

A Semi-Automated Approach to Asset-Based Security Risk Assessment

- Design, Implementation, and Evaluation of Thalaura: A Threat
Modeling and Risk Analysis Tool

Semi-automatisk metod för tillgångsbaserad riskbedömning

Max Mogren

Supervisor : Zelong Wang
Examiner : Andrei Gurtov

External supervisor : Emil Norberg

Upphovsrätt

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida <http://www.ep.liu.se/>.

Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years starting from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: <http://www.ep.liu.se/>.

Abstract

As cybersecurity threats continue to grow in complexity and scale, the need for efficient, consistent, and scalable threat modeling methods becomes important. Incorporating threat modeling into the software development lifecycle (SDLC) is also necessary to ensure that software development conforms to security requirements. However, traditional manual threat modeling approaches are often time-consuming and resource-intensive. Moreover, many existing tools lack automation or the integration capabilities required to support modern development practices and product security risk assessments effectively.

This thesis presents a semi-automated method for performing asset-centric security risk assessments, integrating STRIDE threat classification with the OWASP Risk Rating methodology to identify and evaluate risks within software architectures represented by Data Flow Diagrams (DFDs). To make this possible, a web application called **Thalaura** is created, implementing a novel risk-generation algorithm that considers vulnerabilities and threat vectors combined with certain aspects of the architecture to generate risk regarding the confidentiality, integrity, and availability of assets in the system. Furthermore, security experts can configure security controls that threat modelers can use to mitigate risks detected in the system.

To evaluate the performance of Thalaura, its outputs were compared with the existing automated threat modeling tools: Microsoft Threat Modeling Tool and Threat Agile. Results showed that Thalaura produced more comprehensive and granular risk descriptions, offering greater visibility and consistency by following a clear methodology and Zero Trust approach. A quantitative analysis confirmed strong correlations between configured inputs and risk scores, while qualitative testing demonstrated effective mitigation recommendations through applied security controls, which enable dynamic threat modeling.

Acknowledgments

I want to express my sincere gratitude to my external supervisor, Emil Norberg, for the continuous support throughout the development of this thesis and the accompanying application. I especially want to thank you for always being available to discuss ideas, provide perspective, and engage in thoughtful conversations. I would also like to thank my supervisor, Zelong Wang, and examiner Andrei Gurtov for guiding me through the thesis work and providing insightful feedback. Finally, I would also like to thank my opponent, Danijel Grujicic, for his thoughtful feedback on my thesis.

Contents

Abstract	iii
Acknowledgments	iv
Contents	v
List of Figures	vii
List of Tables	viii
1 Introduction	1
1.1 Motivation	1
1.2 Aim	2
1.3 Research questions	2
1.4 Approach	2
2 Theory	3
2.1 Information Security	3
2.2 Cybersecurity	4
2.3 Threat modeling	4
2.4 Zero Trust	7
2.5 OWASP Risk Rating Methodology	7
2.6 Risk Management Strategies	9
2.7 Related Work	10
3 Method	15
3.1 Overview of Thalaura	15
3.2 Identifying vulnerabilities with STRIDE	20
3.3 Security Controls	24
3.4 Threat model	26
3.5 Monte Carlo Simulation	28
4 Results	29
4.1 Qualitative results	29
4.2 Quantitative results	35
5 Discussion	39
5.1 Qualitative results	39
5.2 Quantitative results	40
5.3 Critique of Method	41
5.4 The work in a wider context	44
5.5 Answering the Research Questions	44
6 Conclusion	46

6.1 Conclusion	46
6.2 Future work	46
Bibliography	48
A Appendix	51

List of Figures

3.1	Vulnerability and threat vector data types	15
3.2	Security controls and security control properties data types	16
3.3	Configuration window and DFD canvas	19
3.4	Risk calculation tree for the OWASP Risk Rating Methodology	20
3.5	Example of security control page for TLS 1.3	25
3.6	Data flow diagram over the threat model	27
3.7	Data flow diagram over the threat model with security controls	28
4.1	Data Flow Diagram in Microsoft Threat Modeling Tool	29
4.2	Data Flow Diagram in Threat Agile	30
4.3	Data Flow Diagram in Thalaura	30
4.4	Threat and risk identification distributions	31
4.5	Difference of risks between system A and B	31
4.6	Risks as presented in Thalaura	34
4.7	Risk Distribution of Monte Carlo Simulation	35
4.8	Distribution of the Amount of Risk Ratings Across STRIDE	36
4.9	Distributions of Factors by Risk Ratings	38

List of Tables

2.1	Description and visualization of DFD components	5
2.2	STRIDE-per-element	7
2.3	Likelihood and impact level categorization	8
2.4	Risk Matrix	9
3.1	Technical impact descriptions and values from the OWASP Risk Rating documentation [26]	17
3.2	Business impact values for Privacy Violation, Non-Compliance, Financial Damage, and Reputation Damage	18
3.3	Description of probability factors	20
3.4	STRIDE mapping to security properties	21
3.5	STRIDE threats against asset states	21
3.6	Vulnerabilities against an asset in transit	22
3.7	Vulnerabilities against an asset in use	22
3.8	Vulnerabilities against asset at rest	23
3.9	Vulnerabilities against security controls and their properties	24
3.10	Technical impact of assets in Thalaura	26
3.11	Threat impact of assets in Thalaura	26
3.12	Quantity and technical impact of data assets in Threat Agile	27
4.1	Sample of detected risks in Thalaura	32
4.2	Sample of detected risks in Threat Agile	33
4.3	Sample of detected threats in Microsoft Threat Modeling Tool	34
4.4	Risk Distribution Across STRIDE Categories	35
4.5	Correlation between a set of parameters against the impact factor	37
4.6	Correlation between a set of parameters against the probability factor	37
4.7	Five-number summary for Probability Factor by Risk Rating.	38
4.8	Five-number summary for Impact Factor grouped by Risk Rating.	38
A.1	Final Correlation Matrix with STRIDE Categories Ordered and Risk Ratings Last	52



1 Introduction

This chapter presents the motivation and aim of the thesis and the questions this thesis aims to answer. Finally, the approach to answering these questions is given.

1.1 Motivation

Threat modeling is an important process in risk assessment where specific threats against a system are identified and prioritized. This process is ideally started early in the Software Development Life Cycle (SDLC) and refined during the development as the software grows [14]. The information concerning these threats can be used to decide on the design of the architecture or what actions to take against the specific threats identified.

However, successfully integrating threat modeling into the SDLC presents many challenges, and there is no clear evidence of the best solution [2, 22]. One promising method for solving some of the challenges is using tools that automate parts of the threat modeling process, such as the Microsoft threat modeling tool (MSTMT), OWASP Threat Dragon, PyTM, and Threat Agile [8]. These tools attempt to reduce the time developers are required to spend doing the threat modeling, which usually involves security experts, consequently minimizing the resources necessary to invest in the threat modeling process.

While many different methodologies and approaches have been made in threat modeling, these new techniques are not sufficiently evaluated and usually lack tool support [22]. Even though some of the tools are open-source and open for contributions, there is a lack of documentation and research regarding the development of underlying mechanisms. For broader adoption and reliability, automated threat modeling tools would greatly benefit from more research and comprehensive documentation on how the threat identification and risk generation functions.

Furthermore, in the context of new telecom products, there may be no predefined threats or vulnerabilities associated with the specific technologies used in the implementation. This significantly limits the effectiveness of existing tools and often requires extensive manual effort to define relevant threat scenarios. These challenges highlight the need for a more flexible and adaptable tool that can support emerging technologies without relying solely on predefined threat libraries.

1.2 Aim

This thesis aims to enhance the field of automated threat modeling by developing a novel risk identification algorithm, which will be integrated into a tool capable of automatically generating and assessing security risks from a Data Flow Diagram (DFD). Through the creation and implementation of this algorithm, the goal is to streamline the identification of system vulnerabilities, reducing manual effort and improving accuracy. Furthermore, the research will critically evaluate existing automated threat modeling approaches, identify their strengths and limitations, and contribute to advancing risk detection methodologies in software security.

