Technische Universität München
Department of Informatics

Masters's Thesis in Informatics

Investigation of tool based modeling techniques for safety and security critical systems

Aleksejs Voroncovs

Technische Universität München

Department of Informatics

Masters's Thesis in Informatics

Investigation of tool based modeling techniques for safety and security critical systems

Analyse von Tool basierten Modellierungsansätzen für sicherheitskritische Systeme

Author	Aleksejs Voroncovs
Supervisor	Prof. Dr.-Ing. Georg Carle
Advisor	Dipl.-Inf. Stephan-A. Posselt
Date	March 25, 2016

Informatik VIII
Chair for Network Architectures and Services

I confirm that this thesis is my own work and I have documented all sources and material used.

Garching b. München, March 25, 2016

Signature

Abstract

Increasing complexity of computer systems raises the amount of possible vulnerabilities in those systems. Thus, providing security becomes a more challenging goal, especially for safety and security critical systems, such as aircraft cabin core system. In order to make sure that a system is secure, experts do a risk analysis of the system, using a special methodology.

Currently, there are many methodologies that provide instructions, how do make a risk analysis. Some of them are freely available in the Internet, some of them are proprietary ones, like the one used at Airbus Group for assessing risks in the cabin core system. This methodology describes the algorithm of actions regarding the identified threats, but the responsibility for identified threats remains fully on security experts.

Until now, risk analysis of the cabin core system has been made manually by Airbus Group experts. Due to its manual nature, risk analysis process is time consuming and error-prone. Therefore, in this thesis an opportunity of automating threat detection is investigated.

There are multiple tools for threat modeling available now in the Internet. Four examples of such tools are: Practical Threat Analysis (PTA), Trike, Microsoft Threat Analysis and Modeling v2.1 and Microsoft Threat Modeling Tool 2016. The idea of how these tools work are presented. In addition, all tools are examined with respect to their application for modeling threats in the aircraft cabin core system.

None of the existing tools is capable to model end-to-end threats, which is necessary for risk assessment in the cabin core system. Hence, a new solution has to be built. A concept of the tool chain, which was developed during this work, offers an automated approach to end-to-end threat modeling. The developed concept assumes the usage of Microsoft Threat Modeling Tool 2016 and new threat modeling tool that is aimed to identify end-to-end threats in the system model.

The insights gained by this thesis can be applied to implement the framework for automating threat identification during risk assessment. Additionally, the developed framework is critically evaluated, providing information about disadvantages of the whole approach and about feasibility of used algorithms.

Zusammenfassung

Steigende Komplexität der Rechnersystemen ehöht die Anzahl der möglichen Schwachstellen in diesen Systemen. Folglich, die Versorgung der Sicherheit wird ein herausforderndere Ziel, insbesondere für sicherheitskritische Systeme, wie z. B. Kabinenkernsystem am Flugzeug. Um die Sicherheit zu gewährleisten, führen die Experten eine Risikoanalyse des Systems mit Hilfe der speziellen Methodologie durch.

Heutzutage gibt es viele Methodologien, die Instruktionen bieten, wie man eine Risikoanalyse durchführen soll. Manche Methodologien sind kostenlos verfügbar im Internet, manche sind privat, wie z. B. Methodologie, die bei Airbus Group eingesetzt wird um die Risiken im Kabinenkernsystem zu bewerten. Diese Methodologie beschreibt den Algorithmus von Maßnahmen bezüglich der gefundenen Bedrohungen, aber die Verantwortung für identifizierten Bedrohungen haben nur die Sicherheitsexperten.

Bisher wurde die Risikoanalyse des Kabinenkernsystems bei Airbus Group Experten gemacht. Wegen der manuellen Herangehensweise ist dieses Prozess zeitaufwendig und fehleranfällig. Deswegen wird die Möglichkeit der Automatisierung der Risikoanalyse in dieser Masterarbeit untersucht.

Es gibt mehrere Tools für Bedrohungsmodellieren, die jetzt im Internet verfügbar sind. Vier Beispiele von solchen Tools sind: Practical Threat Analysis (PTA), Trike, Microsoft Threat Analysis und Modeling v2.1 and Microsoft Threat Modeling Tool 2016. Die Vorgehensweise, wie diese Tools funktionieren, sind vorgestellt. Zudem wurden alle Werkzeuge bezüglich ihrer Anwendung auf Modellieren von Bedrohungen im Kabinenkernsystem überprüft.

Keiner der bestehenden Werkzeuge ist fähig, die Ende-zu-Ende Bedrohungen zu modellieren, was für die Risikoanalyse im Kabinenkernsystem notwendig ist. Deshalb muss eine neue Lösung entwickelt werden. Ein Konzept der Werkzeugkette, das während dieser Masterarbeit entwickelt wurde, bietet einen automatisierten Ansatz zum Modellieren der Ende-zu-Ende Bedrohungen. Das entwickelte Konzept setzt die Nutzung des Werkzeugs "Microsoft Threat Modeling Toolünd eines neuen Tools voraus. Das neue Tool fokussiert sich auf Identifizierung von Ende-zu-Ende Bedrohungen im Systemmodell.

Die von dieser Masterarbeit erworbenen Erkenntnisse können auf die Implementierung des Frameworks für automatisierte Bedrohungsidentifizierung angewandt werden. Außerdem wird das entwickelte Framework kritisch bewertet, Informationen über die Nachteile des ganzen Ansatzes und über die Realisierbarkeit der verwendeten Algorithmen werden auch gegeben.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays modeling is being considered as a useful approach to solving many kinds of problems. One of the application areas for modeling is information security. Up until today, one of the major modeling applications for information security is threat modeling. Threat modeling is a structured and methodical approach that allows to identify threats that are potential for the analyzed information system, classify them by risk, and prioritize mitigation efforts based on the impact, which is posed by those threats.

Security is crucial in security- and safety-critical systems, such as ones on the board of an aircraft. Evolving nature of the technological requirements on the board results in the higher complexity of methods and techniques that are applied to creating functional onboard systems. To ensure a safe, secure, reliable and efficient air transportation system with high capacity, hardware and software security of the airplane's onboard systems must be ensured [14]. Nowadays, aircraft manufacturers and their avionics system vendors are responding to the need for onboard information technology with sophisticated, networked aircraft information systems. Due to the high integration of the airplanes with wired and wireless technologies the airplanes are neither completely regulated nor isolated from external network access. Therefore, the probability of new vulnerabilities that may open access to onboard systems and disturb their operation grows. For this reason security measures have to be introduced. Aircraft information security is necessary to mitigate the risk of external and internal attacks to an acceptable level, to protect aircraft information systems, and to protect the confidentiality, integrity, and availability of information processed by those systems.

In order to provide reliable security mechanisms it is necessary to assure that the used techniques are indeed secure and do not allow the attackers to compromise the system. One way to do this is carrying out a risk assessment of the system. Risk assessment is the process to ensure that the equipment/system is protected from attacks on data and interfaces, both intentional and unintentional. It also includes the threat occurrence

probability estimation, evaluation of the impact arisen from those threats and risk mitigation proposals. The main goal of the risk assessment is to find out all information security risks that are present in the system, so that an organization could use the output of this analysis for introduce countermeasures to eliminate the undesired impact. Risk assessment can be accomplished in multiple ways.

Currently there are many known methodologies for risk analysis due to the fact that no worldwide accepted standards for threat analysis exist. In fact, there exist different risk assessment methodologies, which are suitable for different kinds of systems, and they are widely used. For instance, the one that is used by Airbus Group experts for assessing security risks in cabin avionics systems is a proprietary methodology that was developed internally. Risk assessment methodologies provide the user a guide for classifying, evaluating and processing of existing threats, but threats themselves have to be detected manually. The main drawback is the lack of automation. Automating phases of the risk assessment process would be beneficial in terms of time, flexibility and universality, meaning that the automated approach to risk assessment would faster, would be suited for different methodologies and would be used by different experts. Hence, the possibility of threat analysis automation by making use of threat modeling should be investigated.

1.1 Goals

In this Master's Thesis threat modeling is considered as a potential approach to automated risk assessment. The main goal of this work is to find an appropriate threat modeling approach in order to automate the risk assessment process for the aircraft cabin system. Therefore, this thesis examines the current state of the art in field of threat modeling and gives an overview of existing solutions. As long as the main subject of the research is risk assessment in aircraft cabin systems, the applicability of existing solutions to modeling airborne systems is also studied.

The theoretical background of risk assessment methodologies and current developments in the field of threat modeling must be investigated in order to find out to what extent they satisfy the needs of risk analysis in aircraft systems. Finally, based on the requirements for the airborne system threat analysis, a concept of automated threat detection has to be proposed.

1.2 Outline

Chapter 1: Introduction

Chapter 1 introduces the motivation behind the research, mentions actual problems and defines the goals of the work.

Chapter 2: Background

In chapter 2 the background knowledge about risk assessment and existing threat analysis methodologies is provided. Then, the motivation for using threat modeling tools for assessing risks and an overview of existing threat modeling solutions is given. In the end of the chapter two threat modeling approaches are defined and the choice of the most suitable one for assessing risks in aircraft systems is explained.

Chapter 3: Related work

In chapter 3 studied scientific publications, which are related to the topic of this thesis, are enumerated.

Chapter 4: High-level concept

Chapter 4 proposes a tool chain for addressing automated threat modeling. As long as one of the existing threat modeling Tools, namely Microsoft Threat Modeling Tool, is considered as partially suitable for modeling threats in airborne systems, it is adopted as an important part of the new framework. However, it does not fully satisfy the actual needs, and hence, the missing capabilities of the designed workflow are defined. In order to define the requirements for the missing functionality, an additional step called "Threat mapping" was necessary. Threat mapping is described in the end of the chapter.

Chapter 5: Low-level concept

In order to provide the missing capabilities of the previously developed framework an additional tool has to be implemented. Based on the requirements defined during the threat mapping the low-level concept was developed and described in chapter 5.

Chapter 6: Evaluation

Chapter 6 assesses the accomplished work by a critical review of the developed concept. The results of threat mapping are analyzed, as well as advantages and disadvantages of the new approach are enumerated.

Chapter 7: Conclusion

Chapter 7 gives a summary of the entire research and an overview of future work that is yet to be done for this topic.

Chapter 2

Background

In this chapter background knowledge about the field of research is provided. This includes the introduction to the automated risk assessment, involving threat modeling approach into the risk analysis and an overview of the existing methods to carry out this process automatically.

As information technology including network systems has developed substantially, information security has taken an important role in the functioning of those systems. Security is a critical requirement in the software industry. In general, security refers to the confidentiality, integrity, availability and non-repudiation of a system and its data [1]. Nowadays, information security has become crucial for organizations to enable successful operation of different mission critical applications. It is even more important in security and safety critical functions, such as those fulfilled by systems on the board of an aircraft.

On-board computer systems on the aircraft appear to be sensitive because of the demand on high efficiency and predictability [2]. Each security vulnerability may cause not only financial losses to the airline and aircraft manufacturing companies, but may also be health- and life-threatening for passengers and cabin crew.

Risk assessment or risk analysis is the process of identifying the security risks to a system and determining their probability of occurrence, their impact, and the safeguards that would mitigate that impact. Risk assessment is one step in the process of risk management. The main problem in risk assessment is how to assess all risks in a system/organization so that by using the output of risk assessment, these organizations could define appropriate measures for reducing or eliminating those risks [3].

Proper techniques in risk assessment are significant for taking appropriate countermeasures against various security threats to the organizational assets. Prior to taking countermeasures against security threats a risk analysis has to be done, so that security framework arrangement is systematic and reasonable. Important goals of the security

analysis are to:

- List possible threats - an expert has to decide, what threats can be exploited in the system.
- Estimate the potentiality of threats - an expert has to specify a numerical value that represents, how likely each threat is to happen. These numerical values will then be used in order to decide, which countermeasures have to be taken.
- Evaluate the impact on the organization in case of successful exploitation of corresponding threats.

After that, concrete security measures have to be proposed. Using the output of the risk assessment, organizations can define appropriate measures for reducing or eliminating those risks.

Risk analysis may vary according to the requirements of organizations [4]. The changing nature of threats and vulnerabilities make the understanding of risk, its assessment and management complicated. Hence, a tool, that conducts some steps of the risk assessment automatically, would simplify the workflow for the security team.

2.1 Methodologies

The highly diverse nature of network-based computer systems requires different methods of analyzing security risks. Risk assessment involves risk identification, risk analysis, and risk prioritization. A risk can be defined by the relationship of risk probability and risk impact. Usually, the impact of risk incidents refers to the loss caused by risk incidents, and the loss can be measured by the value of assets [1]. Currently there is no standard for risk assessment [3]. There are many methods that have been developed by many organizations for risk analysis. Some of those are presented in this section in order to provide an idea of their structure.

2.1.1 MEHARI

MEHARI (Method for Harmonized Analysis of Risk) is a free, open source information risk analysis assessment and risk management method, developed, maintained and distributed by CLUSIF – Club de la Sécurité de l'Information Français [5]. According to CLUSIF, MEHARI is a way to manage information security for any type of organization through the provision of a methodological framework, tools, modular components and knowledge bases. The methodology consists of four modules:

- Stake and asset analysis – classification
- Security service audit

- Identification of critical risks
- Risk situation analysis

In the first stage stakes in managing security in the analyzed system are found out. It means that security experts identify possible malfunctions of the system, classify them by likelihood and impact, and try to estimate the value of these malfunctions.

During the second stage the quality of security services is evaluated. Three major aspects of security services are taken into account:

- Efficiency – a measure of ability of security services to effectively ensure the required function faced with competent users or unusual circumstances.
- Robustness – an ability of a security service to resist an action that is intended to bypass this service, or to restrict its effectiveness.
- Permanency – a guarantee that a particular security service will ensure the proper functioning of the system over time.

In the third stage those risks, which are critical for the functionality of a system, have to be identified. MEHARI provides two approaches of identifying critical risk situations. The first approach is direct and it uses the malfunction value scale. Each type of the malfunction, identified during the first stage, is mapped to threat scenarios that have been identified by finding possible causes of the malfunction. According to MEHARI, malfunction values, such as threat potentiality and impact, have to be provided by the expert, who uses this methodology to carry out risk analysis. All of the scenarios with a high level of consequences (levels 3 or 4) should be considered as critical and examined in further detail. The second approach comprises systematic threat identification using the knowledge base and automated procedures, which are provided by MEHARI. The two approaches are complementary and should be run concurrently.

The goal of the fourth stage is to evaluate two characteristic parameters of risk being run by the organization, supposing that the scenario occurs. These parameters are:

- The potentiality of the risk – a qualitative representation of the probability that a certain risk occurs, expressed with a number from 0 (not considered due to unfeasibility) to 4 (very likely).
- The impact of the risk on the organization – a representation of seriousness of the direct and indirect consequences, if a certain risk occurs.

In the end of risk analysis the expert has to examine the obtained results and decide whether the system is secure enough, i.e. the potentiality values of threat scenarios are satisfactory for the normal functioning of the analyzed system. If the values are not acceptable, the expert has to propose security countermeasures, review and update the results of the risk analysis once the proposed countermeasures are implemented.

The output of the risk analysis using MEHARI is represented in the tabular form, where all possible threat scenarios are enumerated and the values of threat likelihood/impact are provided. A fragment of such an output is provided in figure 2.1.

DESCRIPTION	Intrinsic values			security measures				decided values			calculated values			
	Impact	Exposure	seriousness	Dissuasion	Prevention	Confining	Palliation	Confinability	Impact	Likelihood	Impact	Likelihood	seriousness	Accept (A) or transfer (T)
Accidental erasure of files of data, due to a production incident	2	3	**2**	1	1	1	3	1	4		2	3	4	T
Erasure, due to an error, of files of data, by a user authorized legitimately, connected from the internal network	2	3	**2**	1	1	1	3	0		4	2	3	3	A
Erasure, due to an error, of files of data, by a user authorized illegitimately, connected from the internal network	2	3	**2**	1	1	1	3	0			2	3	2	

Figure 2.1: MEHARI output table

The output table contains values regarding all aspects of threat scenario execution, including values of introduced security measures, which decrease the overall seriousness of the attacks. In the last column the expert has to specify, whether the risk of the corresponding threat is acceptable or not. If not, additional security measures have to be proposed, and after that the results of risk assessment have to be revised.

2.1.2 OCTAVE

OCTAVE is a methodology for identifying and evaluating information security risks. It is intended to help an organization to:

- Develop qualitative risk evaluation criteria that describe the organization's operational risk tolerances
- Identify assets that are important to the mission of the organization
- Identify vulnerabilities and threats to those assets
- Determine and evaluate the potential consequences to the organization if threats are realized

The conceptual framework that formed the basis of the original OCTAVE approach was published by the Software Engineering Institute (SEI) at Carnegie Mellon University in 1999 [6]. The OCTAVE methodology consists of eight steps, the relationships between these steps are illustrated in figure 2.2.

