CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Towards Automating a Risk-First Threat Analysis Technique

Bachelor of Science Thesis in Software Engineering and Management

KARANVEER SINGH
MARGIT SAAL
ANDRIUS SAKALAS

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Supervisor: KATJA TUMA
Examiner: Richard Berntsson Svensson

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Towards Automating a Risk-First Threat Analysis Technique

Margit Saal
Dept. of Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden
gussaalma@student.gu.se

Andrius Sakalas
Dept. of Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden
gussakan@student.gu.se

Karanveer Singh
Dept. of Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden
guskaransi@student.gu.se

***Abstract*—During the past decade, secure software design techniques have found their way into the software development lifecycle. In this context, threat modeling (or analysis) methodologies are used to systematically identify threats in the design phase of software development. However, threat modeling is often performed manually, which is time-consuming and error-prone. An existing methodology called eSTRIDE tries to solve the problem of high manual effort by introducing security related enrichment's to the software architecture models and by introducing reductions during the analysis. But the lack of tool support may counteract the advantages of using the methodology. Therefore, the aim of this work is to find out how to support semi-automation of eSTRIDE. We have produced a prototype tool using the design science research methodology, which allows the user to create or modify an extended Data Flow Diagram of their system and perform eSTRIDE. A workshop with ten participants was used to evaluate the tool. We studied the average precision, recall and productivity of the analysis results. Finally, we found the perceived usability of the tool, which was mostly positive.**

I. Introduction

Security and privacy issues are becoming a major concern in organizations developing software products Meland and Jensen [1] Williams et al. [2]. Secure software design is needed to reduce the risk of security breaches in organizations. Security could be achieved using threat modeling techniques to find out what might go wrong with a software project - the potential threats are then identified and prioritized Shostack [3] Cruzes et al. [4]. The goal for the built software is to be available as well as maintain necessary confidentiality and integrity even if attacked Cruzes et al. [4]. Using threat modeling in software projects, security analysts can find threats in the early stages and then plan the project in a way where the threats can be mitigated or tracked Sion et al. [5].

There is existing literature on threat modeling methodologies and even supporting semi-automation of the analysis procedure Shostack [3] Sion et al. [5] Lund et al. [6] Tuma et al. [7]. Many threat modeling methods use Data Flow Diagrams (DFDs) as a base, which are architectural views that show software components and data exchanges between these components Shostack [3] Sion et al. [5]. Most model-based threat modeling techniques explore the diagram and find possible threats in different locations. Many threat analysis techniques use regular DFDs, which do not include already known constraints or security decisions. Not taking this security relevant information into account could lead to wasted effort as all threats or threat categories are elicited and analysed, but later a large amount of them are discarded due to their low priority Sion et al. [5].

STRIDE is a model-based methodology that helps to systematically discover security threats in a system design. It stands for the following threat categories: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege, which are mapped to the elements of the aforementioned DFD base model during threat elicitation Shostack [3]. Analyzing only a handful of software components with STRIDE can reveal many security threats Scandariato et al. [8]. This issue is known as the *threat explosion* problem, which occurs during threat category mapping, resulting in processes potentially being exposed to threats from all the categories, and in large systems the number of these processes can be very high. Therefore, performing systematic threat analysis on large software architecture models requires high manual effort for organizations where security experts can be limited. These issues can also lead to overlooked threats.

Herein we will use the extended STRIDE (eSTRIDE) methodology. eSTRIDE uses extended Data Flow Diagrams (eDFDs) Tuma et al. [9], which include extensions to the regular DFD that specify assets, their traces, security objectives, value of security objectives, domain assumptions and communication channels. The ambition of this risk-first approach is to lower the manual effort required to perform a fully-fledged STRIDE threat analysis Tuma and Scandariato [10] Scandariato et al. [11]. It is referred to as risk-first because the assets are analyzed and potential risks are identified before the security threats are identified. The methodology creates an abstraction before threat analysis by bundling data flows and processes which help resist *threat explosion* Tuma et al. [9]. The aforementioned extensions in the model are later-on used during the analysis to make reductions. The aim of eSTRIDE is to find the most important security threats faster. Many companies overlook the usage of threat analysis in their projects as doing it manually can be very resource-intensive. If the process was semi-automated, a lot of time and money could be saved. The lack of a tool counteracts the advantages

presented by the methodology. Therefore, there is a need to find out how to support the semi-automation of eSTRIDE.

To this aim, this study contributes with a prototype tool implemented by following the design science methodology. This research may benefit people working with threat analysis of software projects in the design phase. The research could also be beneficial to the researchers in the field as we will be looking into how the process of automating threat analysis can be supported. The developed artifact was evaluated with workshops and interviews. The result of this research can also lead to developing even better automation of the methodology in the future.

The rest of this thesis is structured as follows. In Section II we describe the research questions, and in Section III we present the background and position our work in the context of existing literature. Section IV describes the research methodology used in this thesis and the threats to validity. Section V gives an overview of the developed artifact, evaluation results and discussion. Section VI presents the conclusions and possible future work.

II. Research Questions

This section presents the research questions this study aims to answer. The study will firstly focus on the concepts of modeling and representing eDFDs and eSTRIDE reductions. Thereafter, we will find requirements that emerge after evaluating the artifact. In the end, we want to know up to what extent the artifact automates the methodology specifically with a focus on the precision, productivity and recall of an analysis performed with the tool as well as the perceived usability of the tool.

First, we will need to find out requirements for semi-automating eSTRIDE as there is no semi-automation tool for it yet. Hence, we constructed the first research question.

- **RQ1:** What is required to effectively support the automation of eSTRIDE?

The first sub-question is constructed to find the requirements for modeling and graphically representing eDFDs.

- **RQ1.1:** What are the requirements for modeling and graphically representing eDFDs?

The eSTRIDE reductions have been initially proposed as a set of guidelines written in natural language and have not been implemented yet. The second sub-question is constructed to find requirements to semi-automate the eSTRIDE reductions.

- **RQ1.2:** What are the requirements for supporting eSTRIDE reductions?

The second research question aims to find emergent requirements from the first iteration of the design science methodology.

- **RQ2:** What are the emergent requirements that are required for automating eSTRIDE?

The third research question is constructed to evaluate the precision, productivity, recall, and perceived usability of the prototype tool.

- **RQ3:** To what extent is the developed prototype tool automating eSTRIDE?

The first sub-question serves the purpose to quantitatively estimate the performed eSTRIDE analysis.

- **RQ3.1:** What is the precision, productivity, and recall of an analysis performed with the eSTRIDE prototype tool?

The second sub-question serves the purpose to use the perception of participants to qualitatively estimate the usability of the tool.

- **RQ3.2:** What is the perceived usability of the tool?

III. Background and related work

In this section we cover the background and related work for this research.

A. Background

1) STRIDE: is a systematic threat analysis methodology. It consists of steps such as: defining users and realistic usage scenarios, gathering assumptions, constructing a DFD of the system Shostack [3]. Afterwards, the analyst maps the diagram elements to the following STRIDE categories:

Spoofing (S) refers to a person or program that successfully pretends to be another legitimate user or program. Tampering (T) refers to someone modifying application resources, such as memory data, that they are not supposed to. Repudiation (R) refers to someone able to deny doing an action within the system. Information disclosure (I) refers to a threat agent obtaining private information they are not supposed to access. Denial of service (D) refers to attacks on the system that make a system resource unavailable to its intended users. Elevation of privilege (E) refers to someone obtaining access to resources that they should not be able to access, as the resources are normally protected.

Afterwards, threats are refined and documented.

2) DFD: is a diagram that represents how the data enters, leaves and traverses a system. DFD elements comprise of processes, where data is processed; data flows, where information travels from element to element; data stores, where data is stored; and external entities, which are an external source or target for data. DFDs are commonly used in threat modeling.

3) eSTRIDE: is a model-based technique to systematically discover the security threats in a system design Tuma et al. [9]. The ”e” stands for extended as it expands on the STRIDE methodology. Furthermore, the methodology uses the aforementioned eDFDs, which include the necessary extensions for the methodology. One way eSTRIDE fights threat explosion is with abstractions, such as process folding and data flow bundling, that are used to reduce the complexity of the diagram.