By exploring the design, development, and evaluation process of this automated risk detection tool, this thesis seeks to meaningfully contribute to the development of more efficient and scalable cybersecurity tools. The broader aim is to address gaps in current automated risk assessment methods and propose improvements that can benefit both researchers and practitioners in the field of cybersecurity.

1.3 Research questions

1. What data and metadata from data flow diagrams (DFDs) are necessary for developing and implementing an asset-centric risk algorithm using STRIDE threat modeling?
2. How can the identified data and metadata be applied to assess and prioritize security risks?
3. How does the proposed algorithm compare to other threat modeling tools in terms of accuracy, coverage, and consistency?

1.4 Approach

To address the research questions in this thesis, the first step is to determine what relevant information can be extracted from DFDs for automated threat modeling. This will involve analyzing the components of DFDs, such as data flows, processes, and data stores, to identify which elements can provide meaningful input for risk assessment. Additionally, it is essential to define what supplementary information the user must provide to enhance the accuracy of the threat modeling process.

In addition to the structural elements of DFDs, the approach may also require integrating up-to-date vulnerability information, sourced from existing security databases or frameworks, to ensure that the risk identification algorithm can effectively map potential vulnerabilities to specific system components.

Finally, a web application will be developed to implement and demonstrate the proposed threat modeling approach. The application will serve as a practical proof of concept, enabling users to create data flow diagrams, automatically identify risks based on integrated vulnerability sources, and generate risk assessments in a consistent and scalable manner. This tool will not only facilitate the evaluation of the methodology but also provide a foundation for further development and integration into real-world software development workflows.



2 Theory

This chapter explains the foundations for understanding threat modeling and security risk assessment within software development. Firstly, brief introduction to information security and cyber security is presented, highlighting key differences in their roles for protecting information. Then, the section explains key concepts within threat modeling and frameworks such as STRIDE, Data Flow Diagrams (DFDs) and the asset-centric threat modeling approach. Also a brief introduction to Zero Trust is given and a more thorough description of the OWASP Risk Rating methodology used for the risk assessment in addition to risk management strategies. Finally, a review of related work is presented highlighting state-of-the-art approaches to threat modeling.

2.1 Information Security

The ISO/IEC 27000 family defines information security as the preservation of confidentiality, integrity, and availability of information. In addition, other properties such as authenticity, accountability, non-repudiation, and reliability can also be involved [10].

Information can take many different forms, from digital files to physical documents, and even things like intellectual property. It is important to remember that information security is a continuous process that involves structured risk management. This process helps organizations identify and address potential risks to protect valuable information from threats.

Because information security covers a broad range of concerns, it touches on everything from creating security policies and managing assets to ensuring regulatory compliance and protecting physical infrastructure. It's a wide-reaching field that requires a comprehensive approach to both digital and physical security to reduce risks and safeguard sensitive information.

A distinction should also be made between information and communication technology (ICT) security and information security. ICT security focuses on protecting the technological infrastructure itself, treating the infrastructure as the primary asset to be secured. In contrast, information security is concerned with protecting the information or data, regardless of the technology used to process or store it. From the perspective of information security, ICT can be seen as a source of vulnerabilities that can be targeted by threats to compromise the assets [25].

2.2 Cybersecurity

While information security covers a broad range of practices aimed at protecting all forms of information, whether digital or physical, cybersecurity is a specialized subset that focuses specifically on the protection of digital systems and data within the cyberspace domain. Cybersecurity is concerned with safeguarding computers, networks, servers, mobile devices, and data from malicious attacks, unauthorized access, and damage. It focuses on the technologies, protocols, and processes designed to secure digital assets that are vulnerable to threats via ICT [25].

Although threat modeling originates from the broader discipline of information security, its primary application lies within the domain of cybersecurity. This technique is widely leveraged to proactively identify and mitigate risks across digital systems, networks, and software.

2.3 Threat modeling

The four-step framework is a core concept as discussed by Shostack [21] that focuses on four questions to guide the threat modeling process.

- What are you building?
- What can go wrong?
- What should you do about those things that can go wrong?
- Did you do a decent job of analysis?

The first question focuses on understanding the design of the system being developed. This can be visualized in multiple ways, such as sketches on a whiteboard, but Data Flow Diagrams (DFDs) are the most commonly used approach. The second question addresses identifying threats based on the architectural design outlined in the first step. Numerous frameworks exist to facilitate threat identification and analysis, each offering different risk evaluation methods. However, the most common and mature methodology is the STRIDE mnemonic, a technique used to identify threats based on various categories. Once the threats have been identified, it is possible to conduct risk assessments to quantify risks and understand which threats have the most significant impact. The risk assessment helps to answer the third question, which concerns how to address them. There are four strategies to apply when addressing the threat: mitigating, avoiding, transferring, or accepting the risk.

Furthermore, by tracking and classifying the threats as security bugs, it is possible to integrate the mitigation efforts into the SDLC by creating a task to address in the organization's tracking system. The final step is to validate the threat model. This involves reviewing each identified threat to confirm it has been appropriately addressed. For example, this can be achieved by integrating specific tests into the software to verify that the security bugs have been resolved. It is also vital to ensure that the DFD accurately reflects the system. As changes to the system architecture may occur during software development, these updates must be incorporated into the model to maintain its accuracy.

2.3.1 Asset-Centric Threat Modeling

Shostack describes three structured approaches to threat modeling, which are asset-centric, attacker-centric, or software-centric. Each has its own set of advantages and disadvantages [21]. However, in the asset-centric approach, assets are broadly defined as anything that holds value to the organization, including things to protect, items attackers might target, or even resources that can be used as stepping stones for further attacks. Essentially, an asset is

anything significant to the organization that requires protection, whether it's data, systems, or specific classified documents.

The asset-centric approach focuses on identifying and protecting critical assets within a system. One of the benefits is the ability to prioritize threats based on the value of the asset they compromise, guiding security efforts to focus on the most vulnerable asset. However, this approach can be challenging when there is a lack of clarity about what assets exist within the system. Without a well-defined understanding of the assets, assessing their importance or potential risk exposure is difficult. The current state-of-the-art approaches are DREAD, Trike, OCTAVE, and PASTA. However, in the field of threat modeling, STRIDE is a significantly more recognized approach. While STRIDE is traditionally seen as a software-centric approach, it is possible to derive threats against assets based on whether a component is compromised [18].

2.3.2 Data Flow Diagrams

To answer the first question in the threat modeling process, it is crucial to gain a comprehensive understanding of the system's architecture being analyzed. As previously mentioned, this typically involves creating a DFD that illustrates the specific components and interactions within the system. These components and their respective functions can be seen in Table 2.1.






Name	Symbol	Function
External Entity		Represents external actors that interact with the system, such as users, organizations, or third-party APIs.
Process		Task that receives, modifies, or redirects input to output. Processes act as the core functional elements of a system.
Data Storage		Represents permanent or temporary data storage. Can represent databases, files, tables, or even in-memory caches.
Data Flow		Transfers information between external entities, processes, and data storage systems.
Trust Boundary		A defined area of a system that has some level of privilege.

Table 2.1: Description and visualization of DFD components

An *external entity* is typically the initiator of actions within a system. For instance, this could be a user visiting a website and eventually clicking a button, which triggers a request to a certain process. In this case, the external entity is the user interacting with the system, represented as an anonymous internet user. The interactions between external entities and the system should be carefully monitored, as they often represent a primary source of potential threats to the system.

A *process* can represent many things, but in the case of software, this can be seen as any code being run. This can range from simple scripts to entire services that interact with all parts of the system.

The *data storage* component interacts with processes to either send information to processes or receive instructions from them, such as saving, modifying, or removing data.

Furthermore, the *data flow* component visualizes how all of the system's components interact. This can be, as mentioned earlier, a request from an anonymous internet user or a process requesting information from a data storage to provide that information to the user.

