats (more so in the eSTRIDE team). Certain similarities and differences observed from Figure 7 exist due to the underlying techniques procedures. For instance, attack scenario development precedes documentation in both teams (see small difference in Figure 7). Further, compared to STRIDE, the procedure of eSTRIDE demands extending the diagram with analyzed assets and thus we observe a difference for the activity pair asset analysis and updating diagram across treatments. However, we are aware that differences in activity patterns might also depend on factors related to team dynamics rather then the differences in the techniques.

4.4. RQ4. Security Expertise

As shown in Table 7, we handed out an entry questionnaire to understand the background knowledge and experience in security of the participants. Clearly, we trust the self-assessment of the participants.

From the answers, it is clear that the seniority (Q1) is

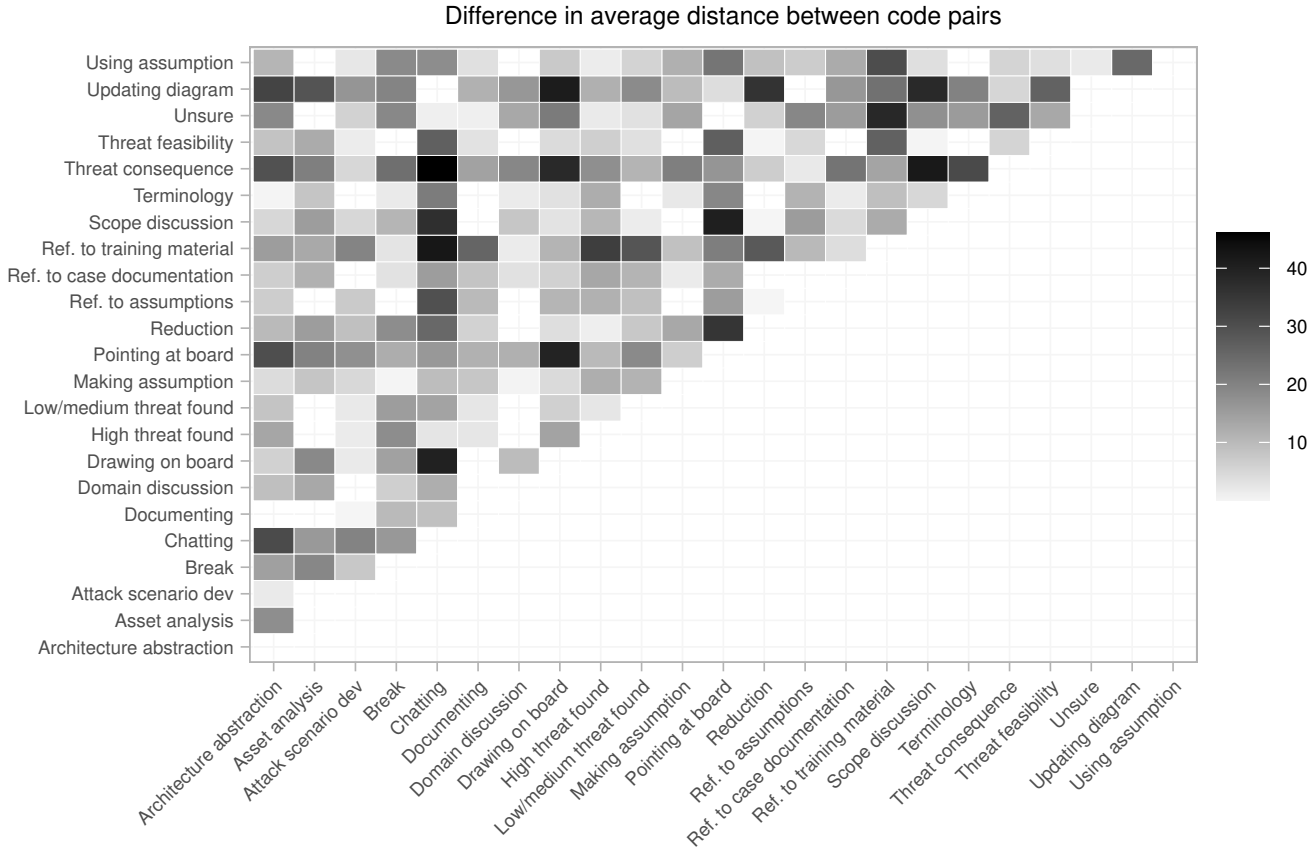


Figure 7: RQ3 (ORG A) The average spatial distance between activity pairs with more extreme values (i.e., activities that appear either close or far in individual treatments). Lighter shade represents pairs that have the most similar distances between the two techniques (small values). Darker shade represents pairs with the least similar distances (big values)