During the first step a set of risk measurement criteria has to be defined. Risk measurement criteria are a set of user-defined qualitative measures against which the effects of a

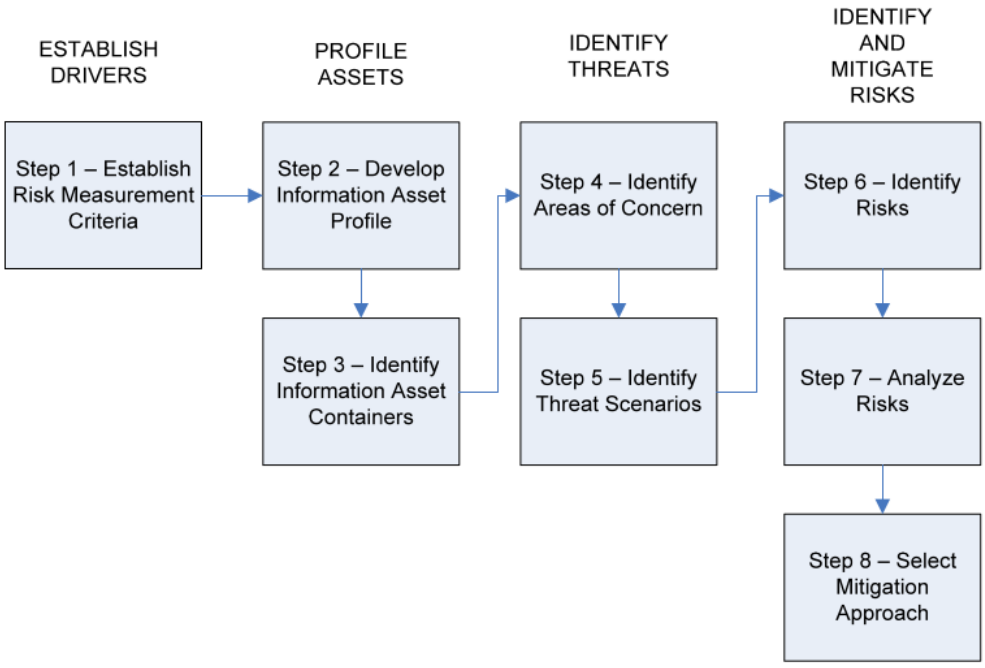

Figure 2.2: OCTAVE workflow

realized risk can be evaluated. The criteria must contain all types of risks, against which threats are going to be evaluated. For example: reputation among staff, reputation among customers, reputation in the community. Possible values for measuring impact of threats against aforementioned criteria must be also provided. An example for this is shown in figure 2.3.

Impact Area	Low	Medium	High
Legal penalty, data privacy violation	Less than 5% cost of typical yearly revenue.	5% to 10% cost of typical yearly revenue.	More than 10% cost of typical yearly revenue.

Figure 2.3: OCTAVE step 1

Step 2 stands for the development of information asset profile. During this step the experts have to define assets of the system and describe their features, qualities, characteristics and value. The profile for each asset is captured on a single worksheet that forms the basis for the identification of threats and risks in subsequent steps. An example of asset definition is shown in figure 2.4.

During the third step information asset containers have to be identified. Containers describe the places where information assets are stored, transported, and processed. Information assets reside not only in containers within an organization's boundaries but they also often reside in containers that are not in the direct control of the organization. Any risks to the containers in which the information asset lives are inherited by the

Information Asset	Rationale for Selection	Description	Owner	Confidentiality	Integrity	Availability
Key material	Leakage will allow access to meter device.	Refers to all secrets stored in the meter device.	Device manufacturer and meter mgmt. personnel	**Key material must be kept secret.**	Only the utility shall be granted to update and revert key material.	Key material must be available for meter mgmt. personnel.

Figure 2.4: OCTAVE step 2

information asset. An example of asset container definition is shown in figure 2.5.

Container	Description	Owner	Type	Class
Meter	Holds various assets. E.g. key material	Metering company	Internal	Technical
Monthly paper invoice	Consumption data on monthly invoice	Utility, Consumer	External	Physical
Service technician	Knows the initial secret of meters	Service company	External	People

Figure 2.5: OCTAVE step 3

Step 4 begins the risk identification process by brainstorming about possible conditions or situations that can threaten an organization's information asset and storing them in the tabular form. These scenarios are referred to as areas of concern and may represent threats and their corresponding undesirable outcomes. An example of identifying areas of concern is shown in figure 2.6.

Area of Concern	Actor	Means	Motive	Outcome
Inadequate link encryption could allow to access metering values.	Investigative Journalists	Put a tap on the link	gain information on energy use monitor consumption behaviour	Disclosure

Figure 2.6: OCTAVE step 4

In step 5 possible threat scenarios are identified and stored in the tabular form. Threat scenarios can arise both from the information from previous steps, as well as experts can define additional independent threat scenarios based on their knowledge.

In Step 6 the consequences to an organization if a threat is realized are captured, completing the risk picture. A threat can have multiple potential impacts on an organization. The activities involved in this step ensure that the entire impact of executed threat scenarios is covered. An example of identifying threat impact is shown in figure 2.7.

In Step 7 of the assessment, a simple quantitative measure of the extent, to which the organization is impacted by a threat is estimated. This relative risk score is derived by considering the extent to which the consequence of a risk impacts the organization against the relative importance of the various impact areas. Figure 2.8 gives an example of risk score calculation.

In Step 8 organizations determine which of the risks they have identified require mitigation and develop a mitigation strategy for those risks. This is accomplished by first

Area of Concern	Actor	Outcome	Consequence
Inadequate link encryption could allow to access metering values.	Investigative Journalists	Disclosure	Disclosure of private information leads to legal penalty. The legal department estimates the total case at £ 500'000.

Figure 2.7: OCTAVE step 6

Impact Area	Rank	Impact	Value	Score
Fines/Legal Penalties	5	High	3	15
Reputation	4	High	3	12
Safety and Health	3	Medium	2	6
Productivity	2	Medium	2	4
Financial	1	Low	1	1
Total Risk Score				**38**

Figure 2.8: OCTAVE step 7

prioritizing risks based on their relative risk score. Once risks have been prioritized, mitigation strategies are developed [6].

Generally, the OCTAVE methodology does not contain any formal requirements for storing and displaying all the information collected and produced during all eight steps. Still, there are examples for doing it in the methodology documentation.

2.1.3 Risk assessment methodologies - summary

There are many other risk assessment methodologies, which are freely available in the Internet. For example: Magerit [7], NIST 800-30 [8] and Microsoft Security Risk Management Guide [9]. There are also multiple proprietary methods that have been developed by various organizations internally for risk analysis specifically for the needs of those organizations. An example of such a proprietary risk assessment methodology is an "Aircraft Information Security Risk Assessment" methodology [10] designed internally at Airbus Group Innovations. Many of open-source risk assessment methodologies have similar structure. In practice, methods to assess risks are often composed of the four following steps: threat identification, vulnerability identification, risk determination and control recommendation. These four steps of risk assessment are based on practical experiences in security assessment [3].

This section gave an idea of how risk assessment methodologies are structured. Risk assessment methodologies do not help security experts find potential threats in analyzed

systems, but rather provide a guideline, how to process and evaluate them. Finding threats in the system is left for security experts. Due to the similar nature of many risk assessment methodologies and to the limited content capacity of this thesis no further risk assessment methodologies will be examined. Instead, the idea of automating risk analysis process by using modeling approach will be introduced in next sections.

2.2 Modeling approach

Due to increasing complexity of computer systems the potentiality of security issues is growing, making the reliability a more difficult goal to achieve. Therefore it becomes more and more difficult to predict potential security risks in such systems. This leads to increasing costs spent on the security risk assessment, as well as to higher probability of a mistake made by security experts during the analysis process. Therefore, in order to have a chance to decrease the amount of work for security experts and at the same time to introduce a reliable mechanism to the risk analysis, an opportunity of automating steps of risk analysis workflow must be investigated.

The automation of security risk analysis process can be achieved by means of threat modeling. Threat modeling is an engineering technique that can be used to help identify threats, attacks, vulnerabilities, and countermeasures that could affect an application or system. A threat model is a machine-readable abstraction of the analyzed system that is processed by a computer program in order to identify different kinds of weaknesses of this system. Often it is considered as a representation of the software or device components in a system, the data flows between them and the trust boundaries in the system. With threat-modeling potential design vulnerabilities can be discovered algorithmically by analyzing the system's security properties and identifying potential threats to the assets in the system.

Unlike testing techniques, such as penetration testing or fuzzing, threat modeling can be performed before a product or service has been implemented; this helps ensure that a product or service is as secure as possible by design. Various aspects can be in the center of the modeling process [11]:

- Assets to be protected
- Attacker's view
- Software architecture of the system

The framework and the way of modeling threats may look differently. Currently, there are multiple methodological approaches published in the scientific literature.

At the present time there is no standard for threat modeling. Hence, various ways exist, how threat modeling can be addressed and implemented. Some of them are

attacker-centric, which means that the process of threat modeling starts with an attacker, evaluating his goals and how he might act within the context of the system. Other threat modeling techniques are software-centric, meaning that the goal of the analysis, which is based on the design of a system, is to step through the model, looking for types of attacks against each element of the model. Additionally, there are also asset-centric threat modeling techniques, which involve building a model based on the assets residing in the system. Putting it together, nowadays there are many different methodological approaches to threat modeling described in the scientific literature.

The mentioned threat modeling techniques are orthogonal, meaning that using them produces different results. It is up to the expert, which threat modeling technique to choose in order to carry out a better risk analysis. Multiple techniques can be complemented with each other and applied to the system within one risk assessment process simultaneously. To give an overview about the available tools for carrying out threat modeling, some of them are going to be introduced in the next section.

2.3 Tools

Threat modeling can be conducted without the usage of any software tools or particular frameworks, but due to its broad extent and growing complexity of analyzed systems, a guided process with specified steps and structured resulting reports may be beneficial for most users [15]. Currently there are some tools for threat modeling available on the market. Threat modeling tools simplify the risk assessment process for the experts, making it less error-prone and easier in cases, when the manual risk analysis is unfeasible or difficult to conduct (e. g. for large systems). To give an idea of the concept of those tools, some of them will be introduced in the next section.

2.3.1 Practical Threat Analysis

Eldan Software Systems Ltd. offers a calculative threat analysis and threat modeling methodology, which is implemented in PTA (Practical Threat Analysis) tool. The main idea of this tool is computing threat risks based on the tabular values that have to be provided for the system properties by the user. PTA is a standalone desktop application. Threat modeling with this tool starts with filling in the information about the analyzed system through the desktop user interface, providing data about system assets, their financial value, possible vulnerabilities and mitigation plans. With PTA the user is able to maintain a growing database of threats, create documentation for security reviews and produce justified recommendations taking into account the importance of various threats and the cost of the implementation of the corresponding countermeasures [12].

After all necessary data is provided PTA automatically recalculates threats and countermeasures priorities and provides decision makers with updated action item lists that reflect the changes in threat realities. The final report about the security status of the system looks as shown in the figure 2.9.

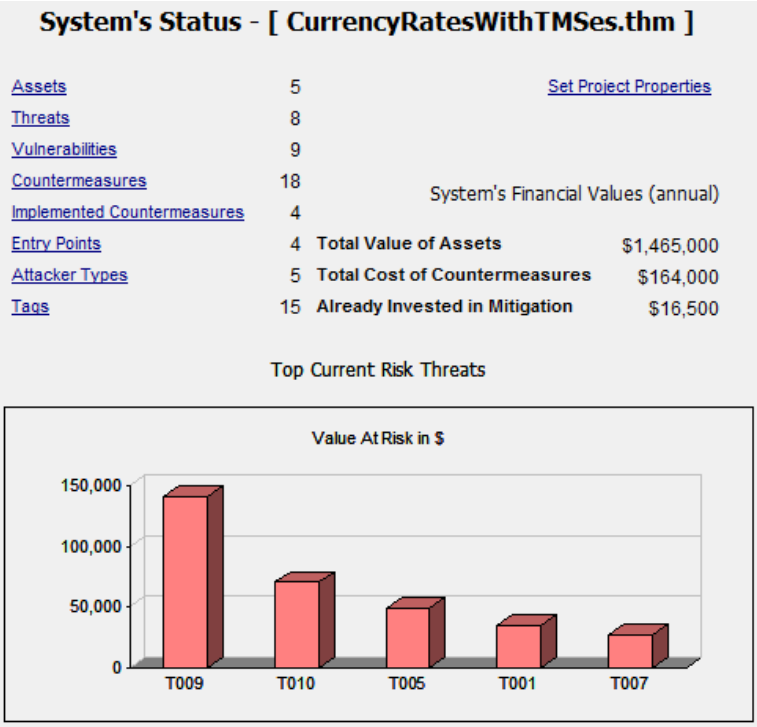

Figure 2.9: PTA user interface

It is worth mentioning that PTA is fully reliant on the user-defined knowledge and is mainly aimed to calculate the value of assets, which are at risk due to existing threats. Both the list of threats and proposed mitigations have to be provided by the user in the tabular format. The only capability of the tool is to calculate the financial impact of threat scenarios provided by the user. The tool does not identify threats automatically, forcing the user to do it completely manually, and therefore it is not powerful for large-scale systems. The tool does not help to reduce the probability of human errors in case of overlooking potential threats.

2.3.2 Trike

Trike is an open source threat modeling methodology and tool. It contains a unified conceptual framework for security auditing from a risk management perspective through the generation of threat models [13]. The tool is represented by several spreadsheets and it uses a risk based approach with distinct implementation, threat and risk models. Threat modeling with this tool starts with filling in the information about the analyzed

Table 2.1: Trike spreadsheet containing system properties

Name	Type	Description	Favored User?	Authenticated?	Attacker?	Uses System?	Used by System?	Shared?	Shared Resources?
Admin	Component Process	The administrator of the system with unrestricted privileges	TRUE	TRUE		TRUE	TRUE	FALSE	TRUE
Authorized user	External Interactor	A user that has an authorized restricted access to the system			TRUE	TRUE	TRUE	FALSE	TRUE
Guest	External Interactor	An external user with minimum privileges			TRUE	TRUE		FALSE	TRUE

application in the predefined spreadsheet framework. The main idea of the concept implemented in Trike is the reliable automatic threat detection based on the tabular representation of a system model. The threat modeling process in Trike is built as follows:

- First, the user has to define all the information about the analyzed system. This includes actors, data model, enumeration of intended actions, use cases, used protocols etc.
- Then, based on this information automatic scripts perform the analysis and produce a list of possible threats, which later has to be analyzed by security experts.

Table 2.1 shows a part of the tabular reflection of the user-defined information in Trike.

In this table the definition of the actors is shown. The fields for storing the information are either text fields (like Name, Type and Description in this example) or dropdown fields (like the rest of the fields that have two predefined possible options – true or false). This information has to be filled by the user. Irrelevant fields are left empty.

After that Trike generates a list potential threats using its hard-coded knowledge base. As Trike is an open source tool, the logic of threat identification may be changed, but it requires an expertise both in security and Trike framework.

An important feature of Trike is combining intended and unintended behaviour of the users, meaning that the user can do malicious actions only as a part of the attack and continue the attack with the normal behaviour of the analyzed system. Threat

information is hard-coded into the knowledge base of Trike and is presented to the security expert right after the model is built. In other words, there is no definition of the knowledge base required.

Trike is heavily reliant on automation. A crucial part of a threat model within the Trike framework is the security requirements that have to be defined in the beginning of threat modeling process. In order to produce a threat model of full value, both technical properties of the system and security objectives have to be described in detail because this information is used in every following step of the analysis.

Comparing Trike to PTA, it is worth mentioning that Trike is more powerful due to its automation. Trike generates possible threats in a separate spreadsheet based on the provided properties of the system and the built-in threat generation logic (which may be enhanced by the user). However, the approach implemented in Trike does not consider the structure of the analyzed system. Hence, it is not applicable for complex hardware/software systems with multiple components, but rather for single software applications.

2.3.3 Microsoft Threat Analysis and Modeling Tool

Microsoft Threat Analysis and Modeling Tool is an instrument for identifying and understanding threats to the business processes, which are inherited by the software application. It is aimed to create and form security strategy, which can be accomplished before implementing the system. Threat modeling process within this tool starts with filling in the information about the analyzed application through the desktop user interface. The tool allows to prioritize business security requirements, giving an opportunity to understand and define a security strategy from a defensive perspective [12].

In order to create a threat model the user has to consolidate already known information about the analyzed system, such as:

- Roles
- Components
- Data

Coupling this information allows the user to define the framework of the analyzed application. Once this is complete, Microsoft Threat Analysis and Modeling Tool automatically analyzes the information and identifies contextualized threats and countermeasures based on a library of known attacks. This library is an extensible and customizable collection of attack patterns that can be contextualized to the application. It also provides step-by-step instructions for mitigating found threats, so that the security team has a concrete specification of countermeasures. Using up-to-date

attack library enables the user to proactively ensure that the system is built securely and remains so.

After threats have been modeled, the tool offers the ability to quantify the level of risks associated with each threat in order to prioritize them and the related countermeasures. The user interface of the latest version (2.1) of Microsoft Threat Analysis and Modeling Tool looks as shown in the figure 2.10.

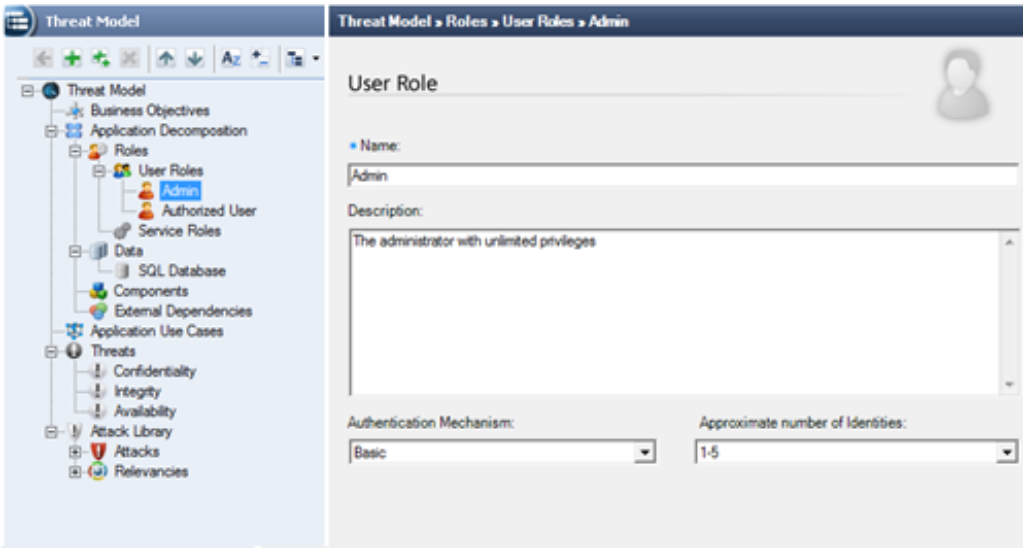

Figure 2.10: MS Threat Analysis and Modeling Tool UI

On the left-hand side of the user interface there is a tree of application properties that have to be filled by the user. There is also a list of threats and attack scenarios that is produced automatically by the tool, but at the same time it can be extended with the user-defined entities.