Finally, a *trust boundary* specifies a contained part of the system. Data flows that cross these boundaries are called entry points from which adversaries can interact with the application or provide data [7]. These entry points must exist for an adversary to attack the system, and are thus an area from which many threats surface and cluster around [21].

2.3.3 STRIDE

To answer the second question in the four-step framework regarding what can go wrong, various methodologies can be used to identify threats. The most common and mature process of doing this is the STRIDE methodology. STRIDE is a mnemonic that stands for Spoofing, Tampering, Repudiation, Information disclosure, Denial of Service (DoS), and Elevation of Privilege (EoP), which describes different variations of threats. STRIDE was initially coined by Kohnfelder and Garg [12] and focuses on maintaining the security properties of confidentiality, integrity, and availability.

A spoofing threat occurs when an entity pretends to be someone or something it is not, with the intent to gain unauthorized access or deceive others. This type of threat typically involves impersonation techniques, such as using stolen credentials or crafting fake identities, to bypass authentication mechanisms and exploit trust within a system.

A tampering threat arises when an unauthorized party modifies data, files, network communications, or API calls to achieve malicious outcomes. This can include altering configuration files, manipulating sensitive data, or injecting malicious code into a system, all of which compromise the integrity of the affected asset.

A repudiation threat happens when an individual denies performing an action, making it difficult or impossible to prove their involvement. This threat is often linked to insufficient logging, auditing, or authentication mechanisms in a system. For example, if a system does not securely log transactions, a user might deny transferring funds or modifying data, leading to disputes or accountability gaps.

Information disclosure threats involve the unauthorized exposure of sensitive or confidential information, enabling attackers to access data they should not see. This can occur through vulnerabilities such as insecure storage, plaintext transmissions, or poorly implemented access controls. Examples include data breaches that expose user credentials, private keys, or trade secrets.

Denial of Service threats are attacks aimed at disrupting access to resources critical for providing a service. These threats typically overwhelm a system, network, or application with excessive requests or exploit vulnerabilities to render it unavailable to legitimate users. Such attacks can cause significant downtime and financial losses for affected organizations.

A threat against Elevation of Privilege occurs when an attacker gains higher levels of access or permissions than they are authorized to have. This can involve bypassing authorization checks, exploiting system vulnerabilities, or corrupting processes to perform restricted actions, such as accessing sensitive data or executing administrative commands. This type of threat is particularly dangerous as it can grant attackers complete control over a system.

There are two approaches to identifying threats: STRIDE-per-element and STRIDE-per-interaction. The first approach establishes certain threats against specific components in the data flow diagram. This allows the modeler to focus on a set of threats, which makes it easier to identify them methodologically. The specific threats against those components can be seen in Table 2.2. This approach views the components and data within the components as victims. On the other hand, STRIDE-per-interaction creates a tuple of the interaction, for

example, external entity—dataflow—process. Then, threats are derived by considering that specific interaction and all three components.

	S	T	R	I	D	E
External Entity	✓		✓			
Process	✓	✓	✓	✓	✓	✓
Datastore		✓	✓	✓	✓	
Dataflow		✓		✓	✓	

Table 2.2: STRIDE-per-element

2.4 Zero Trust

A key aspect that can separate threat models is the concept of Zero Trust. Zero Trust is based on the principle that no entity is trusted by default. This applies to users, devices, applications, and data packets, regardless of whether they are located within or outside the network perimeter [4]. This approach assumes that any entity could be compromised and, therefore, requires continuous verification. Trust is established not based on location or network boundaries, but on dynamic factors such as identity, context, and behavior [19].

Due to the complexity of 5G, 6G, and future networks, the zero trust approach is critical in their evolution [5]. Deploying the concept of Zero Trust can avoid a large number of vulnerabilities, since it constantly ensures that an entity is authenticated. Continually upholding authentication is of the utmost importance in networks, and being unable to guarantee this is the source of many vulnerabilities [3]. This concept may contradict the trust boundaries in the DFDs, so it is crucial to clearly distinguish the purpose of these boundaries.

In some threat models, the trust boundary may be seen as an area of the application where there is no need to look for risks. However, this assumption can be dangerous. If trust boundaries are drawn prematurely without first evaluating the risks posed by individual components, there is a significant chance that critical threats will be overlooked. As mentioned earlier, trust should not be granted by default. Instead, each component should be evaluated for its potential impact and likelihood of being compromised. Based on this risk evaluation, trust boundaries can then be defined more accurately, ensuring that appropriate security controls are applied where needed. Moreover, as systems evolve during development, trust boundaries must be continuously reviewed and updated to reflect architectural or operational changes; otherwise, previously unconsidered risks may emerge unnoticed.

2.5 OWASP Risk Rating Methodology

Once vulnerabilities have been detected in a system, various methods of calculating the risk of a specific vulnerability can be used to prioritize the vulnerabilities with the highest risk rating. This is very important from a business perspective to maximize the efficiency of resources spent on security. Instead of focusing on minor risks with no to low business impact, focusing on risks with a higher technical or business impact is more efficient.

The OWASP Risk Rating [26] is an approach to risk analysis. The first step is identifying a threat and performing a risk analysis. This could be a threat identified by using STRIDE in the previous step. Some important factors are necessary to be known to get an accurate rating. It is, however, first essential to understand what a risk is. The most standard description of a risk is described by the formula 2.1, where the likelihood and impact together make up the risk factor.

$$Risk = Likelihood \cdot Impact \quad (2.1)$$

To estimate the likelihood of the risk, two important factors must be considered: threat agent factors and vulnerability factors. Consequently, each of these factors has four "options" or subcategories, each rated from zero to nine.

Note that there may be multiple threat agents that can exploit a particular vulnerability, so it's usually best to use the worst-case scenario.

The first option when determining the threat agent factors is *skill level*, which refers to how technically skilled the group of threat agents is. Secondly, the *motive* of the group is considered. Thirdly, the *opportunity* for the threat agents to find and exploit the vulnerability is rated. Finally, the *size* of the threat agent is determined. For a particular vulnerability, there may also be several threat agents. If that is the case, the worst-case scenario of those threat agents should be used. For example, in the case of a nuclear power plant's control system, a state-sponsored actor is far more likely to possess the necessary resources, expertise, and motivation to breach highly secured industrial systems compared to hackers.

The vulnerability factors refer to the specific vulnerability and how likely it is to be discovered and exploited. The first consideration is the *ease of discovery*, which estimates how easy the vulnerability is to discover. The second factor *ease of exploit* is thus an estimation of how easy it is for the threat agents to exploit the vulnerability. Thirdly, the *awareness* refers to how well-known the vulnerability is to the threat agents. Finally, *intrusion detection* estimates how likely an exploit is to be detected.

When estimating the impact of the risk, there are similarly two important factors to consider with their respective options. The first factor is the *technical impact* on the application, which estimates the impact if the vulnerability is exploited. The options are loss of confidentiality, loss of integrity, loss of availability, and loss of accountability. This considers the traditional CIA triad with the addition of accountability, which refers to how traceable the threat agents' actions are to an individual [11].

Business impact factors can be more difficult to measure compared to other factors and may heavily vary depending on the business. As mentioned earlier, the difficulties with an asset-centric threat model are identifying the actual assets and their value. This business impact factor is the measurement of the value the asset has. This can include different options depending on the business; however, the most common options are *financial damage*, *reputation damage*, *non-compliance*, and *privacy violation*.

Once all options have been rated, the average can be calculated to produce the overall likelihood, technical impact, and business impact. The average of threat agent factors and vulnerability factors produces the overall likelihood, whereas the average of the technical and business impacts is calculated separately. The impact is either chosen as the technical or business impact, where OWASP always recommends applying the business impact if it is available. The average is calculated and categorized into the different severity levels, low, medium, and high, based on values from table 2.3.