Process folding is when processes are merged into one element. This can be done if the data flows transporting assets between the processes are mounted on the same channel. If the priorities are low or medium, at least one domain assumption has to be done to mitigate one of the objectives of an asset. However, if any of the assets travelling have high priority,

there must be mitigation domain assumptions in place for all the objectives.

Data flow bundling is when two or more data flows are bundled together into one element. This can be done if the flows are travelling on the same channel and from the same source to the same target. However, in case the flows go through a critical area (high priority), one needs to check for end-to-end flows of the assets and see if the unaligned parts of their routes are not critical. The precise guidelines for process folding and data flow bundling are described in Tuma et al. [9].

The aforementioned abstractions are done before the threats are identified. The analyst will consider each end-to-end scenario with assets that have high-priority objectives and map the elements that handle such assets to STRIDE categories. Since the analyst will know the priorities and objectives of each asset, they can focus only on the most important threat categories. Domain assumptions can also help the analyst to determine whether a threat category is applicable in a specific location. As a result, helping to limit irrelevant or low-priority threats.

4) eDFD: is based on the DFD, but includes extensions such as assets, which are valuable data Tuma et al. [9]. The asset's source and target will be marked in the diagram and its path will be considered as an end-to-end flow. Furthermore, each asset will have at least one security objective and priority. An objective could be either Integrity, Confidentiality or Availability. The priorities are either High, Medium or Low, and show the severity of impact if that objective gets compromised. The diagram also holds extensions such as communication channels, which show the network each asset moves on. Finally, a domain expert will also be able to extend the diagram with domain assumptions, which are assumptions that some component or objective is already secure, because security mechanisms are already in place.

B. Related Work

In this section we address the related work with respect to the area of threat modeling and automation of threat modeling.

1) Threat modeling: Threat modeling is an essential part of many companies' development process e.g. Microsoft Sion et al. [5]. It is essential to find possible threats in the early stages of a project and then plan the development in a way where they can mitigate or track these threats Shostack [3]. Making design decisions according to the found threats can reduce the risk of having to change plans and features on the way, which usually can be very costly and time-consuming.

There are many threat modeling methodologies available. Most of them use some sort of an architectural diagram of the project as a base, which means they are model-based. There are methodologies that use DFDs, such as STRIDE and LINDDUN which are software-centric approaches Shostack [3] Sion et al. [5]. These methods are systematic, as they go through the diagram and find potential threats in the system in each location, but they can also be very repetitive and time-consuming, as all threats are found and not just the most relevant ones (or higher-priority ones). Low-priority threats can be ignored later, but spending resources to find all these threats can be inefficient. To counter this issue, some researchers have included security solution elements to their DFDs, which then are considered during elicitation Sion et al. [5].

LINDDUN stands for different threat categories like Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of information, Unawareness and Non-compliance. It is a model-based threat modelling technique that has a primary focus on privacy. The methodology provides a thorough list of privacy threats that can be used as an insight during analysis. Moreover, the technique analyses a DFD and tries to map potential subjects to distinct privacy threats Wuyts et al. [12]. Process for Attack Simulation and Threat Analysis (PASTA) is an asset-centric approach that focuses on developing measures that counter an attack related to the value of the asset. PASTA also uses DFDs, use cases, building attack trees etc., to provide a systematic analysis of the intruders profile, for example, the most likely attack methods and/or desired assets by the intruder UcedaVelez and Morana [13]. The CORAS methodology is also asset-centric and includes a language, a tool and uses Unified Modeling Language (UML) based CORAS diagrams as a starting foundation Lund et al. [6].

Attack graphs are another threat modeling technique, which can be made with the knowledge of vulnerabilities on local hosts and how the different hosts are connectedSheyner et al. [14]. The graphs provide analysts with series of potential exploits.

2) Empirical studies with threat modeling: There have been studies on the STRIDE methodology and its productivity, correctness (precision) and completeness (recall). For instance, one empirical study shows that performing STRIDE was not perceived as difficult, but rather time-consuming Scandariato et al. [8]. They also mention that due to overconfidence, the possibility of threats going undetected is high. Another study was conducted, where students had to identify threats using either STRIDE per element or STRIDE per interactionTuma and Scandariato [10]. The average time spent in identifying threats was 3.5 hours for per element analysis and 3.95 hours per interaction, which shows it can take a lot of time to complete.

3) Automating threat modeling: Lack of implementation tools can weaken any threat modeling methodology Mauw and Oostdijk [15]. This means that automation tools for different threat modeling techniques are important. As threat modeling can be very time-consuming and also requires security specialists, semi-automation tools have been made to try to reduce the resources needed. ThreatModeler is a defense-oriented tool that uses attack libraries for threat analysis Shostack [3]. The tool produces attack trees with the component, requirements that can be violated, threats and attacks while our approach produces a list of threat categories applicable for each component.

The Microsoft Threat Modeling tool is known to be easy to use and it allows the user to draw their diagrams Shostack [3].

Similarly to our approach, the tool then analyzes the model and elicits possible threat categories. However, the Microsoft Threat Modeling tool is based on the STRIDE methodology.

Another tool is Security and Privacy Architecture through Risk-driven Threat Assessment (SPARTA), which extends DFDs with already applied security and privacy countermeasures and takes it into account during the threat elicitation so only the most important threats are prioritized Sion et al. [5] Sion et al. [16]. Similarly to us, SPARTA is also implemented as an Eclipse plug-in and uses a similar meta-model for representing the architectural model. However, SPARTA generates a list of threat categories based on STRIDE and then suggests threat prioritization based on quantifying the risk. On the other hand, our tool relies on the security experts opinion about the asset importance and elicits the threat categories according to eSTRIDE.

The aforementioned CORAS methodology also includes a semi-automation tool Lund et al. [6]. However, as mentioned before, their methodology uses the CORAS specific diagrams and language in contrast to our approach.

More interesting work on automating threat analysis has been recently done, where the authors also introduce additional semantics and guidelines for building threat models Berger et al. [17]. However, they do not handle threat explosion and do not analyze end-to-end flows of assets. No existing work offers support for the semi-automation of the eSTRIDE methodology.

IV. Research Methodology

In this section we describe the research methodology used in this work and the threats to validity of this study.

The selected research methodology is Design Science and the strength of design science research is that it focuses on knowledge-intensive design and helps solve real problems by developing innovative artifacts. In our research, we have created a prototype tool as the artifact to semi-automate threat analysis Hevner et al. [18]. We decided to follow the Design Science Research Process (DSRP) methodology as it helps to ensure focus on the artifact while prioritizing its relevance to the industry Peffers et al. [19]. In what follows we describe the steps of DSRP and briefly show the main outcomes of applying each step in our research.

A. DSRP Steps

1) Problem identification and motivation: Problem identification and motivation focuses on the research problem. There is an architectural threat analysis method called eSTRIDE, which only shows the most relevant and important threats for the current project, but there is no semi-automation tool that implements the methodology. A solution that semi-automates the process of architectural threat analysis in the early stages of a project can ensure that possible threats can be mitigated, which helps saving resources and increases the chance of the project being successful. Whereas, many of the threat methodologies used in existing tools present repetitive and less relevant threats, which then need to be manually analyzed and omitted.

2) Objectives of a solution: Objectives of a solution identifies why the solution is needed and aims to provide a solution for the problem identified in Step 1. The main objective of our solution is to create a tool that implements eSTRIDE threat analysis methodology and effectively supports the semi-automation of this methodology. Our aim is to support eSTRIDE reductions by providing abstractions which can be achieved by *bundling data flows* and *process folding* before performing threat analysis. This helps us to counter the 'threat explosion' problem i.e, when a large number of threats (often irrelevant) are found when STRIDE is performed on a large DFD.

3) Design and development: Design and development focuses on designing and implementing our solution of the research problem. The design of the artifact was iteratively developed and discussed between the team members. We approached this phase strategically. First, we wanted to find out the requirements for modelling and representing eDFDs. This was achieved by studying related work. Second, we wanted to support eSTRIDE methodology including abstractions. We have developed a prototype tool as a design science artifact. The tool can help perform threat analysis on the modelled eDFD and list the categories of potential threats to the user.