Table 7
Entry questionnaire about seniority and security knowledge

	Frequencies of the answers			
Q1. How many years of working experience do you have?				
ORG A:	1 year (2)	2 - 5 years (2)	5 - 10 years (0)	>10 years (3)
ORG B:	1 year (1)	2 - 5 years (2)	5 - 10 years (2)	>10 years (3)
Q2. How would you rate your familiarity with information security?				
ORG A	No background (0)	Security novice (1)	**Security trained (3)**	**Security expert (3)**
ORG B	No background (1)	**Security novice (7)**	Security trained (0)	Security expert (0)
Q3. How many threat analysis sessions have you been previously part of?				
ORG A	**None (5)**	1 - 5 (1)	5 - 10 (1)	10+ (0)
ORG B	**None (7)**	1 - 5 (1)	5 - 10 (0)	10+ (0)

equally distributed between the participants of the two organizations. Also, the participants have, predominately, no prior practical experience with threat analysis (Q3). Across organizations, however, the participants differed with respect to their knowledge about information security (Q2). In ORG A most participants were previously at least trained in security (3) or were security experts (3). In contrast, the participants employed by ORG B considered themselves security novices (7 out of 8), at best. We have also inquired about their current role in the organization. In ORG A two participants were security consultants, two were hired to conduct threat analysis, and three participants were senior software architects with a security focused role. In ORG B two participants were team leaders and the rest were software developers with experience in a variety of programming languages and platforms.

In summary, the above observations confirm the fact that participants from ORG A have a high security expertise with respect to ORG B. In the following, we summarize the effect of such disparity on the outcomes and execution of the two techniques.

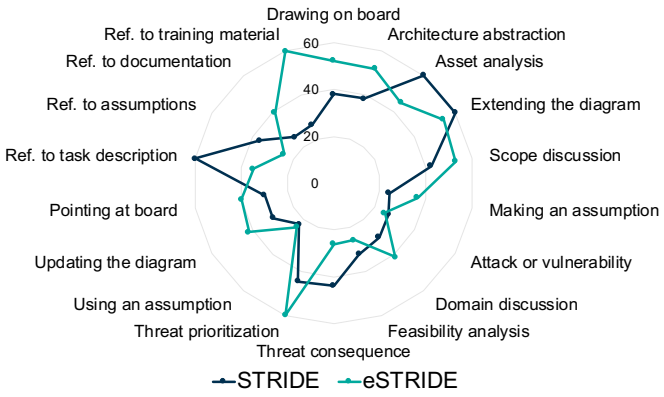


Figure 8: RQ3 (ORG A). The average spatial distance (in number of characters normalized by transcription length) between finding a high-priority threat and other activities for both teams

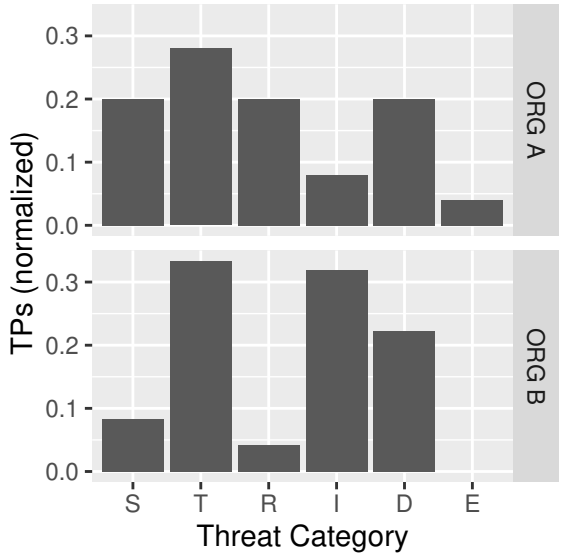


Figure 9: The number of correctly identified threats per STRIDE category normalized by the total number of correct threats found in organization

4.4.1. *Effect on Analysis Outcomes*

Compared to ORG A, both teams in ORG B made mistakes. Namely, the false positive rate ($FP/(TP+FP)$) is 23% and 30% in ORG B, while only one team (STRIDE) made mistakes in ORG A (56%). We remind the reader that most false positives of the STRIDE team in ORG A were assessed as such due to missing information in the documented threat scenarios. The achieved precision (i.e., correctness) of the less experienced teams in ORG B is still high (0.8 and 0.7) compared to the best performance achieved by more experienced analysts in ORG A.

To gather more insight into the effect of security experience on the quality of the threat analysis outcomes, we observed the distribution of the correctly identified threats over the threat categories (i.e., STRIDE) and made a comparison between the two organizations. Figure 9 depicts the number of correctly reported threats per category (irrespective of the technique) normalized by the total number of correct threats in each organization. In ORG B (less security experienced), the teams found more tampering, information disclosure and denial of service threats, compared to spoofing, and repudiation. Elevation of privilege threats were not reported, which is reasonable considering that these are very technical and often used a stepping stone for other threats [29]. Incidentally, similar distributions have been recorded in studies observing STRIDE performance in an academic setting [28, 33].