Likelihood and Impact Levels	Level
0 to < 3	LOW
3 to < 6	MEDIUM
6 to 9	HIGH

Table 2.3: Likelihood and impact level categorization

At this stage, the options have been given a value, and the average has been calculated to provide the factors with their respective severity level. When it has been decided to use either the technical or business impact, the risk can be calculated using the risk severity matrix 2.4, which adheres to calculating the risk as in 2.1.

If a vulnerability, for example, has a low likelihood but a high technical impact, the total risk would be of medium severity. Analyzing risks this way makes it simple to prioritize risks by their severity level.

HIGH	Medium	High	Critical
MEDIUM	Low	Medium	High
LOW	N/A	Low	Medium
Impact * Likelihood	LOW	MEDIUM	HIGH

Table 2.4: Risk Matrix

2.6 Risk Management Strategies

Once risks have been identified and assessed, the next step is determining how to respond to them. In risk management practice, there are four primary risk response strategies: avoid, address, accept, or transfer [21].

2.6.1 Avoiding Risks

Risk avoidance is a proactive strategy designed to prevent a risk from occurring in the first place. For example, an organization might avoid implementing a third-party technology due to critical risks or avoid storing sensitive data in its database.

2.6.2 Mitigating Risks

The most common approach to risk is to address it through design or operational processes, typically by adding features that reduce potential risks. Shostack [21] categorizes mitigation techniques as standard, platform-provided, developer-implemented, operational, and custom. All these methods aim to reduce the likelihood or impact of a risk. Designing custom mitigation techniques is generally not advised since it may have high risks of being taken advantage of and is also expensive to thoroughly test. While implementing standard mitigation techniques, such as those outlined in the ISO/IEC 27000 series [10] often provides an effective baseline for managing risk, this approach is not universally sufficient. In specialized and complex sectors, such as the telecommunications industry, custom mitigation strategies are frequently necessary to address unique security challenges.

2.6.3 Accepting Risks

Accepting a risk is usually done either via business acceptance or user acceptance.

Business acceptance occurs when an organization consciously chooses to tolerate a risk for strategic or operational reasons. For internal software, this might mean exposing certain data or bypassing security controls when the benefits outweigh the potential impact. However, these decisions must also account for external factors like privacy regulations and legal obligations. Furthermore, if the software operates in a specialized domain, such as healthcare or critical infrastructure, its security must meet the appropriate industry standards.

User acceptance applies when the user is in the best position to decide whether to accept a risk. Developers may need to warn users about such risks in a way that is Necessary, Explanatory, Actionable, and Tested (NEAT) [scho] to support informed decision making. An example of this is when a user attempts to connect to a website with an expired or invalid certificate. In this case, the browser issues a warning about the potential dangers yet still allows the user to choose whether to proceed or not.

2.6.4 Transferring Risks

Risk transference is a risk management strategy in which the potential financial or operational impact of a risk is shifted to a third party. This is commonly achieved through contractual agreements or insurance policies, which legally bind the third party to assume liability. A practical example in software development is the delegation of authentication services to a

specialized provider like Auth0. By doing so, the developing organization transfers the security obligations related to managing user credentials and session data. This arrangement ensures that the third-party vendor is contractually accountable for any security vulnerabilities or breaches within the scope of their service.

2.7 Related Work

Sheik et al. [20] propose a hybrid threat modeling approach for Cyber Physical Systems. Their methodology for threat modeling initially follows the STRIDE methodology for finding threats from a DFD. However, they calculate the risk based on the DREAD model once the threats have been identified from STRIDE. Based on the threats evaluated, it is possible to prioritize mitigations on severe risks, from which they suggest creating both an attack tree and a defensive attack tree to understand how the threat is realized.

Xiong and Lagerström [27] conduct a systematic literature review in the field of threat modeling to identify the state-of-the-art methodologies. From their research, 54 articles were reviewed and analyzed. Their findings show that the concept of threat modeling lacks common ground. This is supported by the fact that many of the articles use many different methodologies. However, in terms of state-of-the-art methodologies, the article does identify that a majority of the methods used in threat modeling are manual, and due to the inefficiency of the manual methodologies, there is a natural drive toward developing automated threat modeling processes. The article also suggests more empirical validation methods, integration with threats and vulnerabilities, and better commercial threat modeling tools.

Lohmann et al. [13] expand on Xiong and Lagerström's [27] systematic literature review by conducting a similar review. Similarly, many articles reviewed use different methodologies to perform threat modeling. However, key elements were found that generalize the threat modeling process based on the reviewed articles. Typically, threat modeling is performed during the design phase, where DFDs are used to describe the architecture. An attacker's possible interactions with the architecture are considered potential threats. This can be enhanced by using threat libraries and also by considering security controls that may mitigate an identified risk. The threat modeling process was generalized into four steps. The first step is to gain an understanding of the system and assets. The second step is threat identification. The third step is threat analysis, which concerns calculating the impact. The final step is defining countermeasures.

Furthermore, the most commonly proposed future research was identified as the application of the presented method or tool for the purpose of validation or extension. This is similar to the conclusion by Xiong and Lagerström [27], where they suggested more empirical and fewer examples. Lohmann et al. [13] also propose to involve mitigation controls and their respective risks in the threat modeling process.

In the telecom industry, 5G and beyond continue revolutionizing how society interacts with technology. While these advancements offer unprecedented connectivity and new opportunities, they also introduce a vastly expanded threat landscape that demands robust and adaptive security measures. To address these challenges, security in 5G must be built on three key points: flexible security mechanisms, supreme built-in security, and security automation [17]. Ahmad et al. [1] further argue that because 5G integrates a wide range of technologies such as SDN, NFV, and massive IoT, this diversity necessitates equally dynamic and automated security systems capable of adapting to evolving threats and security controls. This perspective reinforces the importance of integrating threat modeling and risk assessment into the SDLC. By embedding security into the development process, vulnerabilities can be identified and remediated before the software is released into a live environment, reducing the

cost and impact of post-release fixes. Additionally, updates to threat intelligence, vulnerabilities, or security controls can be rapidly incorporated during development, enabling teams to patch systems efficiently and maintain a strong security posture throughout the lifecycle.

2.7.1 Threat modeling in the SDLC

Cruzes et al. [22] focus on the challenges and experiences from implementing the Microsoft threat modeling technique with STRIDE in agile development projects. To identify these issues, a case study was conducted from which 21 challenges were identified. The challenges were mapped to challenges discovered by other authors to establish an overview of the most common challenges in this area. Some examples of the challenges identified are keeping the DFD up to date, limited threat catalogs, slow speed to derive threats, and keeping the threat list up to date.

In general, Cruzes et al. [22] discuss the specific challenges with using the Microsoft threat modeling technique, where threat modeling sessions are held with many members of the team and possibly stakeholders. However, the authors do note that there is a need for highly automated threat modeling techniques, which can help address several of the challenges discussed in the article. One of the key advantages of automation is that it brings structure and clarity to the threat modeling process, making sessions more consistent, repeatable, and easier to facilitate. A clearer vision of how sessions are run also improves team alignment, reduces misunderstandings about roles and goals, and enables more focused identification and documentation of threats. This ultimately leads to more actionable outcomes and a smoother integration of security practices into the development workflow.

Bernsmed et al. [2] note that threat modeling has not been widely adopted in agile software development. Security practices are generally more challenging to adopt in agile software development due to a variety of challenges. In addition, studies in software security are usually quite general, with few studies that focus on specific security practices. Nevertheless, Bernsmed et al. sought insight into how threat modeling can be adopted to suit agile software development projects. Data was collected by conducting four studies, one of which is the study by Cruzes et al. [22]. The other studies were based on interviews, surveys, and document analysis. Based on these studies' findings, challenges and best practices were identified.

A common challenge of the studies when drawing DFDs was a lack of motivation. In this context, they had found that developers perform the threat modeling and thus the drawing of the DFDs. The concerns on this topic were that this process is very time-consuming and tedious, which hurts the productivity of the developers. Another issue is the level of abstraction when drawing DFDs and producing accurate models. Since the DFDs are not linked to any actual software, it may be difficult to assess how accurately they represent the software.