4) Demonstration: Demonstration focuses on presenting the developed artifact. We demonstrated our artifact by conducting two workshops in an academic environment, where the tool was used to solve an existing example problem. First, we gathered demographic data of the participants with an entry survey. Second, we had a training session with the participants, where we introduced threat analysis and the participants had a hands-on session on how to perform manual eSTRIDE. Thereafter, we explained how to use the tool and provided documentation related to the task they had to perform with it. The participants were provided with a DFD and had to extend it using our tool. Then the tool gave the users abstraction recommendations according to the eSTRIDE methodology. In the end, relevant threat categories were printed to the user according to their model. The workshops had in total 10 participants, primarily undergraduate students in the software engineering field.

5) Evaluation: Evaluation focuses on evaluating the artifact and how it relates to the research problem. To evaluate the artifact tool, we conducted 2 workshops to gather quantitative and qualitative data. We collected the data from the resulting threats found by the participants using our tool, conducting entry survey and exit interviews related to the perceived usability of the tool. We conducted the workshops on two separate days.

6) Communication: Communication focuses on the studied problem along with the artifact developed which is presented to relevant audience. We will present our findings in the thesis for other researchers to base or extend their work.

B. Data Collection

We conducted a workshop where we introduced the participants to threat analysis. We reached out to two companies

asking for their security experts to participate in our workshop in order to validate our prototype tool. We did not get any response from those companies. Instead, due to time constraints we used convenience sampling where we contacted our colleagues related to the Software Engineering field. We reached out to them by sending emails with information about the workshop. Everyone who responded was invited to participate in the workshop. The time taken to complete the workshop was 3 hours where we gave the participants breaks throughout the course of the workshop. Figure 1 shows the structure of the workshop for collecting data -

a) Entry Questionnaire: (5 minutes) - We conducted a survey [1] to collect information about the background of our participants and if they had any knowledge related to threat analysis. The survey consisted of both open and close ended questions. The theoretical population were people who are working or studying in the field of Software Engineering.

b) Introduction: (30 minutes) - We conducted a training session. The training material can be found here [2]. It consisted of three parts: (1) Training on security principles and threat analysis (including theory behind eSTRIDE methodology), (2) Hands-on exercise with a pen-and-paper execution of eSTRIDE and (3) Introduction to tool and solving a problem with it.

c) Manual Threat Analysis: (60 minutes) - We conducted a hands-on exercise with the participants where we performed eSTRIDE on a simplified in-vehicle architecture, previously analyzed using eSTRIDE in Tuma et al. [9]. Modern vehicle systems are highly complex and comprise of hundreds of various components called Electronic Control Units (ECUs) which are responsible for specific features of the vehicle. If any one of those ECUs are compromised the driver may potentially lose the functionality provided by that ECU and this might lead to a dangerous situations. To mitigate these potential security breaches the architecture of the vehicle system needs to be modelled and thereafter analyzed to discover the potential threats. The participants were provided with (1) Domain Description including scenarios, (2) Architecture of the system and (3) DFD of the architecture. They were asked to perform the eSTRIDE methodology to elicit possible threat categories for the system and were later compared to the ground truth. They completed this tutorial in a group. The reasoning behind providing them with a problem was to get them familiar with the threat modelling technique and have a valid comparison point for the tool and the time taken to perform it.

d) Introduction to Artifact Tool: (10 minutes) - The participants were provided with a hands on tutorial explaining different elements and functionality of the tool.

e) Semi-Automated Threat Analysis: (45-60 minutes) - The problem is related to a Home Monitoring System (HomeSys) which is a system for remote monitoring of residential homes. The purpose of the system is to provide necessary features for clients to automatically receive notifications and manage different events in their homes.

HomeSys consists of documentation related to its domain, requirement specification and architectural description which comprises of 30 pages. Therefore, it is large enough to reason about security threats. Also, the HomeSys example was used in a previous study to measure performance of STRIDE techniques Tuma and Scandariato [10]. Therefore, we consider it as a valid case for our study.

The HomeSys can have multiple sensors and triggers installed across the home. They collect sensitive data from smoke detectors, temperature sensors, cameras, etc. If the system is compromised the alarms may not be triggered, there could be an unauthorized access to camera feed, etc. With the growing number of sensors and triggers there is a need for a secure design of the system. Threat analysis can be performed on the designed model, to find and mitigate the potential threats.

We provided the participants with system documentation which includes (1) The domain description of the system with scenarios, (2) The requirement specification of the system, (3) Architectural description documented in unified modelling language (UML) and (4) DFD of the architecture. The DFD of HomeSys was larger compared to the simplified in-vehicle architecture. The complete description of the system is available in goo [20]. They were asked to use our artifact tool to perform threat analysis using eSTRIDE. Their results were later on compared to the ground truth. The whole task was to be done individually.

f) Exit Interviews: (15 minutes) - Interviews were conducted during which the interviewee was presented with open-ended questions regarding the usability of the tool. Additionally, we asked questions related to the perception of the interviewee about their correct, incorrect and overlooked threats compared to our ground truth.

C. Data Analysis

We have carried out an eSTRIDE analysis of the systems used in the study. Therefore, we have a 'reference solution' which is based on existing analysis of the HomeSys Tuma and Scandariato [10]. From hereon, we state the 'reference solution' as the ground truth which contains all the possible threat categories that an optimal implementation of the methodology would produce.

An overview of the terminology used in the thesis is presented in the Table I. Every threat category in the report has been noted as either correct (true positive), incorrect (false positive) or overlooked threat (false negative).

We have used both quantitative and qualitative content analysis techniques Linker et al. [21]. Qualitative content analysis was used to analyze responses from the interviewees and quantitative data analysis was used to analyze the outputs the participants produced by performing the semi-automatic eSTRIDE. We have analyzed the data of our participants to see how many true positives, false positives and false negatives were found. The threat categories were compared to the ground

[1] https://forms.gle/rGPmR2LoNJty8LG3A

[2] https://docs.google.com/presentation/d/1IJ_USJ57fvo00DsM8s_v5XwxVAYBaumrReB5iQixS5A/edit?usp=sharing

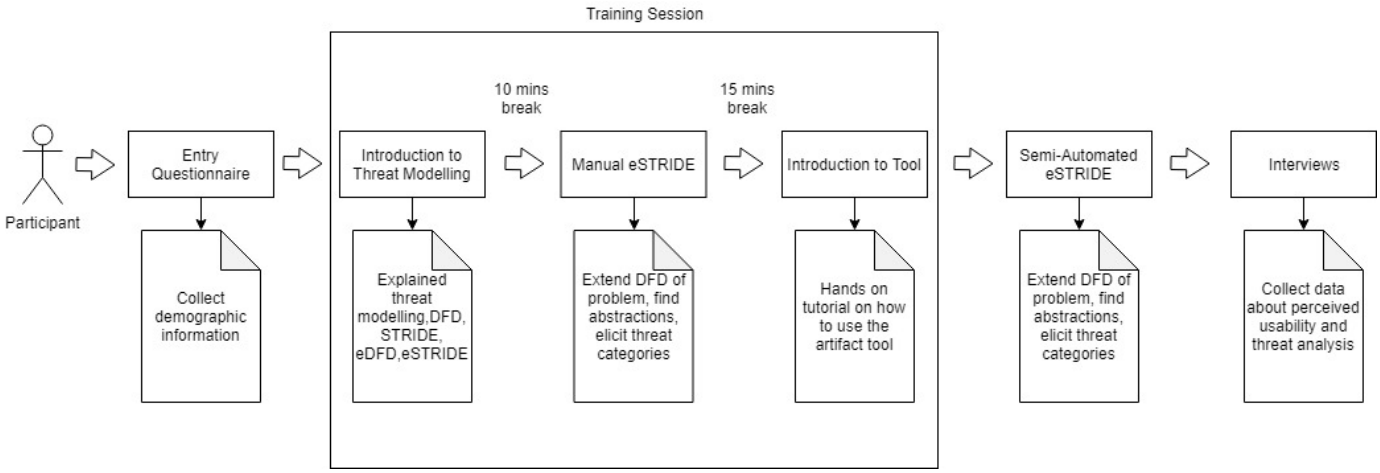

Fig. 1. Workshop Structure

TABLE I
TERMINOLOGY

Term	Meaning
True Positives (TP)	Correct threat category
False Negatives (FN)	Overlooked threat category
False Positives (FP)	Incorrect threat category
Precision (Prec)	TP/(TP+FP)
Recall (Rec)	TP/(TP+FN)
Productivity (Prod)	TP/Time

truth which was created by us and revised by our supervisor. We consider a threat category as a true positive when it has the same name and location compared to the ground truth. We consider threat categories as false positives when they have a different name in a specific location compared to the ground truth. False negatives are threat categories present in a location in the ground truth, but absent in the participants' results. We have also calculated the average precision, recall and productivity, where precision is the percentage of the produced threat categories that are correct, recall is the percentage of the existing threat categories that are discovered and productivity is the threat categories produced per minute.