Microsoft Threat Analysis and Modeling Tool provides several perspectives for modeling the data: visualizations, analytics, reports. The tool allows the user to create living threat models and to produce feature-rich and actionable output. A comprehensive threat model helps model the security design, so that potential vulnerabilities can be exposed before investing time and resources in an unsecure design.

The main limitation of this tool is that it is mainly designed for single web applications. The approach implemented in Microsoft Threat Analysis and Modeling Tool is similar to one used in Trike, because it requires information about the analyzed application and its knowledge base of threats is extensible. Still, the tool is not designed for finding threats in complex systems. It is impossible to provide the tool with information about complex hardware/software structure of a compound systems like the aircraft cabin core system, and hence the tool does not consider it for finding potential threats. The tool is rather not adaptable for complex hardware/software systems with multiple components and

various interactions between them.

2.3.4 Microsoft Threat Modeling Tool 2016

Microsoft Threat Modeling Tool 2016 is a program that helps in finding threats during the design phase of the project. Threat modeling process within Microsoft Threat Modeling Tool 2016 starts with a data flow diagram. From the diagram, potential threats are identified. Unlike Trike and Microsoft Threat Analysis and Modeling Tool, this tool takes the possibly complex system structure into account and detects threats automatically based on the user-provided knowledge base. The user has to specify the component types, which can be a subject of threat exploitations, and their properties; then, the tool automatically searches all components/interactions in the model that correspond to the description of user-defined threats, and reports the ones that were found.

The tool applies a particular threat modeling approach called STRIDE per interaction. STRIDE is an acronym for the threat types of Spoofing, Tampering, Repudiation, Information disclosure, Denial of service and Elevation of privilege [17]. The idea of STRIDE approach is looking at every interaction (edge) in the data flow diagram and searching possible threats, which were previously defined in the knowledge base, for this interaction. While looking up in the knowledge base, the tool checks if there are any threats that are applicable to the current interaction based on its properties.

After all potential threats are listed, for each of them mitigations can be proposed. In some cases the mitigation may require some changes in the system design, making the user to modify the model and to repeat the whole analysis process again.

When the mitigations have been implemented, the product or service is validated against the threat model to ensure that the mitigations work and that design functionality and performance are sufficient. If the design has serious security issues, revisiting the design and the threat model may be appropriate. Summarizing previous tool description, the capabilities of the Microsoft Threat Modeling Tool 2016 are the following:

- Creating data flow diagrams (DFDs) for products or services
- Analyzing data flow diagrams to automatically generate a set of potential threats
- Suggesting potential mitigations to design vulnerabilities
- Producing reports on the identified and mitigated threats

The following example is aimed to deepen the understanding of how the Microsoft Threat Modeling Tool works. A simple system with three components is going to be modeled. These components are: browser client, web server and SQL database. The

model of this system built within Microsoft Threat Modeling Tool framework looks as shown in figure 2.11.

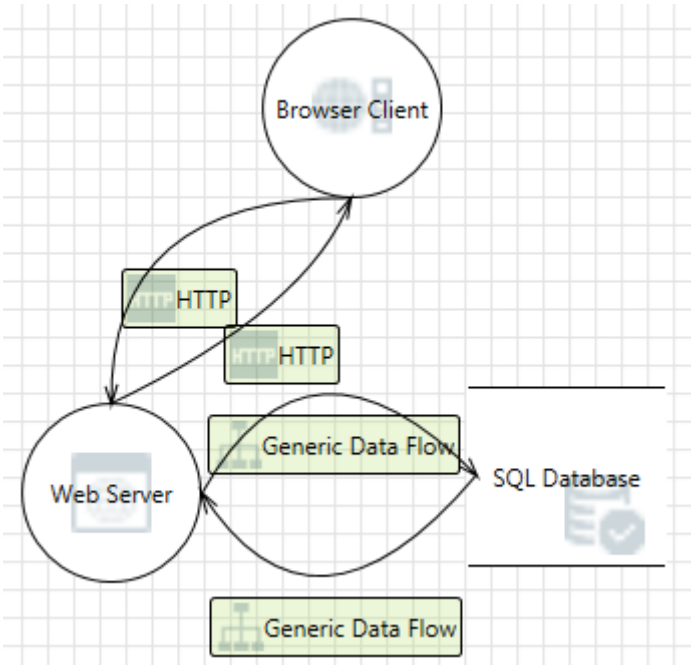

Figure 2.11: Graph-based model

The client interacts with the web server over HTTP, whereas the web server communicates with the database over a general data flow. Both interactions are bidirectional.When the model is built, the analysis view must be switched on. The analysis view provides the information about security threats that can be potentially exploited on the communication links of the system. These threats are expressed in the tabular format, see table 2.2.

Table 2.2: Threats identified by MS Threat Modeling Tool 2016

ID	State	Title	Category	Description	Interaction	Priority
0	Not Started	Spoofing of Destination Data Store	Spoofing	SQL Database may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of SQL Database. Consider using a standard authentication mechanism to identify the destination data store.	Generic Data Flow	High

1	Not Started	Weak Access Control for a Resource	Information Disclosure	Improper data protection of SQL Database can allow an attacker to read information not intended for disclosure. Review authorization settings.	Generic Data Flow	High
2	Not Started	Cross Site Scripting	Tampering	The web server 'Web Server' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.	HTTP	High
3	Not Started	Elevation Using Impersonation	Elevation of Privilege	Browser Client may be able to impersonate the context of Web Server in order to gain additional privilege.	HTTP	High

Then, the existing threat list should be examined and the mitigations have to be proposed. If mitigation measures require changes in the system model, then the model should be modified and the model has to be revised regarding threats one more time.

2.4 Threat modeling approaches

Based on the present risk assessment methodologies and existing implemented tools for threat modeling the following fact becomes clear: risk assessment methodologies are more advanced and adjustable for particular arbitrarily complex systems than existing threat modeling tools. However, these methodologies lack automation.

Existing threat modeling tools may be applied mainly to analyzing software applications rather than to risk assessment of compound systems with many different components, such as aircraft cabin mockup. The only instrument, which is closest to automatic analysis of complex systems, is the Microsoft Threat Modeling Tool 2016. Using only Microsoft Threat Modeling Tool is not sufficient for modeling threats in complex systems because the tool is only capable to detect threats, where two system components and one communication link between them are involved. Threats of this type will be further referred as single-link threats. Unfortunately, MS Threat Modeling Tool implements only the single-link threat modeling approach, which is indeed important but not sufficient for analyzing large composite systems. In complex systems also threats involving

multiple components are possible. These threats will be further referred as end-to-end threats or threat scenarios, meaning that the attacker has access to some component at the one end of the attack and tries to access the asset that is located on the other end of the attack. As long as the output of the existing risk analysis provided by Airbus Group is a collection of threat scenarios, which are considered as end-to-end attacks, these two terms will be further considered as identical. The access point of the attack may not necessarily be directly connected to the component where the asset resides - an attack may involve using functions of other system components in between, as well as exploiting threats on them. Nevertheless, Microsoft Threat Modeling Tool 2016 was chosen as the basis for the further development of concept of the future tool that is going to automate risk assessment process of complex systems.

The other approach that is yet to be implemented and which is aimed to cover this gap is threat modeling with end-to-end threat scenarios. Both of these approaches are going to be described next.

2.4.1 Risk assessment with single-link threats

The first approach in modeling security threats is to search for potential threats on individual communication links within the system model. Single-link threat modeling starts from the design of the system, and attempts to step through a model of the system, looking for types of attacks against each interaction in the data flow diagram. This method assumes a presence of a system model, which is represented by a data flow diagram.

To enable the automatic execution of this approach, it is also necessary to provide a knowledge base that contains an enumeration of possible threats. As long as threats are identified on the communication links, those threats in the knowledge base are described by the properties of system components and characteristics of the data flow between those components. In other words, knowledge base specifies the properties of the interaction links, where a corresponding threat can occur.

Having a system model represented by a data flow diagram and the knowledge base containing all possible threats, the actual list of potential threats in the particular system is issued. This process is done automatically by a tool that implements this concept. An example of a tool that is capable to run through this workflow is Microsoft Threat Modeling Tool 2016.

Although the approach of analyzing risks by identifying single-link threats in the system may be helpful for the threat modeling process, it is still not sufficient for some cases. An example of a threat modeling approach, where finding single-link threats is not sufficient, is security risk assessment of the cabin mockup at Airbus Group. Particularly, more complicated attack scenarios have to be identified during the analysis than separate

potential threats on the communication links.

2.4.2 Risk assessment with end-to-end threat scenarios

The second threat modeling approach, which can be actually described as automation of a risk assessment methodology, such as MEHARI, is modeling end-to-end threats. With this approach, in order to identify attack scenarios, the target of evaluation is analyzed with respect to its objectives and main functionalities. This includes:

- Phases of use
- List of people having access
- Interfaces (internal or external)
- Data flow
- Components (hardware and software)
- Integrated COTS (commercial off-the-shelf) components

There are two general ways to assess risks in an analysis – qualitatively and quantitatively. The latter requires assigning meaningful numerical values to the factors which are used to calculate the risk. Quantitative risk analysis uses discrete ranges (e.g., high, medium, low) in order to characterize the risk. MEHARI assesses risks by the use of threat scenarios. A threat scenario consists of four items:

- Asset – an asset represents information, a system component or its functionality, which has a significant value to its owner or dependent components. An asset can therefore become the target of an attack. A risk analysis elaborates the security risks an asset is exposed to.
- Consequence – a consequence is the loss of the information security property of an asset if a potential attack is executed successfully. In this scope this could be loss of availability (e.g., denial of service), loss of integrity (e.g., misuse or corruption), and loss of confidentiality (e.g., disclosure).
- Cause – the cause is what leads to the risk, for example malicious use of applications, flooding or execution of malicious code.
- Origin – the origin describes where the attack is launched from, for example an attack via the wireless communication means.

The output of the end-to-end risk analysis is a table that contains all potential threat scenarios. As long as there are no solutions currently available for automatic threat modeling with end-to-end threat scenarios, the key point of this thesis is to develop a concept of the tool that is going to address this goal.

Chapter 3

Related work

This chapter gives an overview of the scientific publications that were studied during the current Master's Thesis. The content of all related papers is briefly described and for each paper a short explanation of how they reflect the topic of the current work is provided.

In their publication *Comparison of Risk Analysis Methods: Mehari, Magerit, NIST800-30 and Microsoft's Security Management Guide* [3] authors review and compare four risk analysis methods: Mehari, Magerit, NIST800-30 and Microsoft's Security Management Guide. All four methods are compared on two main criteria – steps that are used to conduct risk assessment and contents of the methods. Chosen risk assessment methods were examined in detail, and in conclusion it turned out that they mostly follow a similar pattern. All of the methods follow the first three of the general steps of risk analysis: threat identification, vulnerability identification, and risk determination. Mehari, Magerit and Microsoft Security Management Guide do not include control recommendation. Authors also reveal that all described methods provide a detailed guide for risk assessment, but only Mehari, Magerit and Microsoft Security Management Guide provide supplementary documents for helping risk assessment.

Investigation of this paper helped to explain the phenomenon of risk assessment and why organizations need it, give an overview of existing threat analysis methodologies and understand the state of the art in this area. The paper references multiple other documents and publications that describe mentioned methodologies in more detail.

The authors Malik, Javed and Mahmud reviewed in detail multiple threat modeling and analysis approaches in their publication *Threat Modeling in Pervasive Computing Paradigm* [12]. The paper examines in detail the threat modeling and analysis approaches being developed at Microsoft and other methods used for threat modeling. The following tools, approaches and resources are described in this publication: Microsoft Threat Analysis and Modeling v2.1, Practical Threat Analysis (PTA), Microsoft Threat Modeling Tool, Threat Model Framework and Methodology for Personal Networks,

Common Vulnerability Scoring System (CVSS). Based on the reviewed methodologies and tools a new threat modeling approach involving the description of security domains is proposed.

The tools reviewed in this paper gave material for the research of the existing tools that solve threat modeling problems. The topic of the publication is directly related to the subject of this Master's Thesis because it explains the idea of multiple existing practical approaches, and the aim of the current Master's Thesis is to develop a concept of the automated analysis of the end-to-end security threats.

In the paper *Secure Operation, Control, and Maintenance of Future E-Enabled Airplanes* [14] IEEE members Sampigethaya, Poovendran and Bushnell are presenting a comprehensive survey of security of the e-enabled airplane. Security of applications like electronic distribution of loadable software and data, as well as future directions such as wireless health monitoring, networked control, and airborne ad hoc networks is discussed. The authors emphasize that current guidance for airplane airworthiness from aviation regulatory agencies, does not cover emerging security threats to the e-enabled airplane. Therefore, to ensure a safe, secure, reliable and efficient air transportation system with high capacity, security of the e-enabled airplane must be addressed. This paper represents an overview of future trends in the engineering of airborne systems with respect do evolving needs of the passengers and changing nature of security attackers' profile.

As long as a significant part of this paper is dedicated to the security of aircraft systems, it is strongly related to the topic of this Master's Thesis. The authors conclude that the use of wireless and off-the-shelf technologies introduces vulnerabilities that mandate careful security considerations due to the potential airplane safety and airline business concerns. Further, the resulting security needs and mechanisms must be integrated into the well-defined processes related to airplane operation, control, and maintenance.

The publication *Threat modeling approaches and tools for securing architectural designs of an e-banking application* [15] written by Möckel and Abdallah elaborates, illustrates and discusses the threat modeling process and its usefulness to the architectural designs of an e-banking application. This paper also seeks for a critical reflection on different approaches and tools, accounting for the complexity and difficulty of the process. Due to the criticality of operations within an e-banking application threat modeling is considered as a useful instrument to mitigate potential attacks by taking proactive measures against them. This paper presents, compares and contrasts several approaches to threat modeling and illustrates their applications to the identification, analysis and understanding of threats relevant to the design of an e-banking application. Although the subject of the publication is securing e-banking applications, it still contains valuable information about the necessary background knowledge of threat modeling, an overview of its foci, concepts and tools.

In his publication *Enumerating All Simple Paths in a Graph* [16] by Frank Rubin presents an algorithm with $O(N^3)$ time complexity for enumerating all simple paths in a graph, where N stands for the number of vertices. The algorithm described in this paper uses Warshall's ordering technique and requires N^3 matrix operations to produce a matrix that contains all necessary information about simple paths in the analyzed graph. This algorithm should be studied in order to produce possible improvements for the algorithm, which is described in the chapter 5 and is used to find out all possible paths from access points to assets in the system model.

The publication *Threat modeling using Formal Methods: A New Approach to Develop Secure Web Applications* [17] by Hussain, Erwin and Dunne made a proposal for threat modeling based on formal methods. Multiple existing threat modeling approaches are described: STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege), DREAD (Damage potential, Reproducibility, Exploitability, Affected uses and Discoverability) and threat modeling using attack trees. The authors claim that existing threat modeling approaches are based on informal and semi-formal techniques. Due to the lack of analysis and proof facilities at the design level of software applications in informal and semi-formal techniques, all the existing threat modeling approaches are incomplete, inconsistent and vague. In the result authors have built a formal model using VDM++ using four core components: STRIDE Model, DREAD Model, Security Mechanisms and Mapping and Integration. Some code fragments of this model are provided in the publication. However, the complete description is out of scope of the paper.

Authors Li and He in their paper *A Unified Threat Model for Assessing Threat in Web Applications* [4] present a unified threat model for assessing threat in web applications. A threat tree model is extended with more semantic and context information about threat to form the new model which is used to analyze and evaluate threat in the software design stage. Unlike other models, the model proposed in this paper can help analyze and evaluate threat in web applications from attackers' perspective in the software design stage. Authors claim that the developed model and model based assessment approach for web applications presented in this paper have 3 advantages: (1) More semantic and context information are provided in this model which can be used to analyze and evaluate threat precisely; (2) The historical statistics information contained in this model together with the dynamic evaluating metric can be used to generate mitigation schemes; (3) The assessing results and mitigation schemes generated in this model can be used to direct secure coding and testing.

In their paper *Security, Internet Connectivity and Aircraft Data Networks* [2] authors Thanthry, Ali and Pendse review the security mechanisms that can be applied in the aircraft onboard systems to address the concerns of the end users. The scope of this research is to determine the viability and need of a security mechanism. The research is also focused on the performance of different security architectures and determining

their usability in the framework of an aircraft data network. As long as the current Master's Thesis is aimed to improve threat modeling technique for Airbus Group, an overview of aircraft information security issues was helpful to get an overall glance about the state of the art in this area. This paper contains a comparison of several security mechanisms that are suitable for aircraft networks.

Chapter 4

High-level concept

This chapter will show the developed high-level approach to automated end-to-end risk assessment that was developed during the research. The created approach to solving threat modeling problem consists of multiple steps. Until now, the security risk analysis has been done by Airbus Group experts manually. For that, an internally developed proprietary methodology is used. The basis of this methodology is MEHARI, which was briefly described before. However, currently there are no tools for conducting risk analysis with MEHARI or a comparable methodology automatically. The reviewed existing threat modeling solutions are not sufficient to maintain the automated threat analysis for the aircraft cabin core system. The purpose of the approach that is described in this chapter is to bring automation into the risk analysis process. Specifically, this approach is focused on the modeling of end-to-end threat scenarios because identifying those is a cornerstone of risk assessment in complex hardware/software systems. The goal of this chapter is to describe each step of this process in detail and to explain the motivation behind each one.