However, other participants argue that DFDs can be seen as a good way of documenting the system and can even be a valuable tool for onboarding new employees or presenting it to customers. Another set of challenges was identified during the analysis of applying STRIDE. Some participants argued that the threats generated by STRIDE are too vague and that it was a waste of time. However, other participants argued that STRIDE was helpful in generating an initial set of threats that could be expanded upon [2]. The studies did not explicitly investigate whether STRIDE contributed new insights that developers were previously unaware of. However, the fact that some participants found it valuable as a way to structure early threat identification suggests that it supported a more organized approach to exploring potential risks.

Additionally, Bernsmed et al. [2] continue their discussion by identifying the results useful for teams working in agile software development. Threat modeling requires better solutions for integration into the SDLC. Bernsmed et al. propose automating the generation of threats and integrating the process with Jira, enabling the creation of tickets to support the systematic

management and resolution of identified security issues. This makes it easier to track specific countermeasures taken and is similar to what Shostack [21] describes as security bugs. Furthermore, there is a need for better tooling compared to the tools currently available. This tool would implement the aforementioned suggestions and integrate them into the continuous delivery. Finally, Bernsmed et al. propose some recommendations for practice and future research that concern further development of threat modeling tools, empirical studies, development of templates for DFDs, and also evaluating the effectiveness of threat modeling in the SDLC.

Theurich et al. [23] also conducted a systematic literature review to gain insights into the state-of-the-art threat modeling challenges and methodologies. The authors present nine challenges and six practices in threat modeling. Similar challenges are identified from the aforementioned papers, such as a lack of motivation when conducting threat modeling, and that it is a very expensive process. Specifically, the structure of meetings, who should participate in the meetings, and maintainability are concerns that are raised. This further strengthens the view that there are many challenges with threat modeling in its current state.

The authors continue the discussion by mentioning different practices in place that address some of the challenges. Regarding DFDs, it is suggested to enhance the DFD notation with security information such as security mechanisms, threat catalogues, and integrated security solutions. One specific practice mentioned is tool support. Examples of this are Microsoft Threat Modeling Tool (MSTM), OWASP Threat Dragon, and Waypoint, which follow frameworks such as STRIDE, VAST, or PASTA. While there have been some attempts at automating design flaws from a DFD [24], which generated low precision compared to manual threat modeling, but with a promising result of recall.

A drawback of analyzing DFDs is that it provides limited support for cyber-physical systems, but it is suitable for software applications. When Theurich et al. map the challenges against the practices, it is evident that tool support for threat modeling is the only practice that fully addresses the fact that threat modeling is expensive. The automated approach is also useful in the sense that the results become repeatable, concrete, and improve documentation if used iteratively in the SDLC. This solves issues as the lack of motivation since the tedious effort becomes more efficient and faster, and also improves visibility towards other stakeholders, other than the developers.

2.7.2 Automatic threat modeling approaches

Due to the expensive process of threat modeling, different automated threat modeling techniques have been established. Granata and Rak [8] conduct a systematic literature review regarding these techniques, focusing on open-source tools. Similar to other reviews, STRIDE is identified as the most common threat classification method. STIDE is a useful technique since it can describe threats into classes. These classes make it easier to generate and analyze the threat model due to the abstraction. The authors also mention that the LINDUNN method can be used in addition to STRIDE. This methodology, however, focuses on privacy issues.

Another approach is using the traditional CIA triad to understand how a threat may compromise the confidentiality, integrity, or availability of an asset or security requirement. Another variation is using different classification methods, such as CAPEC, a knowledge base that has classification on attack patterns. This is very different from the STRIDE approach since it focuses on techniques used by an adversary to exploit known weaknesses and not the weaknesses themselves. The approach of using a knowledge base is similar to the STRIDE approach, as it depends on threat patterns and rules. Some authors claim that there are additional threats that should be derived by using propagation rules that can follow the path of a number of interactions as opposed to analyzing a single interaction. This method is called

attack-path, which can create graphs of the path of an attacker, but it is seemingly more complex than the classification technique.

After setting some criteria on the open-source threat modeling tools Granata and Rak [8] present eight tools. Most of these tools were based on DFDs and had relationship-based threat selection methodology. Interestingly, only three of the tools supported risk analysis and only in some cases suggest appropriate countermeasures. Only one of these tools that supports risk analysis is evaluated in the paper, which is SLA-generator. This tool uses the graph-based database Neo4j and organizes threats based on their asset type. The tool can suggest security controls to be applied from the NIST SP-800-53 [16].

PyTM is another tool that Granata and Rak analyze that implements threat modeling as Python code. The PyTM threat database is built upon data from MITRE and CAPEC, which are manually translated with a set of conditions that must be fulfilled to be generated. This can be done by checking the target value of the threat and checking what security controls are currently implemented. The tool presents an innovative option: adding security controls to negate threats. By applying these security controls, which are countermeasures, it is possible to mitigate some of the threats. This is an interesting approach where users are granted configuration possibilities to adapt their threat model for their own cases.

Furthermore, the Microsoft threat modeling tool is another tool that automatically generates threats. The tool uses DFDs where each node has a component type, which is represented by the possible components in a DFD, and a component value, which is additional information to describe some functionality of the specific component. Additionally, different types of interactions can be selected which has in turn a different set of configurable attributes.

OWASP Threat Dragon is another threat modeling tool. This is based on rule engines to create threats and mitigation. Similar to the Microsoft threat modeling tool, the OWASP Threat Dragon has configurable attributes for the different component types. A data flow component can, for example, have three different parameters: "protocol", "is encrypted", and "is over a public network". Based on the component type and parameters, threats are generated and then prompt the user to evaluate them concerning priority level and threat status. It is also able to suggest possible mitigation for the identified threats. Granata and Rak continue to compare these four tools, which showed that they, for the most part, have a similar number of threats when analyzing the same system. However, OWASP Threat Dragon was an exception and had a much lower amount of threats, also with a lower threat granularity. OWASP Threat Dragon identified 30 threats as opposed to 84-97 compared to the other three tools.

Finally, the authors do note that risk analysis is built on top of threat modeling to generate risks based on the threat likelihood and impact. For future research, Granata and Rak will look into tools that implement risk analysis methodologies. But it does show an interest in combining the threat modeling process with risk analysis.

Tuma et al. [24] explore the automation of security design flaw detection in software architectures based on DFDs. By performing a study of several design models, security experts scrutinized the model to identify five different design flaws: authentication bypass, insufficient crypto key management, insecure data storage, insecure data exposure, and insufficient auditing. To automate the detection of flaws, the DFD models were extended with annotations, including information on data types (e.g., encrypted or plaintext) and security solutions (e.g., authentication mechanisms, access control points). This information is fed into the model to understand how data flows through the system and where protective security controls are located.

To enable the automatic design flaw detection the authors implement a rule-based engine where each rule corresponds to a specific design flaw to check for the presence or absence of correct security measures or handling of the data. For example, absence of encryption on sensitive data will trigger a rule for insecure data exposure. Furthermore, these rules are

linked to STRIDE categories in order to associate security solutions with the threats they are intended to mitigate.

In order to evaluate the system, Tuma et al. compared the results of the rule-based automatic detection of design flaws against manual assessments conducted by security experts in workshops together with the user who had modeled the specific DFD. The results were measured using precision and recall as performance metrics by analyzing the true and false positives between the automatic detection and security experts. The result showed challenges in reducing false positives, which affected the precision of the tool. To conclude, the authors mention that the results suggest that although the automated method cannot match the performance of expert analysis, it can still serve as a complementary tool to improve the effectiveness of security experts.