We have analyzed the responses of our participants by finding themes and patterns in their answers to identify the main topics shared between the interviewees Wildemuth [22]. We familiarized ourselves with the data by going over it multiple times. We started to detect similarities between responses and created different themes while analyzing the data. Thereafter, we created annotations on the data to group them to those themes.

The interview questions were also aimed to answer the research questions. All interview questions can be found in the appendix. Interview questions 2, 3, 4 and 5 on regard the usability of the tool and are related to the RQ3, RQ3.1 and RQ3.2. We also asked our participants to compare their results to the ground truth to investigate their perceived precision, productivity and recall. The perception of precision, recall and productivity questions 1 and 2 are also related to RQ3.2.

D. Threats to Validity

The four perspectives of validity and threats are considered as presented in Wohlin et al. [23].

1) Internal Validity: This validity threat is connected to different factors what might affect the results of the research. As the knowledge base can be insufficient, it means that the process of the research could include a lot of trial and error Hevner et al. [18]. One threat could be that we might have made mistakes during the creation of the ground truth or while assessing their reported results. As the ground truth was created by us and our supervisor there could be human error. This could have been mitigated if security experts would have validated the ground truth, but due to time constraints we could not find available security experts. Nevertheless, this threat is minimized as our ground truth is based on the existing analysis of HomeSys.

A threat could be the maturation of the participants as they could have become tired or bored because of the size of the task. We provided them breaks to lessen the chances of them becoming tired.

The participants also had a similar background as we found out with the entry survey. This lessens the chance of having their individual differences affect the results. To avoid bias in the data analysis, the qualitative data was analyzed independently by all team-members. Then, the results of each independent analysis were compared and discussed between team members.

2) Conclusion validity: This validity threat is concerned with incorrect conclusions made by the researchers due to various sources. The time spent on the task was reported by the participants themselves, which could be a limitation. We did stress the importance of accuracy of the time reporting. During the workshops we were there to try to reduce the possibility of subject influence. Our results might be affected by a small sample size (10 participants) and biased by the use

of convenience sampling. Also, participants may have attended our workshop just because they had an interest for our research topic. In this case, we might have not collected data from the people who represent the whole population, for example, people with no interest in the research topic. Due to this reason, the results might have been biased. With qualitative data from interviews, the sample does not need to be very large. Our sample is also quite homogeneous so there cannot be big variation because of individual differences.

We went over the interview questions and workshop contents with our supervisor and tried to perfect them and also had a dry-run of the workshop with two participants, where we got feedback for everything including the interview questions. Therefore we think that the chance of subjects misunderstanding the questions or exercises is small, but it cannot be completely excluded.

3) Construct validity: This validity threat should show if the observations of the researchers actually correspond to the research questions and the idea of the research. We might have written biased interview questions. To mitigate this limitation, we went over the questions with our supervisor to get feedback and as mentioned before, we also had two people do a dry-run of the workshop and give feedback on the interview questions. According to their feedback, we implemented changes to reduce potential misunderstanding.

Another threat could be that we collected the requirements (RQ1.1 and RQ1.2) based on reading related work and looking into existing tools rather than going to companies who actually collect requirements.

We explained the topic in the workshop with a crash course, the participants also did a manual tutorial and we also made sure to specify to the participants that anonymity will be provided and that we expect complete honesty in the interviews. This was done to reduce the threat of hypothesis guessing, which means that the participants could try to guess what the researchers want. We believe that as the topic is not very personal, the participants understood our need for their honest feedback and felt comfortable giving it.

4) External validity: This validity threat is concerned with the ability to generalize the findings outside the scope of the study. Since the workshop included using the design artifact on a single project, we cannot be sure that we can generalize for different projects or environments Hevner et al. [18]. Since our participants were mainly undergraduate students instead of professionals, it might also affect the generalization of the results. We had no people from the industry, because it was hard to find people who were able to dedicate their time to participate in the study. Nonetheless, as the software engineering field is progressing, students are more and more integrated in the field during studies. This reduces the gap between the knowledge from people who work in the industry and students. Also, half of our participants were working as software developers. We conducted workshop to provide appropriate amount of knowledge to the students, so that they have the competence to perform the analysis. According to some sources, participants in the research do not only have to be people from the industry, as students can be almost on the same level as the people who work in the industry Svahnberg et al. [24]. Since our sample size was small, we feel like the study should be repeated with a larger sample size and more random sampling to be able to generalize outside the scope of the study.

V. Results and Discussion

In this section we give an overview of the developed artifact, describe the results from this study and relate them to existing work.

A. Developed Artifact

We have developed a tool as a plugin for the Eclipse environment that lets the user create eDFD models and detect threat categories based on the eSTRIDE methodology. The tool uses a meta-model that contains information about all the elements of the eDFD and the relationship between them as a base structure for the eDFD diagram. When the user launches the Eclipse Runtime, they can use a graphical model editor based on Sirius framework to model an eDFD. The user can select the needed eDFD component from the palette and change the elements properties. The tool also allows the user to simultaneously modify a textual representation of the eDFD based on the xText framework. Changes made on the graphical representation will be updated in the textual representation and vice-versa. Once the user has created the eDFD of their system, they can use our tool to perform threat analysis. Our algorithms will analyze the model and provide the user with suggestions for abstractions such as bundling data flows and folding processes. The user can then update the model based on the suggestion list. The suggestions show which elements can be merged together to reduce the complexity of the diagram, which counters threat explosion. After bundling the suggested data flows and folding the processes, the user can execute the eSTRIDE algorithm, which will analyze all the elements in the eDFD model and elicit possible (relevant) threat categories in a table format. The developed artifact tool can be found here [3] and screenshots of the developed artifact can be found in the appendix.

B. Participants

We collected information on the background of the participants with a short entry questionnaire. The survey consisted of questions about the participant's education, work experience and perceived familiarity of related topics to the workshop. From the survey we found that most of our participants were of a similar background. Most of the participants were students, half of them working as software developers. Most had 1-3 years of experience both in software architecture and software development. Six participants had less than a year of experience in threat analysis and four had none. None of the participants had ever used a threat analysis methodology nor any threat analysis tools. Most participants had at least some knowledge on how to make a DFD, which is relevant as the

[3] https://github.com/Karanveer4/ArchitecturalThreatAnalysis

threat modeling methodology we base the workshop on, uses an DFD as a base they need to extend. Six participants had at least some knowledge about secure software design. Every participant had used the Eclipse environment before.

C. Limitations of the tool

In the interviews we asked our participants for the limitations of the tool and one main limitation that was mentioned was that the tool does not offer highlighting the path from where the asset starts to where it ends. Currently to track down the start of the asset and where it ends has to be done manually by looking into asset and which target it has. The second big limitation is that the tool does not add the assets automatically to end-to-end flows. The user has to manually add the asset to the elements until the asset target is reached. This can lead to errors in missing to add the asset to the flow and getting a wrong result in the end. One participant said that the properties could be added to multiple elements at once. Currently the user can only modify one element at a time. One participant said that the tool could be a standalone as the current platform (Eclipse) is not comfortable to use. The last limitation we found from interviews was that the tool does not provide any live collaboration. If the tool would provide a cloud feature where users can work together on the same diagram it would eliminate this limitation. Also, the suggestions for reductions that are output by the algorithms currently only look into areas with low and medium priority assets. This means that some possible reductions will not be suggested. Therefore, adding the implementation logic for reductions in high priority areas could be done in the future. As our developed tool is made in the Eclipse environment with several frameworks that have to be installed beforehand, some people may not use it due to this complexity. From interviews we have seen that some participants would prefer that the tool would be available as a standalone program. By having the tool standalone any compatibility issues with different systems could be fixed by gathering feedback from the users. It would also be difficult to introduce the tool to the industry as they might also prefer to have a separate program instead.