4.1 Threat modeling workflow

The proprietary methodology employed at Airbus Group is based on detection of end-to-end attack scenarios. Moreover, these attack scenarios are being detected in a graph-based model, which basically represents a network plan of the system. Among all present threat modeling tools the manner of modeling systems in Microsoft Threat Modeling Tool 2016 conforms with modeling technique in the proprietary methodology at most. System model in the existing risk analysis that uses the aforementioned methodology is represented by a dataflow diagram, which is similar to the approach implemented in Microsoft Threat Modeling Tool. Hence, Microsoft Threat Modeling Tool 2016 was chosen as a starting point of developing an end-to-end risk assessment tool. Still, in spite of a convenient graph-based modeling technique, Microsoft Threat Modeling Tool 2016

lacks the capability of modeling end-to-end threat scenarios, which involve multiple system components and communications within one attack.

To address end-to-end threat modeling the new Risk Analysis Tool is going to be developed. Automation in the risk analysis can be achieved by modeling the analyzed system and building the knowledge base, which will be used by the Risk Analysis Tool to identify possible threats in the model. As long as Airbus Group is also carrying out threat potentiality and impact assessment, the capability of defining rules for potentiality and impact assessment should also be accommodated in the Risk Analysis Tool. In addition, the changing nature of technologies and security environment requires the extensibility of tools, which are being developed, so that editing and extending the knowledge base is also possible. In this chapter a high-level concept of the prospective tool will be stated. The high-level concept consists of multiple interdependent steps that together form a workflow of the whole threat modeling process. For each phase of the presented workflow the description and corresponding reasons for introducing this step are provided.

4.2 Security risk analysis workflow

The new risk analysis tool concept is built from a set of programs, which are used in a multistep pipeline, where data produced in each step will be used in the following steps. From the user's point of view the flow could be perceived as a procedure consisting of three steps:

1. Knowledge base definition
2. Modeling
3. Threat identification.

Considering the low-level point of view, the workflow has more steps, which describe more detailed data transformations during the risk assessment. The extensive workflow scheme including both high-level and low-level viewpoints is shown in the figure 4.1.

This workflow represents a data flow diagram, which depicts the transformations of the user-provided information. Ellipses in this diagram stand for data, rectangles - for tools, arrows - for data flow between tools that process it. On each step new information is added either by requesting information or as a result of processing it with tools. The final output is the list of end-to-end threats (threat scenarios) that are presented in the tabular form, like it is done in the existing risk analysis provided by Airbus Group. A three-phase risk assessment workflow contains eight specific steps, which will be now explained in detail.

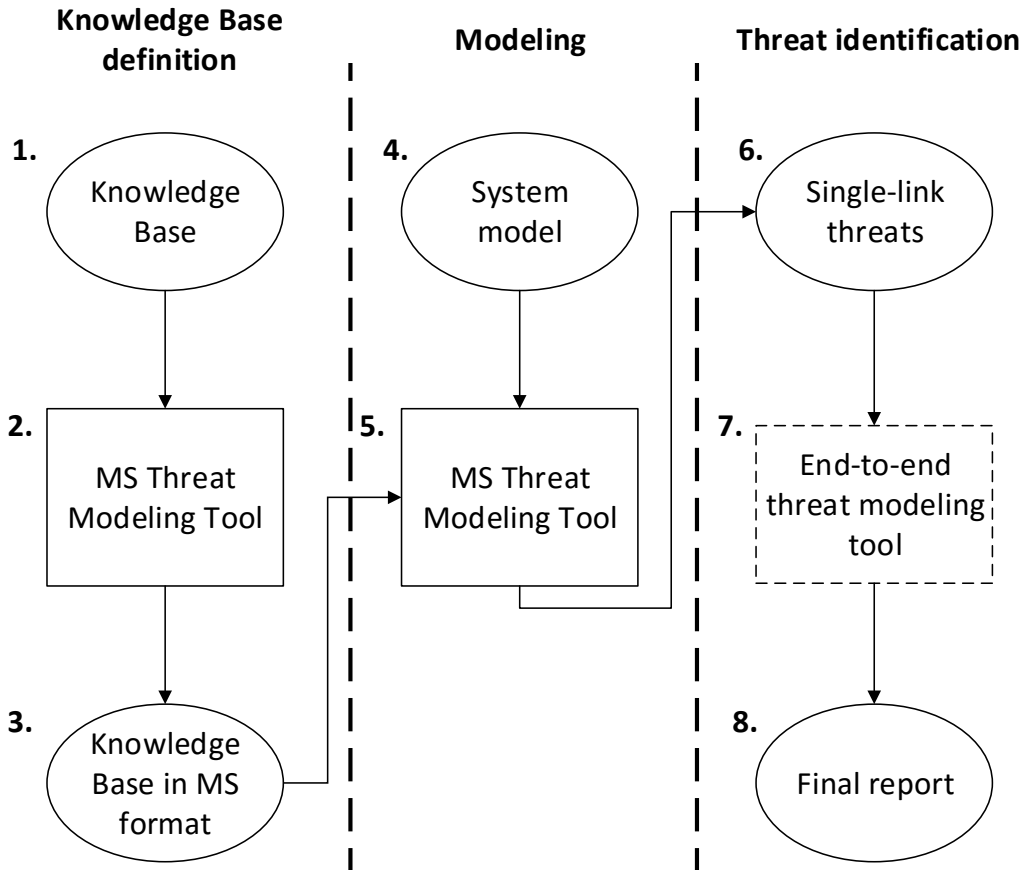

Figure 4.1: End-to-end threat modeling workflow

4.2.1 Knowledge Base definition

The purpose of the first phase called “Knowledge Base definition” is to accumulate all available information about the components and communication links that the analyzed system contains of. Additionally, the knowledge base has to store the information about potential threats that could be exploited on single interactions between two components. All this information has to be provided by security experts, who are responsible for the outcome of the risk assessment. During the first phase “Knowledge Base definition” the information about system components and potential threats has to be saved in the appropriate format that is suitable for Microsoft Threat Modeling Tool 2016.

Collected knowledge base by design is divided into two parts: data stored in Microsoft Threat Modeling Tool knowledge base and data stored in the knowledge base of the new end-to-end threat modeling tool. Microsoft Threat Modeling Tool knowledge base has to contain data about system component types and possible single-link threats because this information is necessary to use the tool for identifying single-link threats.

In order to use identified single-link threats for defining end-to-end threat scenarios the knowledge base of the new end-to-end threat modeling tool has to contain data about threat scenarios. The formal meaning of a threat scenario is the collection of single-link threats along one attack path. The new end-to-end tool provides an opportunity to combine single-link threats, which are detected by Microsoft Tool, into threat scenarios. For that reason a term "threat set" is introduced in the knowledge base of the new tool. A threat set stands for one malicious activity within an attack and contains one or several single-link threat identifiers (which come from the Microsoft Tool) that form a specific malicious activity. Then, the user can combine threat sets in order to define a threat scenario. Within one threat scenario some threat sets can be a precondition for executing others, and therefore new end-to-end tool is designed to provide its users an opportunity to define arbitrary relations between threat sets. All this information has to be stored in the knowledge base of the new end-to-end threat modeling tool. Specific format of both parts of the knowledge base as well as related examples are provided in chapter 5.

As seen in the picture, the first phase of the workflow consists of three elements:

- General knowledge base
- Form-based knowledge representation
- Knowledge base in Microsoft Threat Modeling Tool 2016 format

First of all, the knowledge base should be collected. This includes definition of analyzed system's single elements, potential threats, threat categories and threat conditions. This step is indicated in the first element.

The second element depicts the transfer of the obtained knowledge base to the appropriate format either by using a graphical user interface (see figure 4.2), which is available in the Microsoft Threat Modeling Tool since the version 2016, or by saving this data manually in the XML format.

With the graphical user interface the components and their properties can be defined. All provided data is then stored in the XML format in order to be parsed by the tool during the modeling process.

Third element represents a ready-to-use knowledge base, which is stored in a single XML file. The knowledge base contains information about system components that can be combined to build up a model, data flow link types that aimed to connect the components in a model, single-link threat types that are to be detected by the tool and threat categories in order to group threat types.

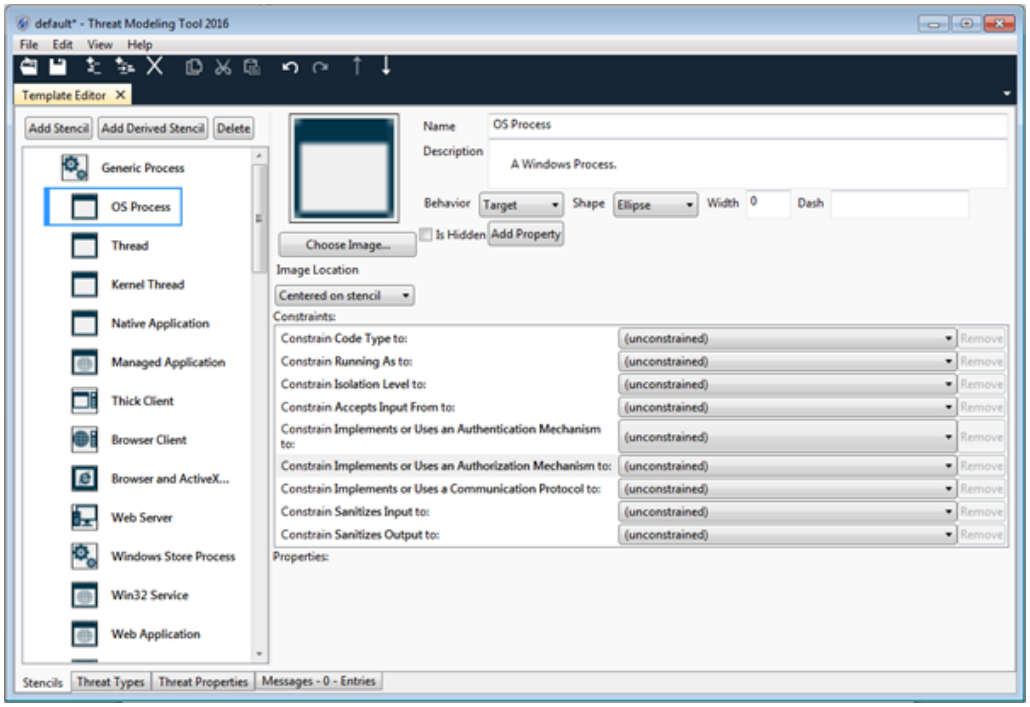

Figure 4.2: Defining the knowledge base using MS Threat Modeling Tool UI

4.2.2 Modeling

During the second phase of the workflow the model itself is constructed and potential single-link threats are detected. The main output of the phase 2 is a list of possible threats that can be exploited on any single communication link in the system. This phase is schematically divided into two steps:

- System model
- MS Threat Modeling Tool

The fourth element in the picture stands for the modeling process. Here the task of the security expert is to create a dataflow diagram from the previously defined components, so that this model matches the structure of the analyzed system most closely. This should be done using the capabilities of the Microsoft Threat Modeling Tool 2016. The graphical user interface for building a model looks as shown in the figure 4.3.

As seen from the picture, there is a menu with all component and data flow types that were taken from the third step of the workflow. In order to build a model the user has to drag and drop necessary components on the canvas and connect those with corresponding data flow edges.

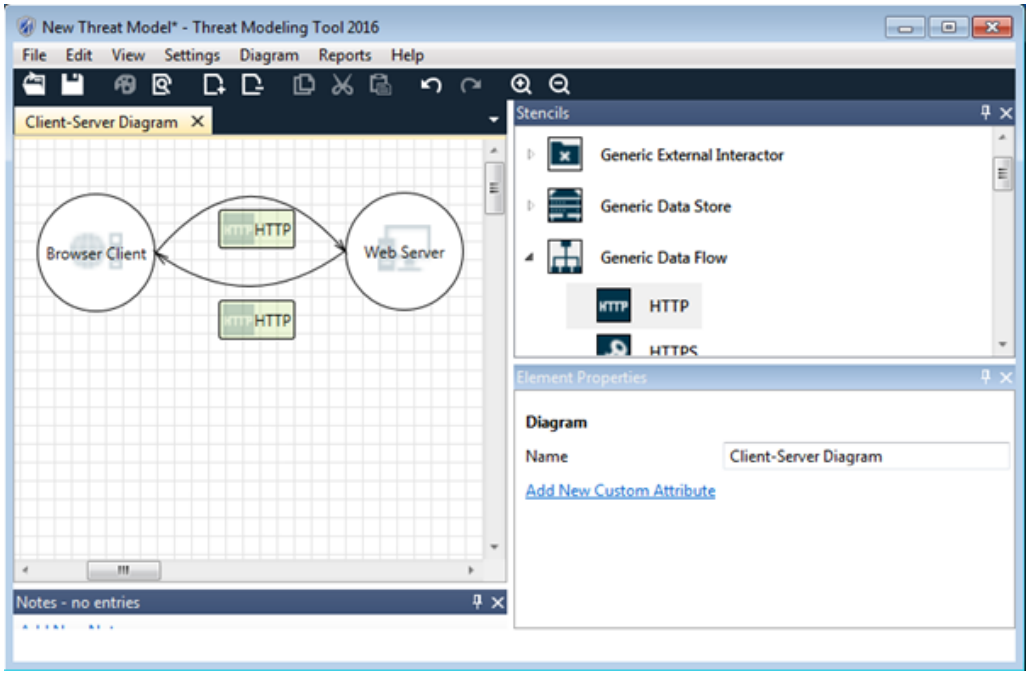

Figure 4.3: Modeling a system with MS Threat Modeling Tool

4.2.3 Threat identification

The main goal of the third phase of the workflow is to identify end-to-end threat scenarios. Before the beginning of the phase "Threat Identification" it is assumed that the analyzed system is already modeled in the Microsoft Threat Modeling Tool 2016. The final phase consists of three items:

- Single-link threat identification
- Processing of single-link threats with additional tool that will be described in the next chapter
- Final report

The item number six in the diagram depicts the list of potential threats that is detected automatically by Microsoft Threat Modeling Tool 2016. This list contains only threats that can be exploited involving individual communication links. Complex end-to-end attacks, which involve exploitation of multiple threats and combining them with the usage of regular system functions, are not yet considered in this list.

The seventh item in the diagram denotes the functioning of the new risk analysis tool. The concept of this tool was developed during the research and it is going to be described in the next chapter. This tool is aimed to cover the deficiency of automatized solutions that are capable to identify end-to-end threats. As it was mentioned in the previous

chapter, there are currently no tools on the market that are designed to detect end-to-end security attack scenarios based on the data flow diagram.

The final item in the diagram, which is marked with number 8, depicts the final report. By design his report has to be generated in the form of a table, where each row stands for a threat scenario. Table 4.1 gives an idea of how this report is currently done at Airbus Group [18].

Table 4.1: A fragment of the existing end-to-end threat analysis

ID	Threat Condition	Cause	Origin	Scenario Description	Elapsed Time	Expertise	Knowledge of Target	Equipment	Feeling of Impunity	Window of Opportunity	Intrinsic Potentiality	Potentiality Rationale
DEVS.C1	Disclosure of Cabin IP Device (same domain)	Eavesdropping	Attack from corrupted Cabin IP Device	An attacker could have corrupted a Cabin IP Device as a starting point for his attack. With this malicious device within the aircraft the attacker is able to eavesdrop traffic and gather information about services of other Cabin IP Devices in the same domain. At a later point in time the attacker would extract the retrieved information.	3	3	3	3	4	3	(3) Likely	To successfully execute the attack an attacker needs to have restrictive knowledge and a short preparation time. The attacker's expertise needs to be proficient and he needs specialized equipment (e.g. specialized connectors). Since some parts of the cabin are not supervised (e.g. lavatories) an attacker has long access during a flight to place the device. In order to collect long-term information the attacker needs to enter that very same aircraft. An attacker has a very low chance of being identified, because the attack is completely passive.
DEVS.C2	Disclosure of Cabin IP Device (same domain)	Malicious use of administration functions	Attack from corrupted Cabin IP Device	An attacker could have corrupted a Cabin IP Device as a starting point for his attack. With this malicious device within the aircraft the attacker is able to gain read access on another device. This allows him to gather information on services and processes on the victim Cabin IP Device with automatized script. At a later point in time the attacker would extract the retrieved information.	2	3	2	3	3	3	(2) Unlikely	To successfully execute the attack an attacker needs to have sensitive knowledge of the system (e.g. administration password) and a moderate preparation time. The attacker's expertise needs to be proficient and he needs specialized equipment (e.g. specialized connectors). Since some parts of the cabin are not supervised (e.g. lavatories) an attacker has long access during a flight to place the device. In order to collect long-term information the attacker needs to enter that very same aircraft in another flight. An attacker has a low chance of being identified, because no data or functions are tampered during the attack.

DEVS.C3	Disclosure of Cabin IP Device (same domain)	Eaves-dropping	Attack from cor-rupted AP	An attacker could have corrupted a Cabin AP as a starting point for his attack. With this corrupted AP in place, the attacker is able to eavesdrop traffic and gather information about services of other Cabin IP Devices in the same domain. If the attacker is on the aircraft he is able to eavesdrop traffic in real-time.	3	3	3	3	4	2	(3) Likely	The potentiality rationale is similar to DEVS.C1 except that the window of opportunity is much smaller, because the replaced WiFi APs are not as accessible as IP Devices in the lavatory. This assumes that WiFi APs are rather hard to be replaced (e.g. behind lining). An attacker has a very low chance of being identified, because the attack is passive.
DEVS.C3	Disclosure of Cabin IP Device (same domain)	Eaves-dropping	Attack from cor-rupted AP	An attacker could have corrupted a Cabin AP as a starting point for his attack. With this corrupted AP in place, the attacker is able to eavesdrop traffic and gather information about services of other Cabin IP Devices in the same domain. If the attacker is on the aircraft he is able to eavesdrop traffic in real-time.	3	3	3	3	4	2	(3) Likely	The potentiality rationale is similar to DEVS.C1 except that the window of opportunity is much smaller, because the replaced WiFi APs are not as accessible as IP Devices in the lavatory. This assumes that WiFi APs are rather hard to be replaced (e.g. behind lining). An attacker has a very low chance of being identified, because the attack is passive.