3 Method

The following chapter describes the methodology used to develop and evaluate the semi-automated tool which is compared to existing tools in the field of threat modeling and risk analysis. The chapter is organized into four sections. Firstly, an overview of the functionality in the developed tool Thalaura is given. Secondly, the mappings and method to identify vulnerabilities is explained along with the data created to test the application. Consequently the third chapter covers the security controls that were added to the data set. Finally, a test scenario is set up to work as input for the creating of threat models for the different tools.

3.1 Overview of Thalaura

To provide a high-level overview of Thalaura, key data types and descriptions of important functions are presented under this section.

3.1.1 Vulnerabilities and threat vectors

In Thalaura, a vulnerability is defined by a title and associated mitigations, which represent the security controls linked to it. Additionally, multiple threat vectors can be configured, which are the methods through which a vulnerability can be exploited to trigger a threat. A threat vector is defined by a description of the exploitation, the STRIDE category, ease of exploit, and ease of discovery.

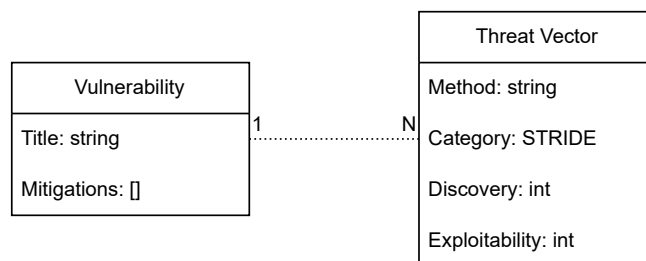


Figure 3.1: Vulnerability and threat vector data types

3.1.2 Security controls

Mitigations against vulnerabilities are referred to as security controls, which are countermeasures that are possible to implement by developers. A security control is a dynamic structure composed of checkboxes and select components, collectively referred to as security control properties. Each security control is associated with a specific component type, indicating which component in the DFD it applies to. Both a security control and its individual properties can be linked to multiple vulnerabilities.

Additionally, a security control property can reference its own data type, creating a composite relationship. This design allows for greater flexibility when configuring a security control and enhances clarity by enabling conditional visibility. For instance, properties linked to another property can remain hidden by default and become visible when a corresponding checkbox is selected.

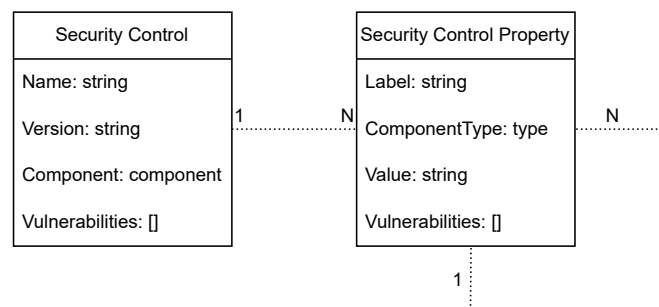


Figure 3.2: Security controls and security control properties data types

3.1.3 Risk generation

Risks are generated when a component is updated by either adding an asset or applying a new security control for the component. When an update is prompted Algorithm 1 will take the assets and security controls into account and identify which security properties of an asset is compromised by the relevant threat vectors. The risks generated is targeted against the confidentiality, integrity and availability of the asset. The risk is based on what security properties are set to be protected in the asset and what security properties the threat vector can compromise. The intersection of the security properties in the asset and threat vector is used to generate the risk which is described as compromised properties.

Algorithm 1 Risk generation algorithm

```

componentId ← selectedComponent
assets ← selectAssets(componentId)
vulnerabilities ← selectVulnerabilities(componentId)
for asset in assets do
  for vulnerability in vulnerabilities do
    for threatVector in vulnerability do
      if threatVector.properties matches any asset.properties then
        generateRisk(compromisedProperties)
      end if
    end for
  end for
end for
  
```

3.1.4 Asset configuration

To adopt the asset-centric threat modeling approach using OWASP Risk Rating, the definition of assets is crucial. The important information to gather about an asset is the business impact factors and the technical impact factors against the security properties' confidentiality, integrity, and availability. The OWASP Risk Rating methodology also captures an accountability property. However, this is disregarded. This decision was made because the traditional CIA triad is a widely recognized framework. Furthermore, since STRIDE addresses repudiation, which is a concept closely related to accountability, the omission of accountability is reasonable.

The resulting technical impact factors are the severity of loss of confidentiality, loss of integrity, and loss of availability. When creating an asset, the user is required to fill out which security properties have an impact if compromised. For each security property, there are six options ranging from insignificant or N/A, which has a value of 0, to a maximum value of 9. All of the impact descriptions and values can be seen in Table 3.1. These descriptions and values were adapted from the OWASP Risk Rating documentation as in [26], with the addition of an option for the value of 0 to account for cases where a particular security property is irrelevant or negligible for the given assets.

Similarly, the business impact factors also have a range of options. These options concern privacy violation, non-compliance, financial damage, and reputation damage. All of the business impact factors can be seen in Table 3.2. Finally, to enhance visibility, the user must assign a threat impact to each security property of the asset. The threat impact serves as a justification for the technical impact factor value and includes a description of the worst-case scenario if the property is compromised. When the technical and business impact factors have been configured, the impact of the risk can already be calculated, which gives the first variable for the Equation 2.1.

Property	Impact Description	Value
Confidentiality		
	Insignificant or N/A	0
	Minimal non-sensitive data disclosed	2
	Minimal critical data disclosed	6
	Extensive non-sensitive data disclosed	6
	Extensive critical data disclosed	7
	All data disclosed	9
Integrity		
	Insignificant or N/A	0
	Minimal slightly corrupt data	1
	Minimal seriously corrupt data	3
	Extensive slightly corrupt data	5
	Extensive seriously corrupt data	7
	All data totally corrupt	9
Availability		
	Insignificant or N/A	0
	Minimal secondary services interrupted	1
	Minimal primary services interrupted	5
	Extensive secondary services interrupted	5
	Extensive primary services interrupted	7
	All services completely lost	9

Table 3.1: Technical impact descriptions and values from the OWASP Risk Rating documentation [26]

Category	Impact Description	Value
Privacy Violation		
	N/A	0
	One individual	3
	Hundreds of people	5
	Thousands of people	7
	Millions of people	9
Non-Compliance		
	N/A	0
	Minor violation	2
	Clear violation	5
	High profile violation	7
Financial Damage		
	N/A	0
	Less than the cost to fix the vulnerability	1
	Minor effect on annual profit	3
	Significant effect on annual profit	7
	Bankruptcy	9
Reputation Damage		
	N/A	0
	Minimal damage	1
	Loss of major accounts	4
	Loss of goodwill	5
	Brand damage	9

Table 3.2: Business impact values for Privacy Violation, Non-Compliance, Financial Damage, and Reputation Damage

3.1.5 Data Flow Diagram

To develop Thalaura the react-flow library was utilized to enable the user to be able to draw the DFD. Components exist in the form of nodes and edges. External entities, processes, data storages and trust boundaries are represented by nodes, whereas the dataflows are represented by edges that link the nodes. On each component, it is possible to select which assets reside inside it. For example, selecting an asset in a dataflow suggests that some sensitive asset is sent between two components. Similarly, a list of security controls can be added with the purpose of mitigating risks in that specific component. The selected security controls can be configured according to the security control properties by the user. Furthermore, a trust boundary has a set of three different options concerning the trust zone. These options are protected, internal, and exposed, which have respective values 1, 5, and 9. These options are used when calculating the risks but have no effect on how an actual risk is generated. Considering the Zero Trust principle, it is important not to remove any risks due to a zone being considered "safe". However, it is possible to assume that an on-prem environment or protected zone has a smaller chance of being compromised in comparison to an open network.

The canvas of the DFD and the accompanying configuration window when editing a node or edge can be seen in Figure 3.3.