In comparison, the threats found by ORG A are more evenly distributed across threat categories. Discussing threat feasibility led teams in ORG A to discard many tampering, information disclosure, and denial of service threats within the trust boundaries of the system. Possibly, the less experienced teams are not able to make such judgements. Another possible explanation is that the teams in ORG A modeled smaller diagrams, in particular with respect to the number of modeled processes. Namely, in ORG A 16% of modeled elements were processes, while in ORG B 26% of elements were processes.

Further, the teams in ORG B took about the same amount of time to build and extend the diagrams as teams in ORG A (see Table 4), but spent more time analyzing threats, which was observed as the most time consuming and challenging task of threat analysis [38]. Despite longer sessions, the less experienced teams in ORG B have a much higher productivity (about 6 TP/hour vs about 3). Higher productivity does not, however, imply identification of more high-priority threats. In fact, Table 5 shows that more experienced analysts identify a bigger percentage of high-priority threats, no matter the technique used.

4.4.2. *Effect on Analysis Execution*

Overall, the focus of activities in ORG B is comparable to ORG A (as observed in Figure 5). In both, more support and diagram building activities occurred during the first day, while during the second day the teams focused on diagram analysis. But in ORG B, the difference in focus between the teams is smaller. This is explained by the cross-functional organization of teams in ORG B, which had a positive effect on team dynamics. Further, we have observed that the less experienced teams in ORG B did not discuss feasibility of threats in detail. This observation is in line with the measured high productivity in ORG B, as teams rarely got stuck in reasoning about the probability of threat occurrence and its impact. However, the teams in ORG B detoured more often during both days. In particular, we have made note of teams inquiring the experimenters on-site for support (e.g., 'Is the created diagram sufficient, can we move on?'). Despite such insecurities and shallow feasibility discussions, the teams were able to quickly learn and correctly execute the analysis regardless of the assigned technique.

5. Discussion

In this section we discuss the results and answer the research questions.

5.1. RQ1: Productivity

Beyond counting the number of TPs, we are also interested in getting insight into whether the two techniques provide different results (conversely, overlaps), in terms of security issues identified. Therefore, we have looked into the correct threats reported by both teams and identified those that are similar with respect to (i) diagram location, (ii) threat category, (iii) vulnerability and threat description. In ORG A, six security threats (4 high, 2 low) were correctly discovered by both teams. In this organization, the STRIDE team discovered 6 threats that were not discovered by the ESTRIDE team (2 medium and 4 low-priority). In such cases, the ESTRIDE team either skipped some diagram locations by using reductions (2 low, 1 medium) or agreed that the attack is not feasible (2 low, 1 medium). The ESTRIDE team discovered 5 threats that went unnoticed by the STRIDE team. In contrast, these threats were of high (4) and medium priority (1). In these cases, the STRIDE team could not find any vulnerability or attack. A possible explanation is that the STRIDE team may not have discussed threat feasibility enough to find feasible attack scenarios or that they were simply overlooked.

In ORG B, 14 security threats (4 high, 4 medium, and 6 low) were common for the two teams. The STRIDE team discovered some security threats that were not discussed in the other team, but many were afterwards marked with a medium (18) or low priority (6). Similar to ORG A, the ESTRIDE team discovered several (11) high priority threats, which were not discussed in the other team.

> Concerning, productivity (**RQ1**), we did not observe a difference between the two techniques. Interestingly, however, the two techniques seem to guide the teams to the discovery of different threats. Furthermore, the ESTRIDE teams found more high-priority threats which were overlooked by the STRIDE teams. In contrast, the STRIDE teams discovered more threats of low-priority.

5.2. RQ2: Discovering High-Priority Threats

We looked into how the teams approached the exploration of the diagrams and the potential relation to finding high-priority threats. Despite the dictated exploration strategy by the techniques, the teams were still free to choose concrete elements to follow (e.g., which particular data flow, or which asset flow to consider next). For instance, the STRIDE teams started exploring the diagram starting from one external entity, but then continued differently. STRIDE of ORG A chose to first analyze all the processes, and then proceed to other elements. The STRIDE in ORG B instead explored the diagram following the steps of the scenario (from the documentation) regardless of element types. In both industrial cases, some high-priority threats were located on the trust borders of the system (i.e., external entities, data stores, and processes communicating with the aforementioned). Thus, an *out-side-in* exploration strategy may be useful to find high-priority threats sooner. In addition, an attack on the system boundary element is usually the first step of a chained attack. Hence, accounting for the first steps systematically (at the beginning) may help in discovering chained attack scenarios.

As observed, the ESTRIDE procedure helps in discovering more high-priority threats, but to discover them sooner, the technique must emphasise the importance of analysing the asset sources and sinks first. In addition, an out-side-in procedure may support the discovery of chained attacks.

> Concerning high-priority threats (**RQ2**), we found that the ESTRIDE teams found *twice as many* high-priority threats compared to the STRIDE teams. Further, most high-priority threats that were discovered by the STRIDE teams were also included in the reports of the ESTRIDE teams. In the context of the conducted case studies, the ESTRIDE teams were more complete with respect to finding high-priority threats. Yet, no evidence suggests that ESTRIDE can identify high-priority threats sooner.