This table represents a list of end-to-end threat scenarios, which may involve both exploitation of potential threats and the utilization of normal system functioning. It contains overall 13 columns:

- ID – a unique identifier of the threat scenario
- Threat Condition – short description of the type of the current threat scenario
- Cause – starting point of the current attack
- Origin – a precondition for executing current threat scenario
- Scenario description – a detailed specification of the current scenario, which may include the description of other threat exploitations and normal function usage
- Elapsed Time, Expertise, Knowledge of Target, Equipment, Feeling of Impunity, Window of Opportunity and Intrinsic Potentiality – numerical properties of current threat scenario that have to be filled by the security expert manually
- Potentiality Rationale – explanation of the choice of numerical values in previous 7 fields

The generation of this report is the main task of the new tool that is marked number 7 in the diagram. Of course, the output of the future end-to-end threat modeling is not going to exactly match the report provided in the table because the analysis made at Airbus Group involves a lot of expert knowledge and individual approach to each threat scenario. Still, during the research this report was analyzed and the concept of automation of this approach was developed. A detailed description of this concept and is provided in the next chapter. It includes input description, logic of the tool and output description.

Before creating a concept of the tool that is going to automatize the generation of the end-to-end report two important problems remained open. In order to produce a reliable solution for automated end-to-end threat modeling, following questions had to be answered first:

1. Does the existing end-to-end report cover all potential system component/function exploitations?
2. Are there any repeating patterns in the existing end-to-end threat scenarios that could be applied in the new end-to-end threat modeling tool?

To solve these two problems, it was decided to make an independent analysis of potential exploitations of unsecure cabin mockup components/functions in order to find the uncovered threats in the present report (if any) and detect reoccurring rules (if any) that could be perhaps used in the new end-to-end threat modeling tool. This step was named threat mapping.

4.3 Threat mapping

This section describes the process of threat mapping and how it helped to develop a concept of the new tool that is aimed to perform end-to-end threat scenario generation. The main idea was to find out, which single-link threats are involved into threat scenarios in the end-to-end report. The goal of this analysis was to give an overall view of what kinds of threats are engaged into threat scenarios and to find drawbacks in the present end-to-end report, as well as to help create the approach of automatic scenario generation that is going to mitigate existent disadvantages in the present end-to-end threat scenario report. In other words, the purpose of threat mapping is to help create a concept of the tool that will produce threat scenarios automatically. The execution of threat mapping consists of several simple steps: collect a list of possible single-link threats in the system, extract single-link threats from the present risk analysis, map threats from both lists with each other to see, which drawbacks the present risk analysis has.

First of all, it is worth mentioning that the existing end-to-end threat analysis is a description of attack scenarios that may involve both exploitation of unsecure system components or functions and the normal usage of system operations. During the research one or multiple threat exploitations were detected in each threat scenario. The cabin mockup model used in the existing risk analysis represents a data flow diagram; likewise it can be modeled with Microsoft Threat Modeling Tool. It was also observed that threats in the manually produced threat analysis correspond to single-link threats, which Microsoft Threat Modeling Tool 2016 is able to detect automatically, if the appropriate knowledge base is provided. This similarity arises from the nature of threat exploitations described in threat scenarios. Specifically, each threat represents a compromise of either a single system component (e. g. passenger service unit, cabin management server etc.) or a single communication link (such as wireless link between digital cabin logbook access point and line switch). Hence, it was inferred that these compromises comply with single-link threats in the Microsoft Threat Modeling Tool notation. To see the whole list of single-link threats that occur in the risk analysis of the cabin mockup, each threat scenario was inspected and all involved threats were extracted into a standalone list. An example of single-link threat extraction from end-to-end threat scenarios is provided in the table 4.2.

Table 4.2: Single-link threat extraction from end-to-end threat scenarios

ID	Threat Condition	Cause	Origin	Scenario Description	Potentiality Rationale
DEVH.C1	Disclosure of Cabin IP Device (higher domain)	Eaves-dropping	Attack from corrupted Cabin IP Device	An attacker could have corrupted a Cabin IP Device as a starting point for his attack. With this malicious device within the aircraft the attacker is able to eavesdrop traffic and gather information about services of other Cabin IP Devices in a higher domain. At a later point in time the attacker would extract the retrieved information.	To successfully execute the attack an attacker needs to have sensitive knowledge and a long preparation time. The attacker's expertise needs to be on expert level (for breaking the domain segregation function or obtaining the HMAC keys) and he needs specialized equipment (e.g. specialized connectors). Since some parts of the cabin are not supervised (e.g. lavatories) an attacker has long access during a flight to place the device. An attacker has a very low chance of being identified, because the attack is completely passive.
DEVH.C2	Disclosure of Cabin IP Device (higher domain)	Malicious use of ad-ministration functions	Attack from corrupted Cabin IP Device	An attacker could have corrupted a Cabin IP Device as a starting point for his attack. With this malicious device within the aircraft the attacker is able to gain read access on another device. This allows him to gather information on services and processes on the victim Cabin IP Device with automatized script. At a later point in time the attacker would extract the retrieved information.	To successfully execute the attack an attacker needs to have sensitive knowledge of the system (e.g. ad-ministration password) and a long preparation time. The attacker's expertise needs to be on expert level (for breaking the domain segregation or obtaining the HMAC keys) and he needs specialized equipment (e.g. specialized connectors). Since some parts of the cabin are not supervised (e.g. lavatories) an at-tacker has long access during a flight to place the device. In order to collect long-term information the attacker needs to enter that very same aircraft in another flight. An attacker has a low chance of being identified, because no data or functions are tampered during the attack.
DEVH.I1	Corruption of Cabin IP Device (higher domain)	Injection of malicious traffic	Attack from corrupted Cabin IP Device	An attacker could have corrupted a Cabin IP Device as a starting point for his attack. With this malicious device within the aircraft the attacker is able to inject traffic and manipulate other Cabin IP Devices in a higher domain. Once the attack is started (the device is placed) there is no way in adjusting the attack.	The attack needs a long preparation and sensitive knowledge about the target in order to prepare at-tack scripts. An attacker has a low chance of being identified since even though data or functions are tampered, no real-time interaction is required.

This table contains specifications of three end-to-end threat scenarios. Single-link threats that are exploited during the execution of a threat are marked green within the description or potentiality rationale of the corresponding scenario. In this way all threat scenarios were examined and likewise all relevant single-link threats were extracted from each of them into a standalone list.

In order to identify the drawbacks of the existing end-to-end threat analysis, an independent threat analysis of the aircraft cabin mockup was made. Therefore, it was decided to collect a set of potential threats that are feasible in the aircraft cabin mockup from freely available sources and to compare this list with a standalone list of threats extracted from the actual cabin mockup analysis. Comparison of the existing end-to-end report with the report from Microsoft Threat Modeling Tool containing single-link threats was considered as a suitable step to create a concept of end-to-end risk analysis automation. The independent single-link threat analysis of the aircraft cabin mockup was aimed to identify a set of threats that are not present in the existing end-to-end report. For the independent analysis mainly two sources of possible threats were used:

- CAPEC – Common Attack Pattern Enumeration and Classification [19]
- List of wireless network attack patterns [20]

The first source is a publicly available catalog of common attack patterns, mainly consisting of general purpose security threats and web oriented threats. The CAPEC enumeration does not cover the set of attacks on wireless communication. Hence, the second source was used to discover possible wireless threats. Both lists were carefully checked through with the assistance of Airbus Group experts and threats, which were considered irrelevant for the cabin core system, were removed from the list. Only those threats remained, which were considered feasible in the aircraft cabin mockup and saved in the knowledge base of Microsoft Threat Modeling Tool for further detection of these threats in the model. The remaining threats were assigned to corresponding components of the cabin mockup model. All these threats were saved into the knowledge base of Microsoft Threat Modeling Tool and a model of the aircraft cabin mockup was built in order to receive an automatically generated list of threats. Automatically detected threats were mapped to those from the standalone list of threats that were previously extracted from the present cabin mockup risk analysis. An extract from the threat mapping table looks as shown in the table 4.3.

Table 4.3: Threat mapping table

Cause	Scenario Description	Potentiality Rationale	Listing of involved threats
Flooding	An attacker could flood the LS interfaces. Depending on the scheduling policy of the switches enforcing domain segregation, a switch could fall into pass all traffic policy or in drop all traffic policy when it is overloaded.	An attacker needs a very short time of preparation to understand the system and prepare attacks. Layman skills and public knowledge of the target is sufficient for the attack. Specialized equipment is needed (e.g. connectors). An attacker has a low chance of being identified since even though flooding is a conspicuous attack, no real-time interaction is required.	TCP Flood, ICMP Flood, UDP Flood, CAM table overflow attack, DHCP starvation attack
Eavesdropping	An attacker could have corrupted a Cabin AP before being imported into the A/C. With this malicious device within the aircraft the attacker is able to eavesdrop traffic and gather information about services of other Cabin IP Devices in a higher domain. If the attacker is on the aircraft he is able to eavesdrop traffic in real-time.	To successfully execute the attack an attacker needs to have sensitive knowledge and a long preparation time. The attacker's expertise needs to be on expert level (for breaking the domain segregation function or obtaining the HMAC keys) and he needs specialized equipment (e.g. specialized connectors). The window of opportunity is small, because the replaced WiFi APs are not accessible in the lavatory. This assumes that WiFi APs are rather hard to be replaced (e.g. behind lining). An attacker has a very low chance of being identified, because the attack is completely passive.	Malware-Directed Internal Reconnaissance, Targeted Malware, Malicious Software Update, Malicious Logic Insertion via Counterfeit Hardware, Malicious Logic Insertion into Product Memory, Intent Intercept, Intent Spoof, Leverage Executable Code in Non-Executable Files, ASIC With Malicious Functionality, Rogue access points, MAC spoofing, Evil twin AP, 802.1X Identity Theft, 802.1X Password Guessing, 802.1X EAP Downgrade, Eavesdropping (Wireless)

Injection of malicios traffic	An attacker could have corrupted a Cabin AP before being imported into the A/C.With this malicious device within the aircraft the attacker is able to inject traffic and manipulate services of other Cabin IP Devices in a higher domain. If the attacker is on the aircraft he is able to inject traffic in real-time.	The attack needs a long preparation and sensitive knowledge about the target in order to prepare attack scripts. The window of opportunity is small, because the replaced WiFi APs are not accessible in the lavatory. This assumes that WiFi APs are rather hard to be replaced (e.g. behind lining). An attacker has a moderate chance of being identified since data or functions are tampered.	Malware-Directed Internal Reconnaissance, Targeted Malware, Malicious Software Update, Malicious Logic Insertion via Counterfeit Hardware, Malicious Logic Insertion into Product Memory, Intent Intercept, Intent Spoof, Leverage Executable Code in Non-Executable Files, ASIC With Malicious Functionality, Rogue access points, MAC spoofing, Evil twin AP, 802.1X Identity Theft, 802.1X Password Guessing, 802.1X EAP Downgrade, Format String Injection, Reflection Injection, Command Delimiters, Argument Injection, Manipulating Input to File System Calls, File System Function Injection, Content Based, OS Command Injection, Cache Poisoning, Functionality Misuse, Man in the Middle Attack, Manipulating User-Controlled Variables, Manipulating Opaque Client-based Data Tokens, Removing Important Functionality

This table contains the information about threat scenarios that is relevant to threat mapping, namely: cause, scenario description and potentiality rationale. Those sections that were considered as single-link threat exploitations are marked green. The right-most column contains listings of threats that were gathered from freely available sources and were considered as coincident with green marked threats from threat scenarios. That is, single-link threats from the right-most column were mapped to threats extracted from end-to-end threat scenarios. In some cases the right-most list of single-link threats did not cover all aspects of malicious activities described in a threat scenario due to the generality of threats from the present risk analysis. Therefore, it cannot be claimed that all threats from the present risk analysis are completely covered by single-link threats from the independent analysis, but all threats that were considered conforming to threat scenario descriptions have been put to the right-most column. After the mapping had been done there was a list of single-link threats from the open sources left unmapped. These unmapped threats identified the shortcoming of the existing end-to-end threat analysis. In order to create a more reliable risk analysis, these remaining threats have to be included in the new risk analysis report.

Furthermore, it can be noticed that threat listings in rows 2 and 3 contain a set of identic single-link threats. This situation occurred with multiple threat scenarios due to repeating threat specifications. An example of such a repeating threat references is “an attacker could have corrupted cabin wireless access point” from scenario descriptions in rows 2 and 3. This threat description is rather abstract and therefore it was mapped to multiple single-link threats from the open sources, meaning that every threat scenario that takes a corrupted wireless access point into consideration will have the same single-link threats in the right-most column of the mapping table. Thus, during the threat mapping repeating patterns in threat scenarios were observed. Therefore, it is assumed that there is a possibility of applying similar templates in order to produce different end-to-end threat scenarios in different parts of the system model. The threat overlap that took place in the threat mapping caused an idea of combining single-link threats into threat scenarios and identifying them automatically.

At this point, having Microsoft Threat Modeling Tool that can be used to model the system and to identify single-link threats, the only lacking capability is to combine identified threats into end-to-end threat scenarios. As long as currently no automated solution for end-to-end risk assessment exists, the actual end-to-end risk assessment made by Airbus Group experts was considered as a potential output of such a tool. Since automated creation of such an end-to-end threat report is the global goal of the whole research, the detailed low-level concept of this tool will be provided in a separate chapter.

Chapter 5

Low-level concept

The main task of the new tool that is being designed here is to find end-to-end threats in the system model. The tool is designed to identify end-to-end threats based on the following available information:

- System model – a network plan of the analyzed system.
- Asset list – valuable information, system components (HW or SW) or their functionality that have a significant value to the owner or dependent components.
- Access point list - system components that are available for the attacker by design.
- Existing single-link threats in the model – threats identified by Microsoft Threat Modeling Tool.
- Threat set relations – in order to execute a threat scenario, exploitation of some threats may be a precondition for others. The structure of threat sets being a precondition for other threat sets may be arbitrarily complex.

Now each item from the list above is going to be explained in detail. In this document all steps of the threat modeling concept will be presented based on a single small example that is aimed to help understand the workflow.

5.1 System model

An important component of the whole model-based threat assessment is the system model. It is going to be built with Microsoft Threat Modeling Tool. Further risk analysis is going to be built upon this model. The model is represented by a network plan that is depicted by a connected directed graph, where a bidirectional communication link is modeled with two edges with opposite direction. The nodes in this graph stand for system components, edges denote communication links between components and

boundaries indicate domain areas. System model should be built using Microsoft Threat Modeling Tool, therefore, component types and single-link threat types should be defined in its knowledge base.

Before building a model, all possible components have to be first defined in the knowledge base by a security expert. As long as the knowledge base will be interpreted by the Microsoft Threat Modeling Tool, it must stick to a certain format. An XML listing 5.1 fragment shows an example for component specification.

Listing 5.1: Definition of component types

```
<ArrayOfElementType>
 <ElementType>
  <Name>Passenger Service Unit</Name>
  <ID>PSU</ID>
  <Description>A representation of a PSU</Description>
  <Hidden>false</Hidden>
  <ParentElement>GE.P</ParentElement>
  <Representation>Inherited</Representation>
  <Image>/Images/ImageDevice7.png</Image>
  <Attributes/>
 </ElementType>
 <ElementType>
  <Name>Line Switch</Name>
  <ID>LS</ID>
  <Description>A representation of a LS</Description>
  <Hidden>false</Hidden>
  <ParentElement>GE.P</ParentElement>
  <Representation>Inherited</Representation>
  <Image>/Images/ImageSwitch7.png</Image>
  <Attributes/>
 </ElementType>
 <ElementType>
  <Name>Main Cabin Switch</Name>
  <ID>MCS</ID>
  <Description>A representation of a MCS</Description>
  <Hidden>false</Hidden>
  <ParentElement>GE.P</ParentElement>
  <Representation>Inherited</Representation>
  <Image>/Images/ImageSwitch7.png</Image>
  <Attributes/>
 </ElementType>
 <ElementType>
  <Name>SomeIP</Name>
  <ID>SomeIP</ID>
  <Description>
   A representation of SomeIP communication flow
  </Description>
```

```
    <Hidden>false</Hidden>
    <ParentElement>GE.DF</ParentElement>
    <Representation>Inherited</Representation>
    <Image>/Images/ImageDataFlow7.png</Image>
    <Attributes/>
  </ElementType>
</ArrayOfElementType>
```

There are four system components defined in the above knowledge base fragment: Passenger Service Unit, Line Switch, Main Cabin Switch and SomeIP communication flow. All components are stored within a parent XML tag “<ArrayOfElementType>”. Each component, that is stored within a “<ElementType>” tag, has a name, which is then shown in the model, and a unique identifier, which is then used by the tool to distinguish components from each other. The tag "<ParentElement>" indicates the identifier of the group of elements that a corresponding component belongs to. Other fields (mostly self-explanatory) are not important at the moment, but necessary to produce a valid knowledge base.

After system components are defined, next step is to build a model. A model can consist of arbitrary number of components (of the same type as well), whose types were defined previously (see listing 5.1). Microsoft Threat Modeling Tool will generate unique identifier for all components of the model, even if they are of the same type. On the picture below a model of a small prototype system with previously defined components is shown.