D. RQ1

To find out what is required to effectively support the automation of eSTRIDE we have researched related work. We developed a prototype tool using an approach that includes the requirements we elicited. The evaluation of the tool shows that the overall perception of the prototype tool was positive. The following sub-questions aim to answer specific concepts of automating eSTRIDE.

1) RQ1.1: To enable modeling and representation of eDFDs, one needs to take into account how to include a data structure for representing graphs, that show different relations and all the included elements. There is also a need for a graphical representation of the model, which should have the generally used representation styles for each diagram element. In the case of an eDFD it could mean for example that the processes are shown as a circle or an ellipse and an external entity would be a rectangle. There should also be a way to create and edit the models. To ensure the correctness of the model a validation process should be created. Our tool requires a meta-model for representing graphs, that shows all the possible components of an eDFD, how they are connected and what they consist of. The eDFD meta-model is made using the Eclipse Modeling Framework as an Eclipse Ecore model. This information is then used to allow the user to model and represent eDFDs with the help of Eclipse Sirius. The graphical eDFD model editor needs to allow making new eDFD elements and also to fill the properties and connections between them. Also, as work for a future iteration, we could use the Acceleo Query Language to validate the eDFD correctness.

2) RQ1.2: To answer RQ1.2 we saw that reductions are usually done to reduce threat explosion. Threat explosion is a problem incurred by security experts when performing threat analysis and they try to counter it by making abstractions. For example, experts often group similar elements of the DFD on the basis of the type of threats they are subject too. This technique is called reduction in regards to STRIDE methodology. To support eSTRIDE reductions, one requires to have an eDFD of the system. The eDFD should be enriched with extended notations like Assets marked with their security objectives and priorities, domain assumptions, domain properties and communication channel. One way abstraction can be achieved is by reducing the number of DFD elements based on some guidelines. There are two initial guidelines to provide abstraction, namely *bundling data flows* and *process folding*. The extended notation plays an important role in aiding the guidelines. We require to have algorithms implementing the mentioned guidelines. The prototype tool contains an algorithm that supports the eSTRIDE methodology and its abstractions. It also has a modelling tool to create DFDs and extend them to eDFDs. The tool fully automates the eSTRIDE mapping, which means that the user does not need to do any eSTRIDE threat categories documentation manually. The tool provides a suggestion list for abstractions, which contains process folding or data flow bundling. The tool does not automatically fold processes or bundle data flows, the user is still responsible to check the elements and do the folding or bundling in the tool manually. The tool also does not provide any attack scenarios, it only displays the threat categories associated with the entity. The security analyst still has to go through the threat category list and make possible attack scenarios.

E. RQ2

To answer RQ2 we have analyzed the data collected from the interviews conducted during our workshops. As we provide a modelling environment in our tool where the user can create an eDFD of a system, some new requirements were found here. Users wanted to have a feature in the tool that highlights various paths of an asset which can help them understand end-to-end flows throughout the system. They also wanted to have features that highlight errors and different priority values of an asset. Another requirement was found where the

user wants to add assets to end-to-end flows without repetitive clicking i.e. choosing different elements in the diagram and adding assets manually. They want to automate the process of adding assets. Once the asset has been created, it should automatically add itself to all the different elements between its source and target to complete end-to-end flow. Additional requirements were found where the user wanted our tool to be a standalone application and not as a plugin in the Eclipse environment. A cloud based feature was also mentioned which lets multiple users work on the tool simultaneously. At this point, the final list of threat categories are displayed on the console. Users wanted to have a new window in the tool which displayed the final results. Adding to that they wanted to sort threats according to threat categories, alphabetically or by type of entity. Based on our data, these are a few emergent requirements that are required for automating eSTRIDE.

F. RQ3

The prototype tool is able to produce data flow bundling and process folding suggestion list. The implemented algorithm checks the created eDFD and generates a formatted list with entities that can be bundled or folded. The tool does not make any modifications to the eDFD, the user is still responsible to make the suggested changes. The tool also automates eSTRIDE mapping. The user can run the eSTRIDE algorithm to get a formatted list of entities and the threat categories associated with it. The user does not need to do any manual mapping as the algorithm takes care of it.

1) RQ3.1: We have compared the results of an analysis performed by workshop participants to a ground truth to find the true positives, false positives and false negatives, which can be seen in Table II. This information was used to calculate the average precision, recall and productivity.

TABLE II
AMOUNT OF TRUE POSITIVES, FALSE POSITIVES AND FALSE NEGATIVES

Participant	TP	FP	FN	TIME (min)
1	91	31	35	48
2	85	24	39	55
3	68	22	59	59
4	55	37	72	45
5	50	11	75	51
6	55	22	71	47
7	70	21	56	55
8	65	31	59	58
9	95	17	35	60
10	103	31	23	52
μ	73.7	24.7	52.4	53
σ	18.6	7.8	18.2	5.2

Precision

Precision shows how many threat categories found by the participant are correct. The average precision of the participants was 74.8% with a standard deviation of 7.1%. The participant's own average perception on their precision was 53.5% (standard deviation 12.0%), which means that on average they had less confidence in the correctness of their results. The reason why participants might have thought that they were not as precise could be that they felt like they did not have enough knowledge about the domain. Also, when they compared their results to the ground truth they might have only checked part of them and this could have led them to perceive that they are only around 50% correct.

Compared to the literature, average precision was recorded at 60% Tuma and Scandariato [10] and 76% Scandariato et al. [11]. We can see that the prototype tool does not differ significantly in term of the measured precision. In addition, the related work measures security threats, whereas we measure security threat categories, therefore a direct comparison is not possible.

Recall

Recall shows how many threat categories go undetected by the participant. The average recall of the results was 58.5% (standard deviation 14.6%). The perceived recall of the participants was 60.3% (standard deviation 18.5%), which is very similar to the calculated average recall. When participants were asked to compare the results, sometimes they had fewer threat categories or less elements with threat categories elicited. This could have influenced the quite correct perception of their recall. However, the larger standard deviation could be because some participants were overly confident e.g. one said that they did not look over any threats. Overconfidence in the participants perception of their recall was also mentioned in Scandariato et al. [8].

We cannot make any direct comparisons with existing work as there are no studies yet on the recall of an analysis made with the eSTRIDE methodology. However, in a descriptive study on the similar STRIDE methodology they got an average recall of 0.36 (36%) Scandariato et al. [8]. Another study found the recall of STRIDE is 0.62 for the per element technique and 0.49 per interaction Tuma and Scandariato [10]. While comparing our results with these existing works, we still need to consider that they elicited threats in these studies, but we elicited threat categories. Nevertheless, our values are in good agreement with the aforementioned works, and can give insight on what the recall could be for threats deduced from an eSTRIDE analysis.

Productivity

The participants spent on average 53 minutes on the task. On average they produced 73.7 correct threat categories for the task. We compared the time spent with the amount of true positives and saw that the average productivity was 1.4 threat categories per minute (standard deviation 0.3). We asked the participants to compare the perceived productivity of both manual training and the tool. As a result, seven participants perceived the tool to be faster, mainly as the tool performs the threat category elicitation automatically.

The productivity of performing STRIDE per element is 3.5 correct security threats per hour Tuma and Scandariato [10]. By analyzing the amount of elements in the DFD, we can

estimate the productivity of performing the threat analysis. Also, in the paper Scandariato et al. [11] they present their results for productivity of performing threat analysis using STRIDE. They report that the average productivity is 1.2 threats per hour, thus even to analyze a small scale DFD, it will take days to finish the analysis.

As far as we know, there is no existing work on the productivity of an eSTRIDE analysis yet, but we could compare with work on the STRIDE methodology. As our tool provides eSTRIDE reduction suggestions, abstractions can be made on the model which results in simplifying the eDFD. After analyzing the HomeSys eDFD (ground truth) with our prototype tool it provided a suggestion list of different elements that could be bundled or folded. We see after applying the reduction suggestions we could reduce the size of the diagram from 65 total elements to 53 more relevant elements. As we have fewer elements to go through, finding the threat categories would consume less time. We can see in existing literature, that there is a correlation between how much time it takes in the threat category elicitation step and how much time a fully-fledged STRIDE analysis takes Scandariato et al. [8]. We could compare that thanks to these eSTRIDE reductions made on the DFD, completing a STRIDE analysis on it would take less time in total.