5.3. RQ3: Focus and Activity Patterns

The time spent in 'detour activities' is significant (between 10-20% in Figure 5). Regardless of the technique and organization, the teams often discussed the terminology of the threat categories: in particular, the spoofing category in relation to tampering and repudiation. Perhaps, this could be expected for novice analysts, but it happened consistently in all teams. Generally, such detours (or disagreements) were minimized by the process enforcer steering the discussion. In ORG A the STRIDE team often referred to the material to reach consensus, instead. Possibly, this motivated the team to stay closer to the instructed procedure on the second day (with no detours). The way participants handled detours may depend on the team dynamics.

In ORG A, detours often happened when discussing threat feasibility (especially so in the ESTRIDE team). This is confirmed by the small average distance of codes for these activities in the transcription. Therefore, feasibility analysis may have slowed down the overall threat analysis. Discussing threat feasibility often leads to estimating the probability of threat occurrence, which is difficult and can lead to 'analysis paralysis', where too much focus is put on a single threat.

In ORG B, the feasibility of threats was not discussed in great detail. Therefore this pattern (of slowing down the analysis) did not emerge as prevalent. Further, we have observed that detours happened due to discussing the terminology and domain, rather then feasibility analysis.

Regarding the focus on activities (**RQ3**), we found that in one organization (ORG A) the ESTRIDE team spent more time on diagram building. Yet, in the other organization (ORG B) the ESTRIDE team built the diagram faster and still found more high-priority threats. Across organizations, threat feasibility discussion lead to 'analysis paralysis' which slowed down the overall analysis. For what concerns the activity patterns, we found that teams in ORG A were careful when making threat reductions, backing those decisions by referring to assumptions. In addition, assumptions were used by teams to justify the existence of threats (in particular high-priority). Our analysis indicates that some similarities and differences in activity patterns were observed due the underlying procedures of the techniques, although differences might also depend on factors related to team dynamics.

5.4. RQ4. Security Expertise

Our results show that less experienced teams (in ORG B) made more mistakes in their analysis. We looked into the FPs of these teams to better understand their nature.

Most incorrect threats in ORG B were duplicates (STRIDE: 6 out of 12 and ESTRIDE: 12 out of 14). Duplicates are threats with the same diagram location, category and attacker scenario. For instance, two spoofing threats of the same external entity with slightly different threat descriptions. In the following example, the second scenario is a special case (and thus a duplicate) of the first one:

"1: The attacker can send data from anywhere in the world."
"2: The attacker can pretend to be an operator."

Other mistakes included incorrectly reported locations (we have no explanation here) or threat categories (this is certainly due to a lack of familiarity with the definitions of STRIDE). Finally, some threats were incorrect with respect to the domain assumptions. For example, assuming an existing security measure for logging user actions on a data base, and reporting a threat to accountability of a process reading from that data base.

In summary, and in light of the scarcity of security professionals, the trade-off of having more FPs in the analysis results (when using less security experienced analysts) could be acceptable in many organizations, particularly in smaller ones. For instance, a security expert could be hired to *validate* the analysis results at the end, which is a much more light-weight and less costly activity than performing the entire threat analysis. In fact, a recent study [5] reports that agile organizations currently employ similar strategies for conducting threat analysis. In our experience, some mistakes can be also quickly corrected via tool support (e.g., threat locations). However, some mistakes may require more training on security vulnerabilities and classes of attacks (e.g., to avoid infeasible attack scenarios).

Regarding the security expertise (**RQ4**) we found that, in the industrial setting, security expertise has an effect (albeit small in our case studies) on the quality of analysis outcomes and analysis execution. Specifically, we found that the teams with security novices tend to make more mistakes (i.e., the precision is lower), discuss threat feasibility in less detail, but are more productive (i.e., about 6 TP/h in ORG B vs about 3 TP/h in ORG A). Interestingly, the difference in precision between the novice and expert teams is not as significant as the difference in productivity. This may suggest that security expertise may be traded for a faster-paced and less precise threat analysis, however future studies are needed to confirm this claim.

6. Related Work

In this section we position our contributions in the context of related work. First, we discuss related treat analysis methodologies and techniques organized with respect to their risk inclusion. Second, we provide an overview of the related empirical studies.

6.1. Risk-First Threat Analysis

The main characteristic of risk-first threat analysis is that the outcomes of risk analysis (i.e., to some extend quantified risk of compromised assets) are used as input to the threat identification and analysis. However, little existing literature actually leverages such information during threat identification.