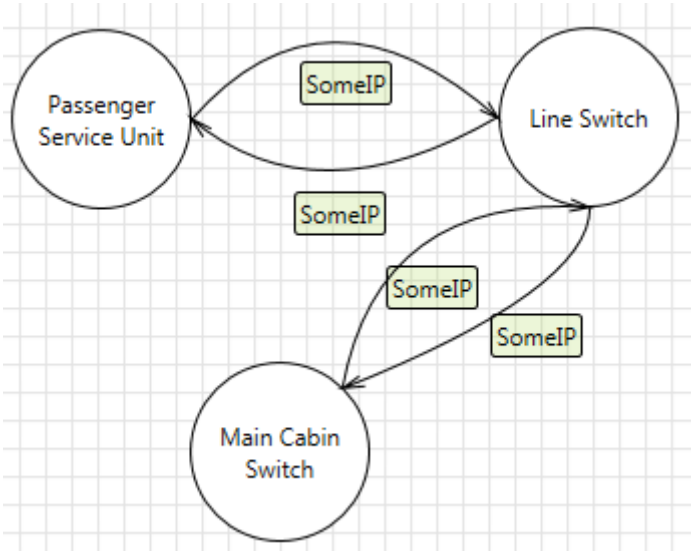

Figure 5.1: Prototype system model

This is a complete model of the analyzed system. There are three components in this model: Passenger Service Unit (PSU), Line Switch (LS) and Main Cabin Switch (MCS).

These components are connected with each other by wired bidirectional links using SomeIP protocol. These links are depicted by directed edges in the graph.

5.2 Asset list

The goal of an attacker is to compromise one or multiple assets in the system by executing a threat scenario. An asset can be anything that has a value to the company, for example, knowledge, software, services, computers etc. Therefore, after creating a model the user has to specify the asset list. These could be either function or data assets. Each threat scenario stands for a compromise of an asset. Assets can reside on nodes (e. g. configuration files on a switch), edges (e. g. transmission of important data through this link) or boundary crosses (e. g. domain segregation). Hence, after the model has been built, user has to specify assets and their locations within the model. The knowledge base section that is responsible for holding information about assets is also stored in XML format. In the XML listing 5.2 example two assets are defined.

Listing 5.2: Definition of assets

```
<AssetList>
 <Asset>
  <Id>LSF1</Id>
  <Name>Line Switch Function</Name>
  <Type>Function</Type>
  <Description>
   Line Switches enforce the domain segregation and perform
   basic firewall activities as a multi-domain device
  </Description>
  <Element>Node</Element>
  <ElementId>807f63d9-168c-435e-8dbd-5257885c7a51</ElementId>
 </Asset>
 <Asset>
  <Id>MCS1</Id>
  <Name>Main Cabin Switch Data</Name>
  <Type>Data</Type>
  <Description>
   All IP traffic passes the main cabin switch. Manipulated
   switching decisions can affect availability and integrity of
   the IP traffic passing the switch.
  </Description>
  <Element>Edge</Element>
  <ElementId>af2ae65d-5aa2-4c00-8e64-97e6a6c81769</ElementId>
 </Asset>
</AssetList>
```

The upper XML fragment denotes the asset definition, which is stored in the knowledge base of the new end-to-end tool. It contains two assets, each of which has a unique identifier, name, type (function or data), description, a property specifying whether this asset resides on a node, edge or a boundary crossing, and the identifier of the element, on which this asset is located within the model.

5.3 Access point list

An essential part of the model is the list of access points. An access point represents a system component or interface that is available to the user. All access points are assumed to be potential starting points of the attack. There are usually multiple groups of people who can access different parts of the analyzed system during its usage or maintenance. Theoretically, a security attack can be conducted by any person from any group of users or maintenance personnel. Defining user groups helps to model threats that come from specific system users - for example, aircraft passengers. Accordingly, prior to building a model it is necessary to specify the components of the model that are accessible to the potential attacker. If there are multiple groups of system users, such as passengers, cabin crew and maintenance personnel for the aircraft cabin, then individual models have to be built for each group of users. The knowledge base of the model must contain information about access points in the XML format. An XML listing 5.3 denotes the definition an access point.

Listing 5.3: Definition of access points

```
<AccessPointList>
 <AccessPoint>
  <Id>PSU1</Id>
  <Name>Passenger Service Unit</Name>
  <Description>Passengers have a direct access to the PSU in
    front of their seats.</Description>
  <Element>Node</Element>
  <ElementId>004f320b-ccc9-4703-9728-37a146946296</ElementId>
 </AccessPoint>
</AccessPointList>
```

In this XML fragment an access point is introduced. Each access point has its own unique identifier, name, description, a property specifying whether this access point resides on a node or an edge and the identifier of the model element, on which this access point is located. The last element identifier is read from the model that was built using Microsoft Threat Modeling Tool, which generates these identifiers automatically.

5.4 Single-link threats

After system assets and model are defined, it is possible to make a single-link threat analysis using Microsoft Threat Modeling Tool. Single-link threats are the basic entities of an attack on a system, because by exploiting them an attacker can come up with a complex end-to-end threat scenario. Single-link threats are identified automatically by means of user-defined knowledge base. This knowledge base consists of several XML files, where single-link threats and their properties are defined. An XML listing 5.4 gives an example of single-link threat definition.

Listing 5.4: Single-link threat definition

```
<ArrayOfThreatType>
 <ThreatType>
  <Id>SNT</Id>
  <ShortTitle>Sniffing Network Traffic</ShortTitle>
  <Category>GI</Category>
  <Description>
   Monitoring network traffic between nodes of a network.
  </Description>
  <GenerationFilters>
   <Include>
    (source is 'MCS' and target is 'LS') or
    (source is 'LS' and target is 'MCS')
   </Include>
   <Exclude/>
  </GenerationFilters>
 </ThreatType>
 <ThreatType>
  <Id>AS</Id>
  <ShortTitle>Action Spoofing</ShortTitle>
  <Category>DI</Category>
  <Description>Disguising one action for another.</Description>
  <GenerationFilters>
   <Include>
    source is 'PSU'
   </Include>
   <Exclude/>
  </GenerationFilters>
 </ThreatType>
 <ThreatType>
  <Id>AI</Id>
  <ShortTitle>Argument Injection</ShortTitle>
  <Category>I</Category>
  <Description>
   Injecting data or command syntax through non-validated and
   non-filtered arguments.
```

```
  </Description>
  <GenerationFilters>
   <Include>
    source is 'LS' and target is 'MCS'
   </Include>
   <Exclude/>
  </GenerationFilters>
 </ThreatType>
 <ThreatType>
  <Id>AMEEF</Id>
  <ShortTitle>Accessing, Modifying or Executing Executable
  Files</ShortTitle>
  <Category>EA</Category>
  <Description>
   Exploiting a system configuration that allows to access
   and an executable file.
  </Description>
  <GenerationFilters>
   <Include>
    source is 'LS'
   </Include>
   <Exclude/>
  </GenerationFilters>
 </ThreatType>
</ArrayOfThreatType>
```

In this XML section the definition of four single-link threats is shown. Each threat definition has following information.

- Unique identifier – distinguishes current threat from the others.
- Short title – will be displayed in the report.
- Category identifier – denotation of the category that current threat belongs to.
- Generation filters – an item containing “<Include>” and “<Exclude>” rules. The expressiveness of these filters will be described.

Tags “<Include>” and “<Exclude>” contain Boolean expressions that specify the properties of a single communication link, which Microsoft Threat Modeling Tool uses to identify potential threats. If “<Include>” rule evaluates to true then “<Exclude>” is evaluated if it is present. If “<Exclude>” evaluates to false then the corresponding threat appears in potential threat list, otherwise the corresponding threat is ignored. With these filters it is possible to examine the type of source node, target node and edge between them. For example, "source is 'MCS'" means that the source node of the communication link must be of the type 'MCS'. The same can be done considering target node and edge: "target is ...", "flow is ...". Also, it is possible to examine the attributes

of source node, target node and edge between them. For example, "source.attribute1 is 'Yes'" means that the value of source node attribute "attribute1" must be equal to 'Yes'. The same can be done considering target node and edge: "target.attribute1 is ...", "flow.attribute1 is ...". One include/exclude rule can consist of arbitrary number of such Boolean expressions linked with AND/OR operators.

This description of the knowledge base format is not complete. There are more ways to adjust the properties of threats. Further information about the MS Threat Modeling Tool knowledge base format can be found in the Threat Modeling Tool SDK.

5.5 Threat set dependencies

When single-link threats are identified, the next step is to find out end-to-end threat scenarios. In order to enable automatic search for end-to-end threats, security expert has to define threat sets. Threat sets stand for combinations of single-link threats, forming a single activity within a threat scenario. Each threat set consists of one or several single-link threats detected by Microsoft Threat Modeling Tool. All in all, a threat scenario is a collection of single-link threats that come from Microsoft Threat Modeling Tool report. The notion of threat sets is introduced for the purpose of reusing same combinations of single-link threats in multiple threat scenarios. Using all this information the tool is going to enumerate all paths in the model between access points and assets, and according to potential single-link threats along each path, detect possible threat scenarios. A threat scenario appears in the end report, if all single-link threats of this scenario are present along the path between access point and asset.

Definition of threat sets is necessary because one end-to-end threat scenario may involve exploitation of multiple single-link threats. Furthermore, exploitation of some single-link threats may be a mandatory precondition for exploitation of other threats. Hence, an expert has to decide, which threats are going to be involved into different activities of threat scenarios, and what are dependencies between them. An XML listing 5.5 gives an example, how to combine multiple single-link threats into sets in order to define activities of threat scenarios.

Listing 5.5: Threat set definition

```
<ThreatSetList>
 <ThreatSet>
  <Name>Corruption of cabin IP device</Name>
  <SingleLinkThreats>
   <ThreatId>AS</ThreatId>
   <ThreatId>AMEEF</ThreatId>
  </SingleLinkThreats>
 </ThreatSet>
 <ThreatSet>
```

```
  <Name>Eavesdropping traffic</Name>
  <SingleLinkThreats>
   <ThreatId>SNT</ThreatId>
  </SingleLinkThreats>
 </ThreatSet>
 <ThreatSet>
  <Name>Traffic injection</Name>
  <SingleLinkThreats>
   <ThreatId>AI</ThreatId>
  </SingleLinkThreats>
 </ThreatSet>
</ThreatSetList>
```

The example above shows the definition of three threat sets, which stand for activities within threat scenarios. Each threat set has the following properties:

- Title – a name of the malicious activity within an end-to-end attack.
- Identifiers of single-link threats (using Microsoft Threat Modeling Tool notation) that an attacker has to exploit in order to conduct current malicious activity. There can be arbitrarily many (but at least one) threats specified in this list. In this example all three threat sets are constructed from the single-link threats defined in the previous example.

Several properties of threat scenarios have to be defined separately. In this part of the knowledge base only the information dedicated exclusively to end-to-end threat scenarios as separate entities will be stored. An example of threat scenario definition is provided in the XML listing 5.6.

Listing 5.6: Threat scenario definition

```
<ThreatScenarioList>
 <ThreatScenario>
  <Name>Eavesdropping and injection of malicious traffic</Name>
  <ID>DEVS.C1</ID>
  <Desc>Attack from a corrupted Cabin IP Device</Desc>
 </ThreatScenario>
</ThreatScenarioList>
\capion{trolo}
```

An XML tag “<ThreatScenarioList>” contains all definitions of threat scenarios. In the current example there is only one threat scenario defined. For each threat scenario three properties have to be provided – name, ID and description.

Then, the expert has to specify dependencies between previously defined threat sets (malicious activities) and thereby define threat scenarios. As long as exploitation of some threat sets may be a necessary precondition for exploitation of other threat sets

within an end-to-end attack, dependencies can be reflected by a directed graph. An example of how a threat scenario can be defined by a directed graph is illustrated in the figure 5.2.

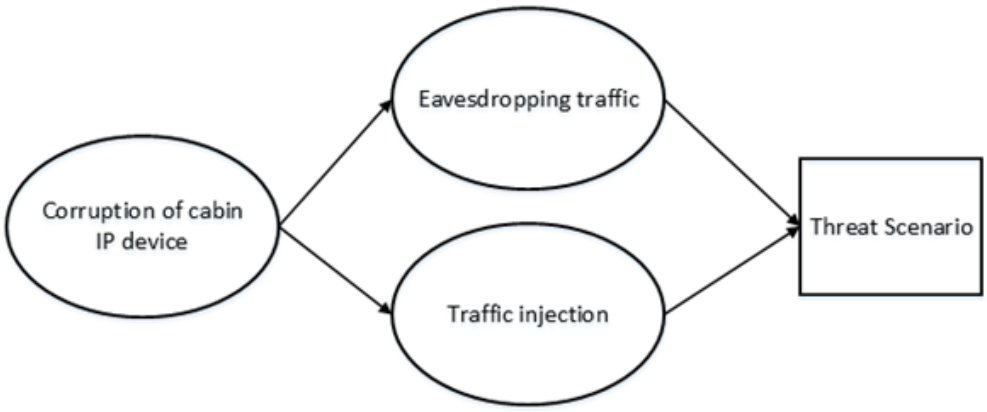

Figure 5.2: Threat set dependencies forming a threat scenario

Threat set dependencies are designed to be defined in the form of a graph. Threat sets are represented by elliptical nodes, threat scenarios by rectangular nodes. Edges in the graph mean that exploitation of threats from the source node is a precondition for exploitation of threats from the destination node. Respectively, the right-most node in this graph, where all arrows terminate, is a threat scenario itself because exploitation of all threat sets is a precondition for conducting this threat scenario. In this particular example the edge from the left-most node to the top-most node, for instance, means that in order to exploit threats from the set “Eavesdropping traffic” an attacker has to first exploit all single-link threats from the set “Corruption of cabin IP device”. The structure of the graph may be arbitrarily complex, as far as it makes logical sense (no cycles).

5.6 Algorithm for generating threat scenarios

At the moment when single-link threats are found and threat sets including their dependencies are defined, the tool has all necessary information to detect end-to-end threat scenarios in the model. The algorithm, which is used by the tool, is going to loop over all access points in the graph-based model, finding all simple paths from those to the assets using a modified Depth-First-Search algorithm for enumerating all simple paths in a graph [21].

In his textbook "The Algorithm Design Manual" Skiena presents the algorithm for enumerating all simple paths between two nodes by using Depth-First-Search algorithm and backtracking. There is no explicit formula that counts the number of solutions as a

function of the number of edges or vertices, because the number of paths depends upon the structure of the graph.

The idea of the provided algorithm can be described as follows. Let G = (V,E) be a finite directed graph without multiple edges or self-loops. V = (1, 2, ..., N) is the list of vertices of G, and E = (e1, e2, ..., em) is the list of edges of G. The backtracking algorithm adds potential edge candidates to the partial solution A, invokes itself recursively and removes them after the recursive function call is finished. The following C++ code snippet expresses the implementation of the backtracking function:

Listing 5.7: Implementation of backtracking

```
void backtrack(int a[], int k, data input){
  int c[MAXCANDIDATES]; /* candidates for next position */
  int ncandidates;      /* next position candidate count */
  int i;                /* counter */

  if (is_a_solution(a,k,input))
    process_solution(a,k,input);
  else {
    k = k+1;
    construct_candidates(a,k,input,c,&ncandidates);
    for (i=0; i<ncandidates; i++) {
      a[k] = c[i];
      make_move(a,k,input);
      backtrack(a,k,input);
      unmake_move(a,k,input);
    }
  }
}
```

In this code snippet a represents an array of edge identifiers that are present in the current path, k stands for the number of edges in the current partial solution (or depth of the recursion), input represents any kind of data structure that holds information about the graph, source and destination nodes. Backtracking ensures correctness by enumerating all possibilities. It ensures efficiency by never visiting a state more than once.

First, the backtracking algorithm checks if current partial solution is valid for the initial problem. If so, it outputs the solution by process_solution function. Otherwise, recursion depth variable k is incremented and possible candidates for the current position in the partial solution are generated by construct_candidates routine. After that, the algorithm loops through the array of generated candidates, adds them to the current solution with make_move routine and makes a recursive call to backtracking function. When the

recursive call is finished, the algorithm removes previously added candidate from the partial solution by using unmake_move routine.

Now it is necessary to explain, what is the partial solution in the case of enumerating all paths between two nodes. The starting point of any path from s to t is always s. Thus, s is the only candidate for the first position and S 1 = s. The possible candidates for the second position are the vertices v such that (s, v) is an edge of the graph, for the ath wanders from vertex to vertex using edges to define the legal steps. In fact, S k+1 consists of the set of vertices adjacent to a k that have not been used elsewhere in the partial solution A. Hence, the solution A can be represented as an array of edge identifiers, where successive array entries define edges that have a common node. The first entry of the array A stands for the edge that comes out of the source node, and the last entry defines the edge, whose destination is the target node.

The edge candidates are represented by edges that are adjacent to the last node of the partial solution. The first position can be occupied by the edges that are adjacent to the source node, second position can be occupied by the edges that are adjacent to the node that is examined on the current backtracking step, and so on. The following code snippet provides the implementation of the candidate generation logic:

Listing 5.8: Implementation of function for generating candidates

```
void construct_candidates(int a[], int k, graph g, int c[],
                                    int *ncandidates) {
  int i;              /*counter*/
  bool in_sol[NMAX]; /*what is already in the solution?*/
  edgenode *p;        /*temporary pointer*/
  int last;           /*last vertex on current path*/
  for (i=1; i<NMAX; i++) in_sol[i] = false;
  for (i=1; i<k; i++) in_sol[a[i]] = true;
  if (k==1) {         /* always start from vertex 1 */
    c[0] = 1;
    *ncandidates = 1;
  } else {
    *ncandidates = 0;
    last = a[k-1];
    p = g.edges[last];
    while (p != NULL) {
      if (!in_sol[p->y]) {
        c[*ncandidates] = p->y;
        *ncandidates = *ncandidates + 1;
      }
      p = p->next;
    }
```

```
    }
}
```

In order to generate candidate edges for the current position, the following input has to be provided: an array of integers a, defining the current partial solution A, an integer k that stands for the recursion depth in backtracking, an instance of the data structure that contains information about the graph, an integer array c that is going to store candidates and an integer candidates, indicating the number of generated edge candidates during the current function call. The candidate edges that are generated during one call of construct_candidates function are the ones that are adjacent to the last node of the current partial solution. In the next step of backtracking each of them is added to the partial solution, a recursive call to backtracking function is made, and when it is finished, the candidate edge is removed from the partial solution. This logic ensures the depth-first manner of the graph traversal while constructing partial solutions.