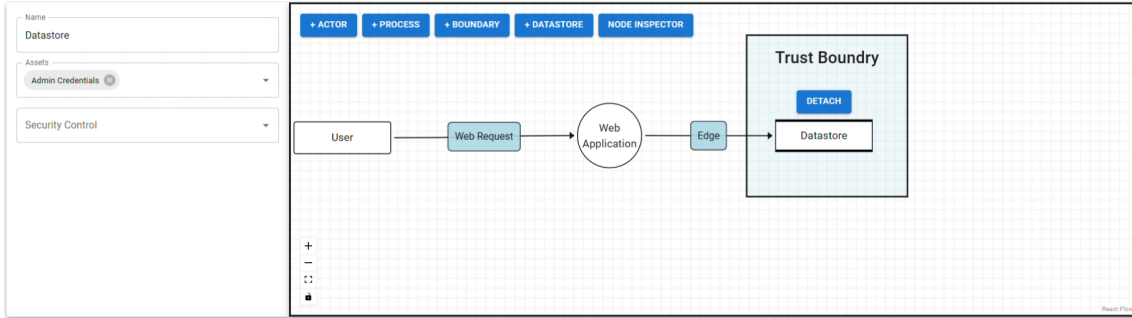


Figure 3.3: Configuration window and DFD canvas

3.1.6 Calculating the risk rating

Once a risk is generated through Algorithm 1, we have access to all variables that concern it. As mentioned previously, the OWASP Risk Rating methodology is used to calculate the risk rating. However, to leverage the method to be used with the risk generation, some adjustments had to be made. Since the risk generation algorithm identifies what security properties of an asset can be compromised, we can adjust the technical impact to only take into account the security properties that are compromised. This is done by setting the technical impact properties that are not compromised to 0.

Then, the impact factor can be calculated by taking the average of all technical impact factors associated with the asset. By design, the denominator in this calculation is fixed at three, which ensures that compromising additional security properties increases the overall impact score rather than diluting it. This approach prevents a risk that only affects confidentiality from being rated equally to a risk that compromises confidentiality, integrity, and availability. In this model, a high severity risk that affects only one property should still score lower than a moderate risk that compromises all three. The technical impact is thus calculated by:

$$T_i = \frac{C + I + A}{3}$$

subject to the constraints:

$$C, I, A \in \{0, 9\} \quad \text{and} \quad C + I + A > 0$$

Finally, when the impact factor has been decided, the probability factor can be calculated. All options are retrieved from the sources seen in Table 3.3, and then the average is calculated, resulting in the probability factor. Then, the risk rating can be calculated by looking at the risk matrix.

Note that in this model, the values for skill level, motive, and size have been set statically to 9. This decision reflects the high baseline threat level typically associated with telecom systems, where potential attackers such as state-sponsored actors are assumed to possess strong capabilities, clear incentives, and significant resources. While these values should ideally be adjusted dynamically, enabling such flexibility was considered out of scope for this work. Instead, fixed values were used to simplify the model and focus on other aspects of the threat and risk evaluation.

Factor	Option	Source of Value
Threat Agent Factors		
	Skill level	Always 9
	Motive	Always 9
	Opportunity	Decided by zone value
	Size	Always 9
Vulnerability Factors		
	Ease of Discovery	Decided by threat vector
	Ease of Exploit	Decided by threat vector
	Awareness	Decided by zone value
	Intrusion Detection	Decided by zone value

Table 3.3: Description of probability factors

3.1.7 Overview of risk rating

To further clarify the relationship between the parameters, a tree of the risk calculation is shown in Figure 3.4. Note that either the technical impact or the business impact is chosen when following this methodology. For this thesis, the technical impact is analyzed since no good business impact information was available.

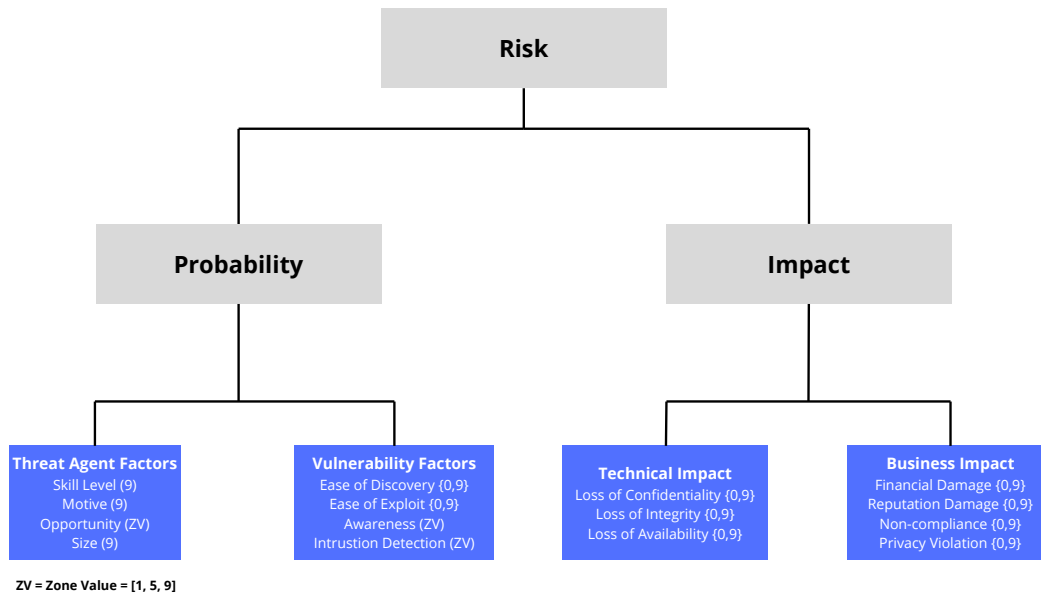


Figure 3.4: Risk calculation tree for the OWASP Risk Rating Methodology

3.2 Identifying vulnerabilities with STRIDE

The goal of the threat modeling for this tool is to identify risks that may compromise the security properties of any assets. The threat impacts against these properties have already been defined when the asset is configured. However, the threat is realized by vulnerabilities in the system, which are exploited by a threat vector. By categorizing the vulnerabilities by STRIDE, it is possible to understand which security properties are threatened. The mapping between STRIDE and the security properties is shown in Table 3.4. This approach is similar to the mapping proposed by Honkaranta et al. [9] with the addition that tampering and elevation of privilege affect the availability of an asset.

Threat Type	Confidentiality	Integrity	Availability
Spoofing	x	x	
Tampering		x	x
Repudiation		x	
Information disclosure	x		
Denial of Service			x
Elevation of Privilege	x	x	x

Table 3.4: STRIDE mapping to security properties

An asset is any form of data that requires protection. Data can exist in three different states: data at rest, data in use, and data in transit. Thus, the assets that are to be protected may also be in these three states. This is an important distinction because the asset must be protected in all three states. There are, however, different countermeasures depending on what state the asset is in. These states can also be linked to components in the DFD. Assets in transit are linked to the dataflows, assets in use are linked to processes where they are stored in memory, and assets at rest are linked to datastores. By this definition, it is now possible to use STRIDE to reflect what threats exist against the assets and how they may be used to compromise the security properties of the asset. We can identify threats against an asset by iterating over the STRIDE mnemonic. According to Shostack, you are doing reasonably well if you can identify a threat per checkbox from a defined table. This can be improved by considering mitigations against those threats as well. [21] Table 3.5 takes the state of assets into consideration, which is similar to Table 2.2, which refers to the original STRIDE-per-element table with a change where spoofing threats are derived from dataflows instead of processes. This shows a difference in the line of thinking when discussing assets instead of system vulnerabilities. There is clearly a threat against an asset in transit when the sender or the target destination is spoofed by an adversary. In reality, it is not the dataflow that is being spoofed, but the threat concerns the transportation of the asset.