CORAS [20] is a model-driven threat analysis methodology. The approach provides systematic guidelines and tools (e.g., asset, threat, risk, and treatment diagrams) to analyze risk during the design phase. After the creation of asset diagrams (step 3), the analysts conduct a high-level risk analysis, where the most important assets (and their threats) are identified. Similar to eSTRIDE, the purpose of this step is to focus the analysis discussion early-on, without going in deeper detail. Next, the threats are identified by means of structured brainstorming. But, the threats are not identified only with respect to the important assets, and afterwards further risk estimation (part of step 6) and risk evaluation (step 7) is required. In comparison, eSTRIDE suggest a detailed account of risks (with respect to security objectives of assets) and existing solutions (treatments) beforehand, and leverages this information to perform reductions (i.e., the pruned table in Figure 2).

Operationally Threat Asset, and Vulnerability Evaluation (OCTAVE) [2, 6, 3] is an asset-centric threat analysis methodology. OCTAVE [2] is organized into three phases. In the first phase, assets and threats are analyzed, current practices and vulnerabilities are scrutinized, and security requirements are derived. In the third phase, threats to the prioritized (in terms of risk) assets are used to prioritize the security strategy. But, the risk analysis is conducted after all the threats have been identified. Interestingly, OCTAVE-S [3] is a light-weight variant targeted to smaller organizations, and has a risk-first flavor. OCTAVE-

S starts with an asset identification and evaluation of security practices. Similarly, SecRAM [21] is a security risk assessment methodology, where primary assets (such as passwords) are first identified and analyzed in terms of impact. With respect to the technology readiness of the analyzed system, SecRAM provides practical guidelines for using the methodology. Best practice catalogues of primary assets, supporting assets, vulnerabilities, threats, and controls may be used to help the analysts. Similar to eSTRIDE, SecRAM and OCTAVE-S introduce risk-first analysis techniques to analyze security threats. In addition, OCTAVE-S suggests to only identify and analyse threats to important assets. In contrast to both, the identified threats are still evaluated for impact and probability (i.e., are prioritized), while eSTRIDE aims to skip this step entirely.

6.2. Risk-Last Threat Analysis

The common characteristic of risk-last approaches is that the outcomes of threat analysis are used as input for risk identification and analysis. In this respect, the works that follow differ from eSTRIDE.

LINDDUN [9] is a privacy threat analysis methodology. LINDDUN is analogous to STRIDE in that it is model-based (using DFDs), and executes a similar procedure (e.g., using a privacy threat-to-element mapping table) to identify privacy threats. In addition, threat tree patterns are provided by the methodology to help threat identification. After all the privacy threats have been analyzed and documented with misuse cases, the methodology suggests to estimate risk levels for each threat (step 8). Notably, the methodology also provides a mapping of privacy objectives to (more than 40) privacy-enhancing techniques (PETs) which could be used for planning mitigations.

Recently, Affia et al. [1] proposed a risk management approach for e-commerce systems. The authors propose to use STRIDE in combination with Information System Security Risk Management (ISSRM) method. Similar to eSTRIDE, the proposed technique starts with asset identification. In what follows, the threats in [1] are identified by performing STRIDE (on the identified assets) and documented in accordance with the ISSRM method. Essentially, the risk values of threats are determined as soon as they are discovered instead of all at once, as in step 4 in Figure 2. Further, instead of using risk information, the authors scope the analysis by eliciting only one threat per each STRIDE category. In contrast to eSTRIDE, there is no notion of systematic threat reduction.

Mollaeefar et al. [24] propose a trade-off analysis technique to solve the problem of analysing risk with multiple stakeholders in the context of privacy concerns. The authors define the problem as a set of (weighted) threats, and a set of security controls associated to said threats. To evaluate the final risk, the technique also takes stakeholders preferences into account. As the threats are input to their risk evaluation, this technique is a risk-last analysis proposal.

PASTA [35] is a methodology targeting business owners for estimating risk by means of attack simulation and threat analysis. The methodology contains seven steps, some of which are similar to the analysis with STRIDE (e.g., creating data flow diagrams, diagram decomposition, determining trust boundaries, and risk and impact analysis - as a final step). In comparison to STRIDE, PASTA suggests a broad list of activities for identifying threats (and vulnerabilities), and modeling attack scenarios.

6.3. Semi-automated Threat Analysis

Several techniques focus the analysis around system assets and include risk as part of their technique, but are semi-automated, thus the list of prioritized threats is not necessarily the main outcome of the analysis. Though the following works are certainly risk-centric, it is hard to determine the exact stage where risk information is used.

Almorsy et al. [4] propose an automated technique using static security metrics (implemented as OCL constraints) to conduct a trade-off analysis with respect to system security. One of the inputs to evaluate these metrics are so-called Security Specification Models, which (among other) contain security countermeasures, objectives and their priorities. But, the proposed technique [4] may also leverage system descriptions models, and abstract source code representations in the analysis. Though all this information may be used in the final trade-off analysis, not all representations are necessary to evaluate the security metrics. For instance, only two (out of seven) metrics include the condition about the priority (in terms of risk) of components (or functions).