In order to demonstrate, how the algorithm works, consider an example graph that is depicted in figure 5.3.

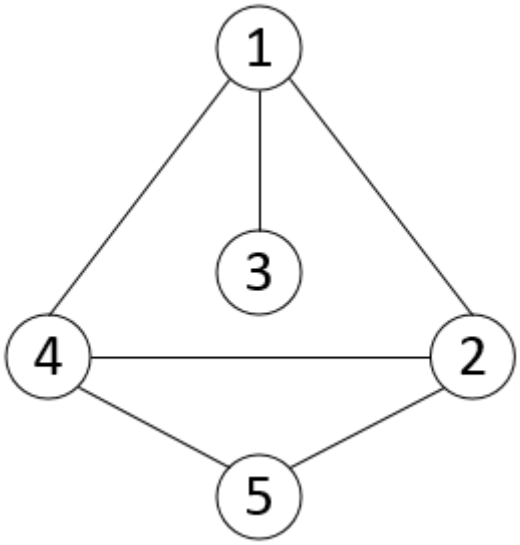

Figure 5.3: An example graph

Suppose, node 1 is the source and node 5 is the target. First, the algorithm checks if the partial solution that has been generated so far is valid for the initial problem. This check takes place in the beginning of each call to the backtracking function. On the first step the partial solution is empty, so it is not valid for the initial problem. Then, the algorithm generates candidate edges that are adjacent to the node 1. They are: (1,4), (1,3) and (1,2). The edge (1,4) is being added to the partial solution and a recursive call to the backtracking function is made. On the second step, as long the partial solution is not complete, new candidate edges are generated. In this particular case they are: (4,5) and

(4,2). The algorithm adds the edge (4,5) to the partial solution and makes a recursive call again. Now, the partial solution (1,4), (4,5) is complete, so it is being reported and the backtracking function terminates. Now, the recursion is one step upper, the candidate edge (4,5) is removed from the partial solution, the second candidate edge (4,2) is added to it and again, a recursive call to the backtracking function is made. There are two candidate edges that have the node 2 as their source: edges (2,1) and (2,5). The edge (2,1) cannot be added to the partial solution, because it already contains an edge that is adjacent to the node 1. Hence, the edge (2,5) is added to the partial solution and the following recursion call detects that it is complete. So, the solution (1,4), (4,2), (2,5) is reported. The algorithm then continues to work in the same manner till all solutions are reported.

Examining different candidate edges on each step ensures that each solution is generated only once. Furthermore, the depth-first fashion of the graph traversal gives an opportunity not to store all valid solutions found so far, but rather to report them immediately. The complete algorithm description can be found in [21].

Then, having all possible paths from access points to assets, the algorithm will look for single-link threats along each path and group them into threat sets, which were defined previously. After that, the algorithm will search for relevant threat set dependency schemes, which correspond to the threat sets identified on the path before. All threat scenarios, whose dependency schemes match the present threat sets, will be added to the end report. To give an idea how this algorithm is going to work, assume the algorithm is going to work on a system model, whose structure was defined previously in this document:

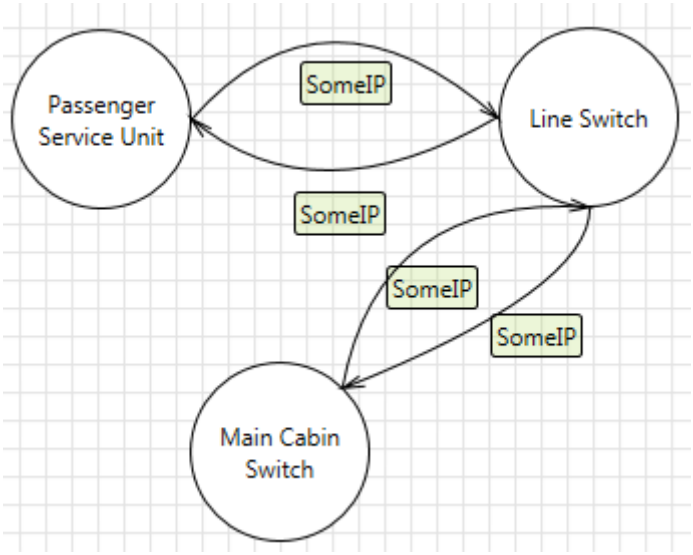

Figure 5.4: Prototype system model

As mentioned in the above XML fragments, the access point in this system is the Pas-

senger Sevice Unit and the asset is represented by the Main Cabin Switch. Pseudocode for generating threat scenarios can be expressed as

Listing 5.9: Pseudocode for generating threat scenarios

```
1 GenerateThreatScenarios(threatSetKB, scenarioKB, graph):
2   threatScenarioList := {}
3   paths := Backtrack(graph)
4   for p in paths:
5    threatSets: = FindThreatSets(p, threatSetKB)
6    threatScenarios := FindScenariosByThreatSets(
7         threatSets, scenarioKB)
8    threatScenarioList <- threatScenarios
9   return threatScenarioList
10
11 FindThreatSets(path, threatSetKB):
12  threatSets := {}
13  for threatSet in threatSetKB:
14   isThreatSetSuitable := true
15   for threat in threatSet:
16    if !pathContainsThreat(path, threat):
17     isThreatSetSuitable := false
18     break
19   if isThreatSuitable:
20    threatSets <- threatSet
21  return threatSets
22
23 FindScenariosByThreatSets(threatSetList,  scenarioKB):
24  resultScenarioList := {}
25  threatScenarios :=  scenarioKB
26  for threatScenario in threatScenarios:
27   for threatSet in threatSetList:
28    for scenarioThreatSet in threatScenario.threatSets:
29     if scenarioThreatSet == threatSet:
30      scenarioThreatSet.marked := true
31      break
32
33  for threatScenario in threatScenarios:
34   isScenarioSuitable := true
35   for scenarioThreatSet in threatScenario.threatSets:
36    if !scenarioThreatSet.marked:
37     isScenarioSuitable := false
38     break
```

```
39    if isScenarioSuitable:
40      resultScenarioList <- threatScenario
41  return resultScenarioList
```

The algorithm consists of three functions: GenerateThreatScenarios, FindThreatSets and FindScenariosByThreatSets. The entry point of the algorithm is the function GenerateThreatScenarios that receives the graph structure and the knowledge base containing information about threat sets (see listing 5.5) and threat scenarios (see listing 5.6 and figure 5.2) as input. The execution logic of all these functions is now going to be described.

GenerateThreatScenarios. At the beginning an empty list of threat scenarios is initialized (line 2). Then, the program searches for all simple paths in the graph from access points to assets using Depth-First-Search/backtracking algorithm (line 3). In the current example there is only one access point – Passenger Service Unit. After that the program loops through the list of paths (line 4) and identifies which single-link threats are possible along this path and whether they form combinations that correspond to threat sets defined previously in the knowledge base (line 5). This is done using the function FindThreatSets that is going to be described later. The extracted threat sets are stored in the list threatSets. In the current example all four single-link threats from the knowledge base are to be identified on the second path. Those are: Sniffing Network Traffic (interaction between LS and MCS), Action Spoofing (outgoing link from PSU), Argument Injection (link between LS and MCS) and Accessing/Modifying/Executing Executable Files (outgoing link from LS). All these threats are present on the second path, because it contains all types of interactions that are required by those threats. Respectively, the algorithm identifies that those threats can be combined to threat sets. Threat set "Corruption of cabin IP device" requires two single-link threats to be fulfilled, whereas "Eavesdropping traffic" and "Traffic injection" require only one single-link threat each. All single-link threats, which are necessary for these three threat sets are present along the current attack path. On the next step, when all potential threat sets are identified, the algorithm looks up into the knowledge base again to find out if those threat sets are enough to match a threat scenario (lines 6 and 7). This is done using function FindScenariosByThreatSets that is going to be described later. If the found threat sets match some threat scenarios in the knowledge base, then they are added to the final list of threat scenarios (line 8), which is returned as an output of the algorithm in the end. In the current case three identified threat sets comprise a threat scenario denoted on the figure 2. Hence, this threat scenario is added to the end report.

FindThreatSets. This function has a goal to identify single-link threats along given path and to extract those threats from the knowledge base, which can be formed from the found single-link threats. Two parameters are received as input: a simple path in the graph from an access point to an asset and threat set knowledge base. In the beginning an empty list threatSets that is going to contain relevant threat sets is created (line

12). Then the function loops through all threat sets defined in the knowledge base to retrieve only the relevant ones for the current attack path (line 13). This is done via another for-loop over single-link threats within the current threat set (line 15). On each iteration the algorithm checks, whether a current single-link threat is possible along the attack path (line 16), and if not, then the whole threat set is filtered out (lines 17 and 18). Otherwise, if all single-link threats are present on the current attack path, then the current threat is added to the resulting list (lines 19 and 20). In the end the list threatSets with all relevant threat sets along the given path is returned (line 21).

FindScenariosByThreatSets. The task of this function is to identify, which threat scenarios from the knowledge base can be executed on the given path in the model, knowing possible threat sets (malicious activities) that are feasible on this path. For the execution of this function two parameters have to be specified: a list of threat sets that were denoted as possible ones on the attack path and the knowledge base containing information about end-to-end threat scenarios. The starting point of this function is the initialization of an empty list resultScenarioList that is going to store plausible threat scenarios (line 24). Then the algorithm copies the knowledge base of threat scenarios to the list threatScenarios (line 25) and loops through it (line 26) in order to find applicable ones. In a second-level for-loop the algorithm iterates over the provided list of threat sets (line 27) in order to check if they are present in the current threat scenario. For this reason a third-level for-loop is needed (line 28), because each threat scenario is a structure of one or multiple threat sets and dependencies between them. If currently examined threat set is present in the threat scenario (line 29), then its flag marked is set to true (line 30). The idea of this marking is that only those threat scenarios, which have all threat sets marked, will be added to the final report. The logic of avoiding irrelevant threat scenarios is implemented in the second first-level for-loop. The algorithm iterates over the list of all threat scenarios (line 33) and checks if all threat sets within this scenario are marked. For that a flag isScenarioSuitable is used (line 34). The second-level for-loop iterates over threat sets within the current end-to-end scenario (line 35) and discards the whole scenario if at least one threat set is not marked (lines 36, 37 and 38). Otherwise, current threat scenario is being added to the resulting list (lines 39 and 40). Finally, the list containing all relevant end-to-end threat scenarios is returned (line 41).

5.7 Algorithm complexity

In this section the time complexity of the threat scenario generation algorithm will be examined. The time complexity of this algorithm can be expressed as $\max(O(N^N ABC), O(N^N ADEF))$, where N is the number of vertices in the model graph, A is the number of paths from access points to assets in the graph, B is the number of threat sets defined in the knowledge base, C is the number of threats in those sets, D is the number of threat

scenarios defined in the knowledge base, E is the average number of threat sets along attack paths and F is the average number of threat sets in end-to-end scenarios. The $O(N^N)$ complexity arises from the modified Depth-First-Search algorithm for enumerating all paths from the source node to the target node in the graph-based system model. The runtime of the algorithm is exponential due to the number of simple paths between two nodes in a graph in the general case. If a graph has a clique structure, then the amount of simple paths between any two nodes is exponential. Although exponential complexity is not practicable in the general case, the runtime stays still feasible for actual applications because graph-based representations of modern hardware/software systems do not tend to have dense structures. For instance, enumerating paths between two nodes in the model of the cabin core system, which is developed by Airbus Group, takes 3 milliseconds. O(ABC) stands for the invocation of the function FindThreatSets from GenerateThreatScenarios (O(A) is the number of iterations of the for-loop in the main function GenerateThreatScenarios and O(BC) is the number of iterations of two nested for-loops in the function FindThreatSets). Similarly, O(ADEF) stands for the running time of the function FindScenariosByThreatSets that is being invoked O(A) times from the main function GenerateThreatScenarios and runs in O(DEF) time due to three-level nested for-loop itself.

As seen from the complexity estimation, the real execution time of the algorithm, which detects threat scenarios in the model, is strongly dependent on the input provided from the user. If the user provides a knowledge base, where O(BC) > O(DEF), i.e. number of threat sets times the number of single-link threats inside threat sets is larger than the number of threat scenarios times the number of threat sets along the attack path times the number of threat sets in one end-to-end scenario, then the execution time of detecting threat sets along the attack paths (fulfilled by the function FindThreatSets) is going to take the most significant part of the algorithm's runtime. In the opposite case the execution time of finding threat scenarios by threat sets in the knowledge base (done by function FindScenariosByThreatSets) becomes more important for the overall computing time.

5.8 Outputs

In this section the outputs of the designed tool will be presented. As mentioned before, a report containing feasible threat scenarios has to be provided in the tabular form. A fragment of such a report is provided in the following table. This small report contains information about the identified threat scenario based on the model and knowledge base of the example system defined previously in this chapter.

Threat Condition	Cause	Origin	Scenario Description

Corruption of cabin IP device	Eavesdropping and injection of malicious traffic	Attack from a corrupted Cabin IP Device	First, the attacker has to exploit "Action Spoofing" and "Accessing, Modifying or Executing Executable Files" in order to conduct the activity "Corruption of cabin IP device". After successfully executing "Corruption of cabin IP device" the attacker has to exploit "Sniffing Network Traffic" in order to conduct the activity "Eavesdropping traffic" and "Argument Injection" in order to conduct the activity "Threat Injection". After successfully executing "Eavesdropping traffic" and "Threat Injection" the attacker can execute "Eavesdropping and injection of malicious traffic" threat scenario.

Table 5.1: Automatically generated report

This table contains a single threat scenario description. The report is designed to be similar to one produced manually during the risk analysis at Airbus Group using a proprietary methodology based on MEHARI. There are four columns in the tabular report.

- Threat Condition – the value for this field is taken from the structure of the corresponding threat scenario, specifically this value stands for the name of the left-most threat set in the dependency scheme. The reason for putting the name of exactly this threat set to the report is because it indicates the starting point of an end-to-end attack, i.e. there are no preconditions for executing this malicious activity.
- Cause – this field implies the name of the threat scenario.
- Origin – the values of this field is defined by the value of the threat scenario description.
- Scenario description – this field stands for the brief description of the attack steps. The value is going to be produced automatically by parsing the graph structure of the current threat scenario and converting it to the string form.

The aforementioned threat scenario report does not contain numerical assessments of different aspects regarding end-to-end threat scenarios. This part is left to be filled entirely by the security expert, because the tool is not designed to make any numerical evaluations. Also, the security expert is free to edit the values of all table cells to make the report more informative, specific and human-readable.

5.9 Summary

This chapter described a concept of the new threat modeling tool that is going to automate end-to-end threat analysis. The new tool is going to be a part of the multistep threat modeling workflow that was presented in the previous chapter. The operation principle of the tool was described step by step, showing a prototype example of how it would look like on a small example system. The time complexity of the end-to-end threat detection algorithm was estimated as well, and it was concluded that it highly depends on the input provided by the user. A deeper analysis of feasibility of the algorithm regarding its time and space complexity will be provided in the Evaluation chapter.

Chapter 6

Evaluation

In this chapter theoretical and practical work that was accomplished during this Master Thesis are evaluated. The evaluation comprises the analysis of the existing risk assessment that was manually produced by Airbus Group experts and the improvements achieved by developing an automated end-to-end threat modeling approach. This chapter also examines the feasibility of the developed end-to-end threat scenario detection algorithm that was described in the Low-level Concept chapter. Finally, the developed concept of the end-to-end threat modeling tool is critically judged.

6.1 Threat mapping

The first step to assess the existing risk analysis and to come up with a concept of automated end-to-end threat detection was to carry out threat mapping. This process was described in the chapter that was dedicated to high-level concept of the automated end-to-end threat modeling. To accomplish threat mapping first an independent single-link threat analysis was made. For this a large list of potential threats that are feasible in the cabin system was collected from open sources. This list contained overall 209 threats, which were considered as single-link threats and assigned to the components of the cabin mockup. To be able to compare the outputs of the existing risk assessment provided by Airbus Group and the results of the independent threat analysis it was decided to extract single-link threats from end-to-end threat scenarios. The idea was go through each end-to-end threat scenarios and to find out, which single-link threats are exploited during them.

In the end of the threat mapping each single-link threat that was extracted from threat scenarios was matched to ones from the list of independent analysis. It is worth mentioning that the mapping covered all single-ling threats that were extracted from threat scenarios, i.e. threat scenario list did not contain any single-link threats that were

not taken into consideration during the independent threat analysis. However, the matching of single-link threats was not perfect. This happened due to the fact that threat scenarios contained usually general single-link references. For example, a threat scenario description contained the following expression: "An attacker could flood the LS interfaces." In this threat scenario, flooding the interfaces of a line switch was considered as a single-link threat and it was mapped to the following threats from the open sources: TCP Flood, ICMP Flood, UDP Flood, CAM table overflow attack, DHCP starvation attack. This example illustrates that the single-link threat extracted from a threat scenario is more general and spans a wider range of malicious actions than ones from the independent threat analysis. Similar situation occurred with threats extracted from the most threat scenarios. This circumstance led to the idea that single-link threats from the independent risk analysis were too specific in contrast to those from the existing risk assessment. This was the first major difference that was identified between the present risk assessment and the separately held analysis. Assuming that in the future risk assessment is going to be done using a set of tools that implement the concept described in the chapters 3 and 4, one can define less specific single-link threats in the knowledge base in order to mitigate the difference that was revealed after the threat mapping stage. Alternatively, the experts may accept the concrete nature of threats, but in this case the final report containing a list of end-to-end threats is going to be larger.