Also using our prototype tool, it would produce fewer threat categories compared to a STRIDE analysis, but as far as we know, there is no existing work on the productivity per threat category to compare with. We think this would be an interesting future work. After analyzing our ground truth with STRIDE, we would get 223 threat categories, but using the eSTRIDE reductions and mapping the amount of generated threat categories would be 123.

2) RQ3.2: We have organized two workshops to evaluate the usability of the tool. The workshop contained an introduction to the methodology and afterwards participants performed an example task using manual eSTRIDE to gain an understanding on how the methodology works. The second part of the workshop was to work with the tool to analyze a larger problem individually. After the workshop we conducted interviews where we had open questions for the usability of our tool, more specifically -

- ”How would you describe your overall experience with using the tool?”
- ”How clear would you say the tool displayed the final threat categories and suggestions?”
- ”How do you compare doing the diagram extensions manually and in the tool?”
- ”What features would you like to see in the tool that do not exist at this point?”

The questions consisted of multiple sub-questions to get more information about the usability of the tool. Seven participants had a good impression of the tool. Five participants were confused on how to use the tool in the beginning, but once they were familiar with the tool they said it was easy to use. Five participants noticed that the tool requires a lot of clicking for some of tasks that could be reduced to a couple of clicks instead. Five of the participants said that the tool provides an easier and more organized way of keeping track of the eDFD and changes made to it compared to the manual way. The tool allows the user to interact with the eDFD and freely manipulate the diagram, which provides a better overview of the eDFD when it gets larger. Moreover, four participants said that mapping assets to an end-to-end flow was the most difficult task. It required participants to go through each element individually while adding the assets that the flow is carrying. The easiest task for the participants while using the tool, was adding assets and their values as it wasn't very click-intensive.

In brief, by analyzing the data we found that the tool provides an easier and more organized way of keeping track of the different elements in the eDFD. Furthermore, the users had a positive impression on the usability of the tool once they got accustomed to using it. We also asked whether our participants would use the tool in the future, of which three participants said 'yes', whereas four participants would use it if they would have to perform threat analysis or work in a place where it is required.

VI. Conclusions

The purpose of this work was to study and understand how to support the semi-automation of the risk-first analysis methodology called eSTRIDE. To this aim, we have produced a prototype tool artifact as an Eclipse plugin. Automation tools for different methodologies are important as they can reduce the required resources. Therefore, our tool aims to automate the eSTRIDE analysis and to offer a graphical user interface for building eDFDs. Within our study, we defined requirements related to the graphical user interface for modeling and representation of eDFDs and to the analysis algorithms needed to support eSTRIDE.

From the workshops we found emergent requirements such as highlighting the asset path, a cloud feature for collaboration, adding values to multiple elements at once and to have asset values in different colors according to their priorities. Likewise, we found the average recall, productivity and precision of an analysis performed by the participants. Specifically, the average recall was 58.5%, average precision was 74.8% and the average productivity was 1.4 threat categories per minute.

The measured recall and precision did not differ significantly compared to manually performing STRIDE Scandariato et al. [11] Tuma and Scandariato [10]. The measured productivity was not directly comparable with other papers as we elicited threat categories, but they deduced threats Scandariato et al. [8] Tuma and Scandariato [10]. Nevertheless, when using our eSTRIDE reductions, the diagram would have less elements, which would result in less time compared to STRIDE analysis and therefore may lead to better productivity.

In our case, we had a total of 65 elements in the HomeSys DFD and after applying eSTRIDE methodology we got 12 reduction suggestions which reduced the number of elements to 53. Additionally, performing a STRIDE analysis on our

ground truth would generate 223 threat categories, but whilst using our prototype tool it generates 123 threat categories.

Finally, we studied the perceived usability of the tool. According to the perception of the participants, the tool provides an easier and more organized way to track the diagram, is perceived to be faster and reduces human error compared to manual analysis.

This prototype tool is the first attempt of semi-automating eSTRIDE. The participants of our evaluation mentioned several ways to improve the tool in future work. For instance, possible improvements could be eDFD validation or the development of the aforementioned emergent requirements.

Acknowledgment

The authors would like to thank Katja Tuma for the invaluable support she provided throughout the study presented in this thesis work.

References

[1] P. H. Meland and J. Jensen. Secure software design in practice. In *2008 Third International Conference on Availability, Reliability and Security*, pages 1164–1171, March 2008. doi: 10.1109/ARES.2008.48.

[2] L. Williams, G. McGraw, and S. Migues. Engineering security vulnerability prevention, detection, and response. *IEEE Software*, 35(5):76–80, Sep. 2018. ISSN 0740-7459.

[3] Adam Shostack. *Threat modeling: Designing for security*. John Wiley & Sons, 2014.

[4] Daniela Cruzes, Martin Jaatun, Karin Bernsmed, and Inger Anne Tondel. Challenges and experiences with applying microsoft threat modeling in agile development projects. pages 111–120, 11 2018. doi: 10.1109/ASWEC.2018.00023.

[5] Laurens Sion, Koen Yskout, Dimitri Van Landuyt, and Wouter Joosen. Solution-aware data flow diagrams for security threat modeling. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pages 1425–1432. ACM, 2018.

[6] Mass Soldal Lund, Bjornar Solhaug, and Ketil Stolen. *Model-driven risk analysis: the CORAS approach*. Springer Science & Business Media, 2010.

[7] Katja Tuma, Gul Calikli, and R Scandariato. Threat analysis of software systems: A systematic literature review. *Journal of Systems and Software*, 144, 06 2018. doi: 10.1016/j.jss.2018.06.073.

[8] Riccardo Scandariato, Kim Wuyts, and Wouter Joosen. A descriptive study of microsoft's threat modeling technique. *Requir. Eng.*, 20(2):163–180, June 2015. ISSN 0947-3602. doi: 10.1007/s00766-013-0195-2. URL http://dx.doi.org/10.1007/s00766-013-0195-2.

[9] Katja Tuma, Riccardo Scandariato, Mathias Widman, and Christian Sandberg. Towards security threats that matter. In *Computer Security*, pages 47–62. Springer, 2017.

[10] Katja Tuma and Riccardo Scandariato. Two architectural threat analysis techniques compared. In Carlos E. Cuesta, David Garlan, and Jennifer Pérez, editors, *Software Architecture*, pages 347–363, Cham, 2018. Springer International Publishing. ISBN 978-3-030-00761-4.

[11] Riccardo Scandariato, Kim Wuyts, and Wouter Joosen. A descriptive study of microsoft's threat modeling technique. *Requirements Engineering*, 20(2): 163–180, Jun 2015. ISSN 1432-010X. doi: 10.1007/s00766-013-0195-2. URL https://doi.org/10.1007/s00766-013-0195-2.

[12] Kim Wuyts, Riccardo Scandariato, and Wouter Joosen. Empirical evaluation of a privacy-focused threat modeling methodology. *Journal of Systems and Software*, 96: 122–138, 2014.

[13] Tony UcedaVelez and Marco M Morana. *Risk Centric Threat Modeling: process for attack simulation and threat analysis*. John Wiley & Sons, 2015.

[14] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing. Automated generation and analysis of attack graphs. In *Proceedings 2002 IEEE Symposium on Security and Privacy*, pages 273–284, May 2002. doi: 10.1109/SECPRI.2002.1004377.

[15] Sjouke Mauw and Martijn Oostdijk. Foundations of attack trees. volume 3935, pages 186–198, 07 2006. doi: 10.1007/11734727_17.

[16] L. Sion, D. Van Landuyt, K. Yskout, and W. Joosen. Sparta: Security privacy architecture through risk-driven threat assessment. In *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 89–92, April 2018. doi: 10.1109/ICSA-C.2018.00032.