Halkidis et al. [12] have developed an approach for a semi-automated risk analysis of threats by analyzing annotated UML diagrams. The authors built a mathematical model of the systems and its defenses, and analyzed it by means of fuzzy fault trees. Similar to eSTRIDE, Halkidis et al. [12] extend the design model with existing security countermeasures (e.g., Secure Pipe). But, the identified vulnerabilities and approximations of risk values are the input for the automated evaluation of risk.

Chen et al. [7] proposed a risk-driven approach for a trade-off analysis of Commercial Off The Shelf (COTS) products. In particular, the authors developed an automated way to extract the vulnerabilities of COTS from a vulnerability database (i.e., CVE), estimate threat risks, and conduct a trade-off analysis by analyzing attack paths.

Finally, in the field of security requirements engineering (SRE) several works [14, 25, 11, 10, 30, 36] are centered around system assets (modeled as goals) and may consider their risks. However, threat analysis is usually performed before the requirements are elicited. For an account of related SRE works we refer the interested reader to [32].

6.4. Empirical Investigations of Threat Analysis

Two recent studies [8, 5] conduct case studies to investigate the challenges of performing a STRIDE analysis. In [5] the authors conduct semi-structured interviews in four agile organizations to investigate the perceived challenges by practitioners conducting the analyses. Interestingly, despite the fact that threat analysis is time-consuming, the practitioners of all four agile organizations see value in performing

threat analysis at regular time intervals. Similar to this work, the case studies involve industrial practitioners and use the coding technique to discover patterns in the collected data. But, the focus of the mentioned works [8, 5] is to record challenges in agile organizations. In contrast, our work is an empirical comparison of two techniques with respect to performance and execution.

Recently, Stevens et al. [31] conducted a case study investigating the efficacy of threat analysis in an enterprise setting. The authors develop qualitative measures to determine the efficacy of the Center of Gravity (CoG) technique. The CoG originated in the 19th century as a military strategy [37] and is by nature a risk-first technique (but has not been extensively used to analyze software security). The authors design a six-step protocol (including surveys and classroom sessions) and involve 25 practitioners in the study. Similarly to this study, they report a very high accuracy of the results handed-in by industrial practitioners. In addition, they provide empirical evidence for a perceived usefulness of threat analysis even after 30 and 120 days, which is very promising. In comparison, our study is novel in that it investigates the timeliness of high-priority threats, and the activity focus of a risk-first and a risk-last technique.

McGraw conducted a study including 95 companies [23]. The study reports on the security practices that are in place in these companies. The BSIMM model does not mention STRIDE per se, rather it highlights the importance of threat analysis. Microsoft has not published evidence of the effectiveness of the STRIDE-per-element technique [29]. Similarly, eSTRIDE (coupled with eDFD) [34] is a recently proposed technique, evaluated solely on the basis of an illustration.

Tuma et al. [33] conducted a controlled experiment comparing the two STRIDE variants, STRIDE-per-element and STRIDE-per-interaction. Similarly to this work, their study quantitatively measures the precision, and productivity of both variants. Their study concludes that there is no statistically significant differences in precision, recall, and productivity of the two STRIDE variants. Yet, the authors speculate that enlarging the analysis scope from one (or two) elements to an end-to-end scenario might have an effect on performance. Their findings are based on quantitative measures, while we adopted a mixed methodology, including a qualitative analysis of recorded sessions.

Scandariato et al. [28] have analyzed STRIDE-per-element and evaluated the productivity, precision, and recall of the technique in an academic setting. The purpose of their descriptive study was to provide an evidence-based evaluation of the effectiveness of STRIDE. Our study, on the other hand, provides a comparative evaluation (by means of a controlled experiment) of STRIDE-per-element and the recently proposed eSTRIDE.

Labunets et al [19, 18] have performed an empirical comparison of two risk-oriented threat analysis techniques by means of a controlled experiment with students. The aim of these studies was to compare the effectiveness and perception of a visual technique with a textual technique. The main findings in [18] show that the visual method is equally effective and leads to equal quality of analysis outcomes compared to the textual method.

Existing literature reports on different measures, such as perception of techniques compared to misuse cases (MUC). The work of Karpati, Sindre, Opdahl, and others provide experimental comparisons of several techniques. Opdahl et al. [26] measure the effectiveness, coverage and the perception of the techniques. Karpati et al. [16] present an experimental evaluation of MUC Map diagrams focusing on identification of not only vulnerabilities but also mitigations. Finally, Karpati et al. [17] have experimentally compared MUCs with mal-activity diagrams in terms of efficiency.

7. Threats to Validity

With respect to the *external threats* to validity, we consider the threat to generalizability of the results. The study was conducted in two different automotive organizations, yet it is not clear to what extent can our findings be carried over to organizations from other domains. In addition, the number of participants was small (15 in total).

With respect to the *internal threats* to validity, we mention the confounding factors that may have influenced the results. The most important confounding factor is team dynamics. The performance of a team might depend on how well the participants work together. It is virtually impossible to control for this factor in an industrial context, as participants are selected based on convenience and availability.