In the end of threat mapping all single-link threats extracted from the existing risk analysis were matched to one or multiple threats from the independent analysis. Additionally, some threats from the independent threat analysis were left unmapped. This was the second major difference between the existing threat analysis and the independent one. The list of unmapped threats indicated the deficiency in the existing end-to-end threat analysis. Therefore, an important improvement of the automatic threat identification over the manual risk assessment is the absence of the human factor during the report generation. When using the tool chain for automated threat identification, if the expert decides to enlarge the number of potential threats, then he has to add them to the knowledge base. In contrast, if the expert decides to add potential threats to the existing manually produced risk assessment, then he has to examine system components, where newly added threats are possible, go through each threat scenario in order to check if new threats affect the existing scenarios, and define new threat scenarios that arise due to new threats. All operations have to be done manually. The framework of automated threat identification facilitates the easier extension of the threat knowledge base. The list of unmapped threats from the independent threat analysis can be integrated into the automatically produced risk assessment easier than into the manual one.

The complete table of the threat mapping can be found in the attachment. Basically, it consists of a table with threat scenarios and lists of single-link threats, which were mapped to those scenarios.

6.2 Advantages and disadvantages of the automated concept

An important disadvantage of the automated end-to-end threat modeling approach is the human-readability. The developed concept offers the generation of the report using only predefined sentence structures, whereas during the manual report each threat scenario can be reviewed individually and formulated in detail. For this reason the developed concept does not strictly specify, whether the output of the threat analysis is mutable or immutable. The opportunity to update the information within threat scenarios has to be left for the expert, if necessary. Additionally, the implementation of the aforementioned concept has to provide an opportunity for the user to fill in the properties that are responsible for the potentiality and impact of threat scenarios. For this purpose, a corresponding user interface should be provided for filling in the numerical and textual values. The concept of the automated threat identification does not cover the potentiality and impact evaluation. These values have to be provided by the user.

The last important difference between the approach of the developed concept and the one used during the manual analysis is workflow automation. Through the automatic processing of threat searching the human factor is avoided during the threat identification stage. This could be useful for large systems that are too complex to be grasped by the expert at once. While doing a manual threat analysis in a system containing many components and interactions, there is a probability of missing threats. Conversely, the developed concept of automated threat analysis includes an exhaustive search for threats and threat scenarios in the graph model, i.e. missing threats due to human factor is evicted.

6.3 Limitations of the end-to-end tool

The crucial part of the developed tool chain concept is the new tool that is aimed to automate threat scenario detection (component number 7 in the figure 4.1). This tool is yet to be implemented, and therefore it is marked with a dashed border in the figure 4.1). The low-level design description of this tool is provided in chapter 5. The current design of this tool has several limitations.

Firstly, the developed tool concept does not consider the order of single-link threats along the attack path. The current concept provides a possibility to define threat sets (that consist of one or multiple single-link threats) and combine them into threat scenarios. After that the tool finds all possible attack paths between access points and assets, detects all potential single-link threat along each path, and checks if these single-link threats can be combined into valid threat sets. Then, if valid threat sets are detected along the path, the tool checks if these threat sets can be combined into valid threat

scenarios according to the provided knowledge base. At the moment the tool does not consider the order and the location of single-link threat exploitations. For example, an attacker has to exploit a single-link threat, say a buffer-overflow attack, exactly on the access point and before the another attack, say injection of malicious code. At the same time, end-to-end threat detection logic discovers a potential buffer-overflow attack in the end of the path, which can only be exploited after the injection of malicious code, which was detected on the previous part of the attack path. In this case a threat is still going to be generated, because the design does not currently consider the order and the location of single-link threat exploitations. Therefore, current end-to-end threat detection logic can lead to false positives.

Another problem of the developed concept of the end-to-end threat modeling tool is ignoring system configurations. What the tool is currently aimed to do, is finding out whether all single-link threats within corresponding threat scenarios are present along the attack path. However, in some cases this logic may also lead to irrelevant threat scenarios. To give an example of such a situation, a simple system with three components can be considered. The example system consists of a browser client, which has a communication link to a switch that is connected to the database server. A possible threat in this framework is the elevation of privilege by sending malicious queries from the client to the database. This attack is considered as a standalone threat scenario and therefore it will be generated by the end-to-end threat modeling tool, although the switch could be configured in such a way that it filters out all illegitimate queries. Hence, false positives may occur in the end report.

6.4 Algorithm complexity and feasibility

The crucial part of the developed concept for automated threat analysis is the algorithm of generating feasible threat scenarios using the model of the analyzed system and the knowledge base provided by the user. In this section the feasibility of running this algorithm on modern computers is going to be analyzed.

As mentioned in the previous chapter, the time complexity of the described algorithm is $\max(O(N^N ABC), O(N^N ADEF))$, where N is the number of vertices in the model graph, A is the number of paths from access points to assets in the graph, B is the number of threat sets defined in the knowledge base, C is the number of threats in those sets, D is the number of threat scenarios defined in the knowledge base, E is the number of threat sets along the attack path and F is the number of threat sets in one end-to-end scenario. The uncertainty of the algorithm arises from the dependence of the runtime on user input. Although the runtime of threat scenario generation algorithm seems impractical at a glance, it is still feasible for systems like the on-board cabin core system due to the moderate size of the graph-based model. Additionally, there are parallelization

opportunities for the described algorithm.

The Depth-First-Search/backtracking algorithm for enumerating all routes between two nodes in order to find possible attack paths can also be run in parallel. Specifically, in the main backtracking function (see 5.7) there is a for-loop that iterates over generated candidates, which are added them to the partial solution, then the recursive call to the backtracking function takes place, and when it is finished, the previously added candidate is removed from the partial solution. Since after adding the candidate to the partial solution the algorithm does not depend on the logic, when another candidate is added, the execution of the iterations of this for-loop can be done in parallel.

Furthermore, there are other execution parallelization opportunities in the threat scenario generation logic (see listing 5.9). The pseudocode of the function FindThreatSets can be expressed as:

```
11 FindThreatSets(path, threatSetKB):
12  threatSets := {}
13  for threatSet in threatSetKB:
14   isThreatSetSuitable := true
15   for threat in threatSet:
16    if !pathContainsThreat(path, threat):
17     isThreatSetSuitable := false
18     break
19   if isThreatSuitable:
20    threatSets <- threatSet
21  return threatSets
```

And it is being invoked from the following for-loop:

```
4 for p in paths:
5  threatSets:=FindThreatSets(p, threatSetKB)
6  threatScenarios:=FindScenariosByThreatSets(threatSets,
7       threatScenarioKB)
8  threatScenarioList <- threatScenarios
```

As can be seen from the pseudocode, the inner-most for-loop (line 15) can be run in parallel because all threats are examined separately and the iterations do not depend on each other. However, the algorithm will have to run all iterations in the worst case of parallel execution because the iteration that is going to break the loop (line 18) may finish its execution last. Hence, one can run this loop in parallel but cannot rely on pruning. The next-level for-loop (line 13) can also be run in parallel because its iterations do not depend on each other's results. Within an iteration one threat set is extracted from the knowledge base and examined in the inner-most for loop. Due to the fact that the only writing is in the line 20 and it does not affect other loop iterations, the for-loop

in the line 13 can be parallelized. The loop in the line 4 can be also run in parallel for same reasons.

To sum up the evaluation of the algorithm, although it has a potentially high complexity there are still a variety of opportunities to execute it in parallel. Still, in the case of the sequential execution of the algorithm the runtime stays reasonable when modeling most of the systems. The uncertainty of the algorithm's complexity originates from its strong dependency on the user's input.

6.5 Concept evaluation

The main outcome of the whole research is the concept of the tool chain for modeling end-to-end threat scenarios and automatically detecting them. Currently neither open-source nor commercial solutions for end-to-end threat modeling exist on the market. In this section the advantaged and disadvantages of the created concept will be discussed.

The first aspect that has to be emphasized is the safety of threat searching. Comparing it to the manual threat searching method, the algorithm described in the concept guarantees an exhaustive examination of the graph model, avoiding the possibility of overlooking existent threats. In the case of manual threat identification, the probability of a mistake is higher. Therefore, an implementation of this concept would be suitable for a wider range of security experts, including those who do not have a very deep knowledge about the analyzed system and are more likely to make errors while searching for threats manually. The developed concept provides a more universal solution for threat modeling.

The second useful feature of using the developed concept in the process of risk assessment is its automation. By getting rid of several routine steps that would otherwise have to be done manually by experts automated threat detection saves time and hence adds flexibility and reduces costs required for risk analysis.

The developed approach has also some drawbacks. The first of them is the large size of the knowledge base that the user has to provide prior to modeling the system. Each entity in the knowledge base, such as system components, data flow types, threat types, assets and access points, has to be described in XML format. Filling in a knowledge base is a time consuming operation. However, once it is filled, the user can build a model of any structure and obtain the report containing threats automatically. The knowledge base can also be dynamically adjusted and complemented. The concept presumes that the tool will deliver the actual report taking latest changes in the knowledge base into consideration without the need to rebuild the model.

Another drawback of the developed design of automated end-to-end threat modeling is the simple structure of the final output. In comparison with the manually accom-

plished threat analysis, where the author can put all his knowledge and expertise into the description of each threat scenario, the automated approach is only able to parse the knowledge base and the raw data from there. For this reason it is assumed the implementation of the concept will provide the output with editable fields to give an opportunity to the user to provide a more meaningful and detailed information about threat scenarios.

The present state of the art in the field of automating end-to-end threat modeling does not allow estimate the success of the developed approach and the advance in the research area. The developed approach is rather a starting point for further investigations of the problem. In order to ascertain or refute the capability of the developed concept to solve concrete problems successfully it is necessary to:

- Implement the tool described in chapter 5 and build up the whole chain of programs that is aimed to address end-to-end threat modeling
- Test the developed tools by applying them to threat modeling in different systems and checking the obtained results

As long as until now no practical tools for end-to-end threat modeling existed, the developed concept can be considered as an improvement in the area of risk assessment solutions. Despite the drawbacks of the developed concept, it can nevertheless be considered as a starting point for future tools for end-to-end threat analysis. Further researches and enhancements of this concept should improve the existing state and bring the end-to-end threat modeling to the next level.

Chapter 7

Conclusion

Previous chapters expressed the need for risk assessment automation and proposed a concept of a tool chain that is aimed to satisfy this demand. To achieve this, threat modeling was adopted as a technique that is aimed to introduce automation opportunity into risk assessment process. In order to discover the best possible threat modeling application for risk assessment, a comprehensive research of the scientific literature and existing solutions for threat modeling was made. Investigation of threat modeling solutions demonstrated the missing capabilities of existing tools. Hence, it was decided to build a new concept of automated threat modeling that would be suitable for assessing risks in aircraft cabin systems.

The developed concept depicts the usage of an existing threat modeling solution together with a new tool, which is yet to be implemented. An important part of the developed tool chain is the usage of Microsoft Threat Modeling Tool 2016, which is suitable for modeling aircraft cabin systems. The main advantage of the Microsoft Threat Modeling Tool 2016 is the way, how system models are designed – data flow diagrams. This approach is similar to the one used during manual risk assessment of the cabin core system done by Airbus Group experts. However, neither Microsoft Threat Modeling Tool nor other existing threat modeling solutions are able to cover an important aspect of risk assessment – modeling end-to-end threats. An important feature of the cabin core system is the presence of multiple hardware and software components, multiple access points for potential attackers and multiple assets that must be protected. Such a framework facilitates the possibility for an attacker to exploit vulnerabilities in the system and combine them with normal system functions. By doing this an attacker can gain access to system components that are not directly available for the attacker by system design. These kinds of security attacks are considered as end-to-end attacks, which cannot be modeled with existing threat modeling solutions. The main task of the developed tool chain is to model end-to-end threats in order to raise the awareness of end-to-end attack threats for security experts during the risk assessment process.

The new tool is aimed to cover missing capabilities and ensure automated threat modeling that would be suitable for airborne systems. An independent threat analysis with the assistance of Microsoft Threat Modeling Tool was made in order to define the requirements for the new tool and to find out the missing features of the manual risk analysis. The list of threats involved into the independent threat analysis were mapped to ones extracted from threat scenarios of the existing manual risk analysis. This step is called “Threat mapping”. Threat mapping helped to specify the functionality of the new tool. The independent threat analysis and threat mapping pointed out two major drawbacks of the manual risk analysis: a list of threats that were not covered in the existing threat analysis and the generality of present threats. Based on this information, a concept of the tool chain for automated threat analysis was proposed. Both threat mapping and low-level concept of the new tool are described in detail. The developed concept is aimed to address both disadvantages of the existing manual risk analysis and to enable automated threat scenario generation. Finally, the accomplished work and the proposed threat modeling concept were critically evaluated.

7.1 Future work

As long as the main outcome of this Master Thesis is the concept of the tool chain for end-to-end threat modeling, future work is to improve and implement it. In order to check the relevance of the output produced by the described tool chain it is necessary to implement the end-to-end threat modeling tool and to build up the framework that includes all the parts of the developed concept into one standalone system. Currently, the proposed concept represents the usage of Microsoft Threat Modeling Tool together with the newly designed end-to-end threat modeling tool. For the sake of good user experience, both solutions should be integrated into one system.

As described in the Evaluation chapter, there are limitations in the concept of the end-to-end threat modeling tool – the presence of false positives and disregard of the single-link threat order. Presumably, proposing solutions for both problems and implementing them will lead to better results of the whole tool chain. Currently the output of the pure implementation of the developed concept is going to contain unfeasible threat scenarios in some cases. By solving these open problems the output of the tool chain is going to become more relevant to real end-to-end threats. Specifically, addressing these problems is going to avoid inappropriate threat scenarios if the order of single-link threat exploitations is not practicable or system components are configured in such a way that exploited threats can be neglected.

For enumerating all simple paths in the data flow diagram the design of the new end-to-end threat modeling tool currently proposes the usage of the modified breadth-first-search algorithm. A further step of the concept improvement could be the investigation

of more efficient algorithms for path enumeration. For example, in his paper [16] Frank Rubin proposes an algorithm, which is based on matrix operations. The paper claims to enumerate all simple paths between source and destination nodes using $O(N^3)$ matrix operations. Additionally, if this algorithm appears to be useful in the current framework, the opportunity of its execution parallelization has to be explored.

Finally, in order to be convinced that the developed concept is a useful solution for automated risk assessment, it is necessary to apply it on multiple systems. Using the developed tool chain for modeling different system structures will show if it is applicable for these systems. The modeled systems should be diverse, so that all possible disadvantages of the developed framework are identified, if any. After that, in the case if the application of the developed framework detects its weaknesses, new solutions for mitigating those weaknesses should be proposed and implemented.

Bibliography

[1] R. Shirey, "Rfc 4949 - internet security glossary," Tech. Rep., 2007.

[2] R. P. N. Thanthry, M. S. Ali, *Security, Internet Connectivity and Aircraft Data Networks.* IEEE, may 2006.

[3] K. S. Amril Syalim, Yoshiaki Hori, *Comparison of Risk Analysis Methods: Mehari, Magerit, NIST800-30 and Microsoft's Security Management Guide.* IEEE, mar 2009.

[4] K. H. S. o. C. Xiaohong Li, *A Unified Threat Model for Assessing Threat in Web Applications.* IEEE, 2008.

[5] G. M. J.-L. R. N. V. Dominique Buc, Jean-Philippe Jouas, *Mehari Risk Analysis Guide.* Club de la Sécurité de l'Information Français, 2004.

[6] L. R. Y. W. R. W. Richard A. Caralli, James F. Stevens, *Introducing OCTAVE Allegro: Improving the Information Security Risk Assessment Process.* Software Engineering Institute at Carnegie Mellon University, 2007.

[7] *Magerit - version 2 - Methodology for Information Systems Risk Analysis and Management.* Ministerio de Administraciones Publicas, jun 2006.

[8] A. F. Gary Stoneburner, Alice Goguen, *Risk Management Guide for Information Technology Systems, NIST Special Publication 800-30.* National Institute of Standards and Technology, 2002.

[9] *The Security Risk Management Guide.* Microsoft, 2006.

[10] P. A. Xavier Depin, *Risk Management Methodology for Aircraft Security.* Airbus Group Innovations, oct 2008.

[11] *Three Approaches to Threat Modeling.* http://myappsecurity.com/approaches-to-threat-modeling/, 2012.

[12] U. M. Nazir A Malik, Muhammad Younus Javed, *Threat Modeling in Pervasive Computing Paradigm.* IEEE, nov 2008.

[13] M. E. Paul Saitta, Brenda Larcom, *Trike v.1 Methodology Document*, jul 2005.

[14] L. B. Krishna Sampigethaya, Radha Poovendran, *Secure Operation, Control, and Maintenance of Future E-Enabled Airplanes.* IEEE, dec 2008.

[15] A. E. A. Caroline Möckel, *Threat Modeling Approaches and Tools for Securing Architectural Designs of an E-Banking Application.* IEEE, aug 2010.

[16] F. Rubin, *Enumerating All Simple Paths in a Graph.* IEEE, aug 1978.

[17] D. P. D. Shafiq Hussain, Dr. Harry Erwin, *Threat modeling using Formal Methods: A New Approach to Develop Secure Web Applications.* IEEE, sep 2011.

[18] M. M. Dr. Oliver Hanka, Nils Tobeck, *System Description and Risk Analysis of the Critical Cabin Infrastructure.* Airbus Group Innovations, jul 2015.

[19] *CAPEC - Common Attack Pattern Enumeration and Classification.* https://capec.mitre.org, 2015.

[20] L. Phifer, *A list of wireless network attacks.* http://searchsecurity.techtarget.com/feature/A-list-of-wireless-network-attacks, 2009.

[21] S. S. Skiena, *The Algorithm Design Manual.* Springer, 2008.