[17] Bernhard J. Berger, Karsten Sohr, and Rainer Koschke. Automatically extracting threats from extended data flow diagrams. In Juan Caballero, Eric Bodden, and Elias Athanasopoulos, editors, *Engineering Secure Software and Systems*, pages 56–71, Cham, 2016. Springer International Publishing. ISBN 978-3-319-30806-7.

[18] Alan Hevner, Alan R, Salvatore March, Salvatore T, Park , Jinsoo Park, Ram , and Sudha . Design science in information systems research. *Management Information Systems Quarterly*, 28:75–, 03 2004.

[19] Ken Peffers, Tuure Tuunanen, Marcus Rothenberger, and S Chatterjee. A design science research methodology for information systems research. *Journal of Management Information Systems*, 24:45–77, 01 2007.

[20] Lab material - empirical study: Threat modeling. https://sites.google.com/site/empiricalstudythreatanalysis/define. [Online; accessed 28-May-2019].

[21] Johan Linker, Sardar Sulaman, Martin Host, and Rafael de Mello. Guidelines for conducting surveys in software engineering. 05 2015.

[22] B.M. Wildemuth. *Applications of Social Research Methods to Questions in Information and Library Science, 2nd Edition*. ABC-CLIO, 2016. ISBN 9781440839054. URL https://books.google.se/books?id=uv98DQAAQBAJ.

[23] Claes Wohlin, Per Runeson, Martin Host, Magnus C Ohlsson, Bjrn Regnell, and Anders Wesslen. *Experi-*

mentation in software engineering. Springer Science & Business Media, 2012.

[24] Mikael Svahnberg, Aybke Aurum, and Claes Wohlin. Using students as subjects - an empirical evaluation. pages 288–290, 01 2008. doi: 10.1145/1414004.1414055.

APPENDIX
INTERVIEW QUESTIONS

Perception of usability of the tool:

1) Have you used threat analysis in any of your previous projects ?
    a) If yes, how do you compare using a threat analysis methodology to just eliciting security related requirements?
    b) If no, is there any particular reason you never used it before?
2) How would you describe your overall experience with using the tool?
    a) What was the most difficult part of performing the task with the tool?
    b) What was the easiest part of performing the task with the tool?
    c) While using the tool were you ever confused how to use it? Where/Why?
    d) How do you compare using the tool with using common sense to identify threats?
3) How clear would you say the tool displayed the final threat categories and suggestions?
    a) Do you have any suggestions on how to improve displaying the categories and suggestions?
4) How do you compare doing the diagram extensions manually and in the tool?
    a) How do you compare doing the abstractions manually and in the tool? (if applicable)
    b) How do you compare doing the eSTRIDE analysis manually and in the tool?
    c) How do you compare the precision of the manual analysis to the tool?
    d) How do you compare the recall of the manual analysis to the tool?
    e) How do you compare the productivity of the manual analysis to the tool?
5) What features would you like to see in the tool that do not exist at this point?
    a) Do you see any limitations in this tool? If yes, What are they?
6) Do you think you would use the tool in the future? Why?
7) Is there anything else you would like to say before we end the interview?

Perception of precision, recall and productivity:

1) Compared to the ground truth, from 0-100 percent how precise do you think your results are? Why?
2) Compared to the ground truth, from 0-100 percent, how many threats do you think your results overlooked? Why?
3) What threats did you miss and why?
4) What threats are incorrect and why?