Another potential confounding factor is the different background knowledge across teams. We control for this factor by dedicating a whole workshop (3 hours in ORG A, and 5 hours in ORG B) to training the participants. In addition, we have sent out a short exit survey (with about 10 questions) where we asked the participants whether they felt sufficiently prepared to carry out the task. In both organizations, participants felt like they had a clear understanding of the task, were sufficiently prepared for it, and had a very good understanding of the industrial case under analysis.

We also mention the risk of confirmation bias as some of the researchers are authors of one of the techniques. Our assessments of the reported threats directly impact the measured performance (e.g., the number of TPs effects the precision and recall). We mitigated this threat by discussing our assessments (as a form of quality check) with participants (in ORG B) and reference experts (in ORG A). After the workshops were finished and the reports were assessed (about two weeks), we organized a meeting with the industrial partners to discuss the less clear-cut assessments. The industrial partners were closely familiar with the systems under analysis and could argue for the existence of a threat marked as a false positive (which was never the case) or the misidentified threat priorities. In case of disagreements (2 priorities in ORG B), the assessment was discussed until a consensus was reached.

Finally, we could not replicate the study in ORG B with exactly the same methodology, as the organization did not al-

lowed us to tape-record the sessions. Some measures had to be adapted, which might have led to more imprecise results.

8. Conclusion

This study investigates the benefits and shortcomings of performing a risk-first (eSTRIDE [34]) compared to risk-last (STRIDE [29]) threat analysis in an industrial setting. We conducted two case studies with industrial participants employed by two organizations (based in different countries). In this setting, we gathered empirical evidence about the performance and execution of the two techniques. The contributions of this work are three-fold: (i) a quantitative comparison of performance, (ii) a quantitative and qualitative comparison of execution, and (iii) a comparative discussion of the benefits and shortcomings of the two techniques. This study found no differences in the productivity and timeliness of discovering high-priority security threats. Yet, we showed that the risk-first approach produces twice as many high-priority threats (in both organizations). On the other hand, the risk-last technique found more medium and low-priority threats. Further, we found that security expertise has an effect (albeit small in the context of this study) on the quality of analysis outcomes and analysis execution. To find high-priority threats sooner (in addition to their complete account), the eSTRIDE procedure can be easily tailored to an out-side-in diagram exploration strategy. An interesting future direction could be observing the performance of the two techniques by conducting a longitudinal study to understand whether eSTRIDE's benefits (prioritizing the discovery of high-priority threats) out-weight the limitations (required effort to build eDFDs and sacrificed coverage of low-prioritized threats). In addition, future in-vitro studies with (more participants) could help to generalize the results about the effectiveness of eSTRIDE, especially when compared to other risk-first approaches.

ACKNOWLEDGEMENTS

The authors would first like to express gratitude to the participants of this study who made our observations possible. This research was partially supported by the Swedish VINNOVA FFI project "CyReV: Cyber Resilience for Vehicles - Cybersecurity for Automotive Systems in a Changing Environment".

References

[1] Affia, A.A.O., Matulevičius, R., Nolte, A., 2020. Security risk management in e-commerce systems: A threat-driven approach. Baltic Journal of Modern Computing 8, 213–240.

[2] Alberts, C., Dorofee, A., Stevens, J., Woody, C., 2003. Introduction to the octave approach. Pittsburgh, PA, Carnegie Mellon University .

[3] Alberts, C., Dorofee, A., Stevens, J., Woody, C., 2005. Octave-s implementation guide, version 1.0. Manuel électronique. Pittsburg, PA,: Software Engineering Institute, Carbegie Mellon university .

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

[5] Bernsmed, K., Jaatun, M.G., 2019. Threat modelling and agile software development: Identified practice in four norwegian organisations, in: 2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), IEEE. pp. 1–8.

[6] Caralli, R.A., Stevens, J.F., Young, L.R., Wilson, W.R., 2007. Introducing octave allegro: Improving the information security risk assessment process. Technical Report. Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst.

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

[8] Cruzes, D.S., Jaatun, M.G., Bernsmed, K., Tøndel, I.A., 2018. Challenges and experiences with applying microsoft threat modeling in agile development projects, in: 2018 25th Australasian Software Engineering Conference (ASWEC), IEEE. pp. 111–120.

[9] Deng, M., Wuyts, K., Scandariato, R., Preneel, B., Joosen, W., 2011. A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. Requirements Engineering 16, 3–32.

[10] Elahi, G., Yu, E., 2007. A goal oriented approach for modeling and analyzing security trade-offs. Conceptual Modeling-ER 2007 , 375–390.

[11] Elahi, G., Yu, E., Zannone, N., 2010. A vulnerability-centric requirements engineering framework: analyzing security attacks, countermeasures, and requirements based on vulnerabilities. Requirements engineering 15, 41–62.