DEVELOPED ARTIFACT

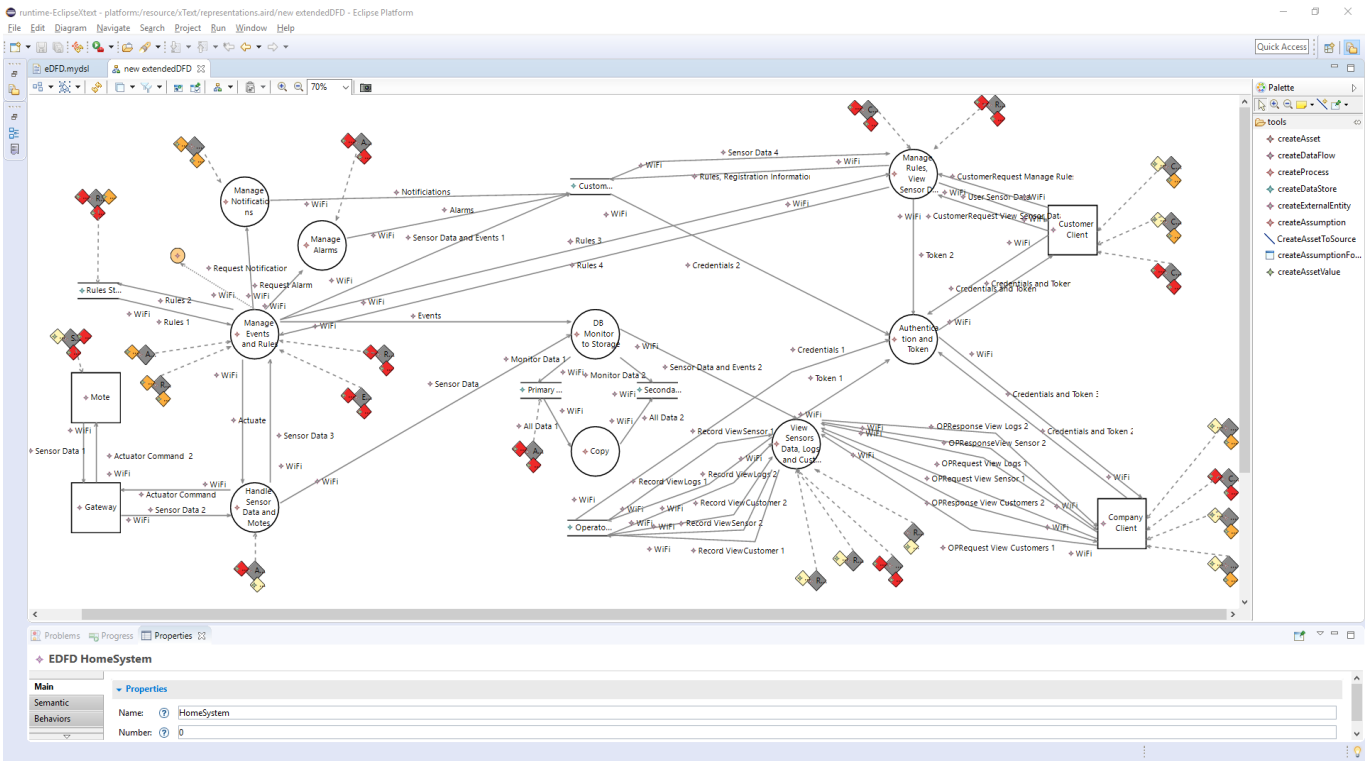

Fig. 2. Graphical Modelling Environment

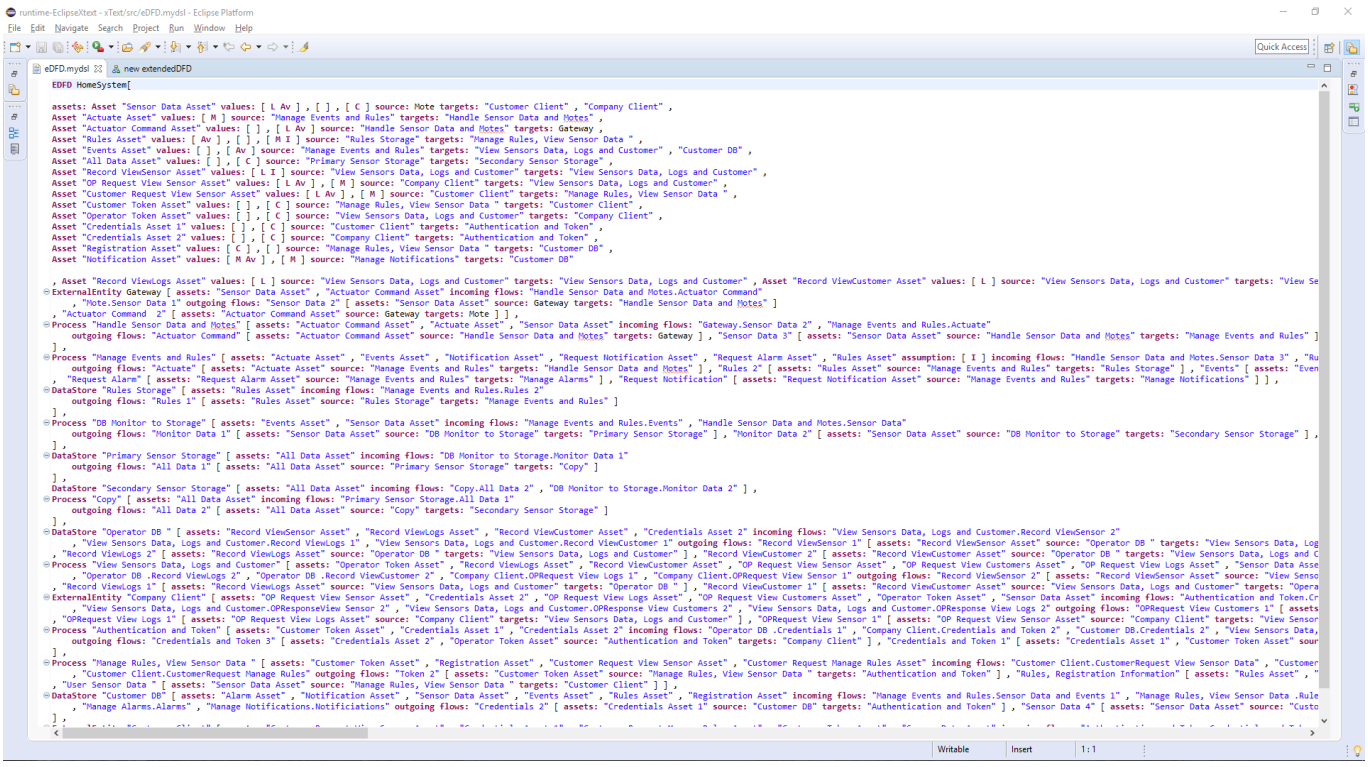

Fig. 3. Textual Modelling Environment

```
Process Folding Suggestions                        |
---------------------------------------------------
Manage Events and Rules       Manage Notifications


Data Bundling Suggestions:
                             Entity                                         |                                                                   Flows
---------------------------------------------------------------------------------------------------------------------------------------------------
Customer Client                                                             | [CustomerRequest View Sensor Data, CustomerRequest Manage Rules]
Company Client                                                              | [OPRequest View Customers 1, OPRequest View Sensor 1, OPRequest View Logs 1]
View Sensors Data, Logs and Customer                                        | [Record ViewLogs 1, Record ViewCustomer 1, Record ViewSensor 2]
Operator DB                                                                 | [Record ViewSensor 1, Record ViewCustomer 2, Record ViewLogs 2]


0 Suggestion List, 1 eSTRIDE, 2 exit
1
eSTRIDE Threat Table:

                             Entity                                         |                                                         STRIDE Category
---------------------------------------------------------------------------------------------------------------------------------------------------
Sensor Data and Events 1                                                    | [Tampering, Information Disclosure, Denial of Service]
Sensor Data and Events 2                                                    | [Tampering, Information Disclosure, Denial of Service]
Actuator Command  2                                                         | [Tampering]
Actuate                                                                     | [Tampering]
OPResponse View Logs 2                                                      | [Tampering, Information Disclosure]
Handle Sensor Data and Motes                                                | [Spoofing, Elevation of Privilege, Tampering, Information Disclosure]
Mote                                                                        | [Spoofing]
User Sensor Data                                                            | [Tampering, Information Disclosure]
Manage Rules, View Sensor Data                                              | [Spoofing, Elevation of Privilege, Tampering, Information Disclosure]
OPRequest View Logs 1                                                       | [Tampering]
Rules, Registration Information                                             | [Tampering, Information Disclosure, Denial of Service]
Token 1                                                                     | [Tampering, Information Disclosure]
Token 2                                                                     | [Tampering, Information Disclosure]
Customer DB                                                                 | [Tampering, Information Disclosure, Denial of Service]
Credentials and Token 2                                                     | [Tampering, Information Disclosure]
Rules 4                                                                     | [Tampering, Denial of Service]
Rules 3                                                                     | [Tampering, Denial of Service]
Request Alarm                                                               | [Tampering, Denial of Service]
Credentials and Token 3                                                     | [Tampering, Information Disclosure]
Actuator Command                                                            | [Tampering]
Rules 2                                                                     | [Tampering, Denial of Service]
OPResponse View Customers 2                                                 | [Tampering, Information Disclosure]
Rules 1                                                                     | [Tampering, Denial of Service]
Credentials and Token 1                                                     | [Tampering, Information Disclosure]
Copy                                                                        | [Spoofing, Elevation of Privilege, Tampering, Information Disclosure]
Primary Sensor Storage                                                      | [Tampering, Information Disclosure]
Authentication and Token                                                    | [Spoofing, Elevation of Privilege, Tampering, Information Disclosure]
All Data 2                                                                  | [Tampering, Information Disclosure]
All Data 1                                                                  | [Tampering, Information Disclosure]
Sensor Data                                                                 | [Tampering, Information Disclosure]
Monitor Data 2                                                              | [Tampering, Information Disclosure]
Operator DB                                                                 | [Tampering, Information Disclosure]
Monitor Data 1                                                              | [Tampering, Information Disclosure]
Sensor Data 4                                                               | [Tampering, Information Disclosure]
Request Notification                                                        | [Tampering, Denial of Service]
Sensor Data 1                                                               | [Tampering, Information Disclosure]
Sensor Data 2                                                               | [Tampering, Information Disclosure]
Sensor Data 3                                                               | [Tampering, Information Disclosure]
```

Fig. 4. Threat Category Documentation

SURVEY QUESTIONS

Age

- ◯ Under 18
- ◯ 18-20
- ◯ 21-25
- ◯ 26-30
- ◯ 31+
- ◯ Other: _____

Education (Highest level obtained)

- ◯ High School
- ◯ Bachelors Degree
- ◯ Masters Degree
- ◯ Phd Degree
- ◯ Other: _____

Occupation

- ☐ Student
- ☐ Software Developer
- ☐ Security Engineer
- ☐ Software Architect
- ☐ Tester
- ☐ Other: _____

Years of experience in Software Engineering

- ◯ <1
- ◯ 1-3
- ◯ 3-5
- ◯ 5-10
- ◯ 10+
- ◯ None

Fig. 5. Survey Questions

Years of experience in Software Architecture

- ◯ <1
- ◯ 1-3
- ◯ 3-5
- ◯ 5-10
- ◯ 10+
- ◯ None

Years of experience in Threat Analysis

- ◯ <1
- ◯ 1-3
- ◯ 3-5
- ◯ 5-10
- ◯ 10+
- ◯ None

Have you used any Threat Analysis methodologies before?

- ◯ Yes
- ◯ No

If you answered "Yes", which Threat Analysis methodologies have you used?

Your answer

Have you used any semi-automation tools for Threat Analysis before?

- ◯ Yes
- ◯ No

If you answered "Yes", which Threat Analysis semi-automation tools have you used?

Your answer

Fig. 6. Survey Questions

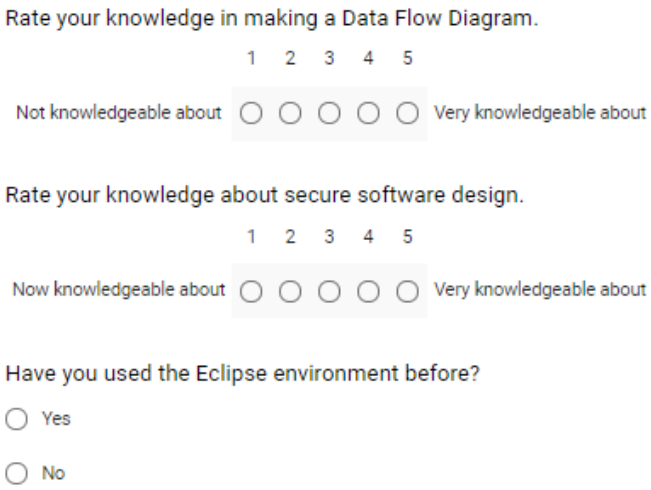

Fig. 7. Survey Questions