[12] Halkidis, S.T., Tsantalis, N., Chatzigeorgiou, A., Stephanides, G., 2008. Architectural risk analysis of software systems based on security patterns. IEEE Transactions on Dependable and Secure Computing 5, 129–142.

[13] Howard, M., Lipner, S., 2006. The security development lifecycle. volume 8. Microsoft Press Redmond.

[14] Jabangwe, R., Nguyen-Duc, A., 2020. Siot framework: Towards an approach for early identification of security requirements for internet-of-things applications. e-Informatica Software Engineering Journal 14, 77–95.

[15] Karahasanovic, A., Kleberger, P., Almgren, M., 2017. Adapting threat modeling methods for the automotive industry, in: Proceedings of the 15th ESCAR Conference, pp. 1–10.

[16] Karpati, P., Opdahl, A.L., Sindre, G., 2011. Experimental comparison of misuse case maps with misuse cases and system architecture diagrams for eliciting security vulnerabilities and mitigations, in: Availability, Reliability and Security (ARES), 2011 Sixth International Conference on, IEEE. pp. 507–514.

[17] Karpati, P., Sindre, G., Matulevicius, R., 2012. Comparing misuse case and mal-activity diagrams for modelling social engineering attacks. International Journal of Secure Software Engineering (IJSSE) 3, 54–73.

[18] Labunets, K., Massacci, F., Paci, F., 2017. On the equivalence between graphical and tabular representations for security risk assessment, in: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer. pp. 191–208.

[19] Labunets, K., Massacci, F., Paci, F., et al., 2013. An experimental comparison of two risk-based security methods, in: Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on, IEEE. pp. 163–172.

[20] Lund, M.S., Solhaug, B., Stølen, K., 2010. Model-driven risk analysis: the CORAS approach. Springer Science & Business Media.

[21] Marotta, A., Carrozza, G., Battaglia, L., Montefusco, P., Manetti, V., 2013. Applying the secram methodology in a cloud-based atm environment, in: 2013 International Conference on Availability, Reliability and Security, IEEE. pp. 807–813.

[22] McGraw, G., 2006. Software security: building security in. volume 1. Addison-Wesley Professional.

[23] McGraw, G., Migues, S., West, J., . Building security in maturity model (BSIMM). https://www.bsimm.com. Accessed: 2017-08-25.

[24] Mollaeefar, M., Siena, A., Ranise, S., 2020. Multi-stakeholder cy-

bersecurity risk assessment for data protection, in: 2020 International Conference on Security and Cryptography (SECRYPT 2020), Springer. pp. 341–348.
[25] Mouratidis, H., Giorgini, P., 2007. Secure tropos: a security-oriented extension of the tropos methodology. International Journal of Software Engineering and Knowledge Engineering 17, 285–309.
[26] Opdahl, A.L., Sindre, G., 2009. Experimental comparison of attack trees and misuse cases for security threat identification. Information and Software Technology 51, 916–932.
[27] Oyetoyan, T.D., Cruzes, D.S., Jaatun, M.G., 2016. An empirical study on the relationship between software security skills, usage and training needs in agile settings, in: 2016 11th International Conference on Availability, Reliability and Security (ARES), IEEE. pp. 548–555.
[28] Scandariato, R., Wuyts, K., Joosen, W., 2015. A descriptive study of microsoft's threat modeling technique. Requirements Engineering 20, 163–180.
[29] Shostack, A., 2014. Threat modeling: Designing for security. John Wiley & Sons.
[30] Sindre, G., Opdahl, A.L., 2005. Eliciting security requirements with misuse cases. Requirements engineering 10, 34–44.
[31] Stevens, R., Votipka, D., Redmiles, E., 2018. The battle for new york: A case study of applied digital threat modeling at the enterprise level, in: SEC'18: Proceedings of the 27th USENIX Conference on Security Symposium, USENIX Association. pp. 621–637.
[32] Tuma, K., Calikli, G., Scandariato, R., 2018. Threat analysis of software systems: A systematic literature review. Journal of Systems and Software 144, 275–294.
[33] Tuma, K., Scandariato, R., 2018. Two architectural threat analysis techniques compared, in: European Conference on Software Architecture, Springer. pp. 347–363.
[34] Tuma, K., Scandariato, R., Widman, M., Sandberg, C., 2017. Towards security threats that matter, in: Computer Security. Springer, pp. 47–62.
[35] UcedaVelez, T., Morana, M.M., 2015. Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis. John Wiley & Sons.
[36] Van Lamsweerde, A., 2004. Elaborating security requirements by construction of intentional anti-models, in: Proceedings of the 26th International Conference on Software Engineering, IEEE Computer Society. pp. 148–157.
[37] Von Clausewitz, C., 1873. On war. volume 1. London, N. Trübner & Company.
[38] Yskout, K., Heyman, T., Van Landuyt, D., Sion, L., Wuyts, K., Joosen, W., 2020. Threat modeling: from infancy to maturity, in: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results, pp. 9–12.