Asset State	S	T	R	I	D	E
Asset in transit	x	x	x	x	x	
Asset in use		x	x	x	x	x
Asset at rest		x	x	x	x	

Table 3.5: STRIDE threats against asset states

3.2.1 Inherit vulnerabilities against assets in transit

An asset in transit is at risk of spoofing from either the source or the target destination. For example, it may also be exposed to tampering through a man-in-the-middle attack. Furthermore, the target destination may later deny having received the asset, which represents a repudiation threat if there is no logging or evidence to confirm the transaction. An asset in transit is also vulnerable to uncontrolled resource consumption, where an adversary may perform a flooding attack to disrupt the target component. An elevation of privilege threat is slightly more difficult to define in the context of how it may affect the asset in transit; however, if the communication channel is overtaken by session hijacking, an adversary may compromise multiple properties of the asset. Six threats were easily identified, and assets in transit look very vulnerable.

Vulnerability	Threat Vector	Mitigation	STRIDE	E	D
Missing Authentication	Bypassing authentication mechanisms	TLS 1.3 Authentication	Spoofing	9	9
Unsecured Communication Channel	Man in the middle attack	TLS 1.3	Tampering	5	9
	Data flow sniffing	TLS 1.3	Information Disclosure	9	9
Uncontrolled Resource Consumption	Flooding attack	Rate Limiting	Denial of Service	9	9
Insufficient Logging	Relying on the lack of monitoring and timely response to achieve their goals without being detected	Logging	Repudiation	5	7

Table 3.6: Vulnerabilities against an asset in transit

3.2.2 Inherit vulnerabilities against assets in use

The inherent vulnerabilities against assets in use are a distinct class of vulnerabilities that arise when sensitive data is actively being processed or modified by a system component. During execution, this data resides temporarily in memory, making it susceptible to threats and vulnerabilities that would not apply to data at rest or in transit.

Vulnerability	Threat Vector	Mitigation	STRIDE	E	D
Missing Input Validation	Code injection	Input validation	Tampering	9	9
Insufficient Logging	Relying on the lack of monitoring and timely response to achieve their goals without being detected	Logging	Repudiation	5	3
Cleartext Storage of Sensitive Information in Memory	Reading memory	mlock()	Information Disclosure	3	3
Uncontrolled Resource Consumption	Forcing system to run out of memory	cgroups	Denial of Service	9	9
Incorrect Privilege Assignment	Accessing restricted functionality or sensitive information	SELinux	Elevation of Privilege	3	3

Table 3.7: Vulnerabilities against an asset in use

3.2.3 Inherit vulnerabilities against assets at rest

To understand the threats against an asset at rest, it is possible to look at the datastore component in the DFD. A datastore may contain assets which are at rest. According to the STRIDE-per-element approach there are four prelevant STRIDE threats against datastores. These are tampering, repudiation, information disclosure and denial of service. Consequently, this can help us understand the threats against the asset that is stored in a datastore component.

Vulnerability	Threat Vector	Mitigation	STRIDE	E	D
Missing Authorization	Bypassing authorization mechanics	Role Based Access Control	Elevation of Privilege	5	7
Insufficient Logging	Relying on the lack of monitoring and timely response to achieve their goals without being detected	Logging	Repudiation	5	3
Cleartext Storage of Sensitive Information	Reading sensitive data in plaintext	Encryption	Information Disclosure	5	7
Missing Input Validation	Injecting malformed or invalid data into the database.	Input validation	Tampering	9	9
Allocation of Resources Without Limits	Expensive query flooding	Resource Limitation	Denial of Service	5	7

Table 3.8: Vulnerabilities against asset at rest

3.2.4 Residual vulnerabilities from security controls

In addition to the inherent vulnerabilities against components, residual vulnerabilities were also added against security controls and security control properties, which can be seen in Table 3.9.

Vulnerability	Threat Vector	Source	STRIDE	E	D
Input validation does not normalize input	Encoded Injection	Input Validation	Tampering	5	3
Denylist should not be relied upon as the primary method	bypassing filter	List Validation	Tampering	5	7
TLS does not provide inherent replay protections for 0-RTT data	mounting a replay attack by simply duplicating a flight of 0-RTT data	TLS	Spoofing	1	1
Undefined Certificate Validity	adversary gains access to a certificate and uses it to impersonate a trusted client or server	mTLS	Spoofing	1	1
Too short certificate validity period	Exploiting old certificates	Certificate validity period	Tampering	3	3
Passwords are not encrypted with hash function	Obtaining encryption key and decrypting all passwords	Password storage	Information Disclosure	5	3
Improper Authorization	Bypassing authorization mechanics	Role Based Access Control	Elevation of Privilege	3	3
Client Impersonation	Exploiting the lack of cryptographic client authentication by connecting over TLS	mTLS	Elevation of Privilege	1	3

Table 3.9: Vulnerabilities against security controls and their properties

3.3 Security Controls

Creating a comprehensive catalog of vulnerabilities and security controls is a comprehensive and challenging task. A common approach used by other tools involves importing information from external catalogs and mapping it to system-specific parameters. However, this process is not straightforward, as there are numerous external catalogs describing threats and vulnerabilities using different terminologies and classification schemes.

One example of a catalog is the Common Weakness Enumeration (CWE), a community-developed list that includes a hierarchical taxonomy and detailed descriptions of software weaknesses. In CWE, a weakness is defined as a condition that may lead to the introduction of vulnerabilities.

Another valuable resource is the OWASP Proactive Controls, which aim to mitigate the most critical web application security risks, including those identified in the OWASP Top 10 [6]. By adopting the techniques outlined in this list, developers can significantly reduce the likelihood of introducing security vulnerabilities.

This project focuses on three of the most impactful security controls from the OWASP list:

- **C1: Implement Access Control**
- **C2: Use Cryptography to Protect Data**
- **C3: Validate All Input and Handle Exceptions**

Due to the complexity of modeling numerous types of security controls, only these three were implemented in the tool. In total, eight additional vulnerabilities and threat vectors were introduced based on these controls. An example of how the security controls are configured can be seen in Figure 3.5

The screenshot shows a web interface for configuring security controls. On the left, a sidebar lists various controls: TLS 1.3 (selected), Ciphers Suite, My Secret Encryption Method, mTLS, Certificate validity period, and IPSec 2.0. The main panel displays the configuration for 'TLS 1.3'. It includes a 'Name' field with the value 'TLS', a 'Version' field with '1.3', and a dropdown for 'Applicable component' set to 'Dataflow'. Below these are 'Ciphers Suite' and 'mTLS' sections, each with an 'EDIT' and 'DELETE' button. At the bottom, there is a 'Certificate validity period' dropdown set to '398 days' and an 'ADD PROPERTY' button.

Figure 3.5: Example of security control page for TLS 1.3

3.3.1 C1: Implement Access Control

Access control is a fundamental security principle that ensures users and processes only access resources they are authorized to use. There are various types of access control mechanisms, with Role-Based Access Control (RBAC) and SELinux being two notable examples. In this work, RBAC and SELinux were modeled as mitigation techniques for elevation of privilege threats, as shown in Table 3.7 and Table 3.8. However, no specific vulnerabilities were identified that would be mitigated by SELinux within the scope of this model. As a result, only vulnerabilities related to RBAC were included in the dataset.

3.3.2 C2: Use Cryptography to Protect Data

Encryption plays a crucial role in protecting sensitive data both in transit and at rest. To address this, TLS 1.3 was configured to secure dataflows and protect assets in transit. For data at rest, a security control representing encryption was added to the relevant datastores. Additionally, the `mlock()` system call was created as a security control for processes, helping mitigate the risk of cleartext storage of sensitive information in memory.

3.3.3 C3: Validate All Input and Handle Exceptions

To address risks related to unvalidated input, a security control for input validation was introduced for both processes and datastores. This control helps mitigate common vulnerabilities such as injection attacks and improper handling of unexpected input, which can lead to application crashes or unintended behavior.

In addition to these OWASP Proactive Controls, several other security controls were implemented to cover common vulnerabilities across all component types. In total, 15 security controls and 22 associated vulnerabilities were added to the dataset.

3.4 Threat model

To evaluate the application, a test case threat model was created to compare different tools and how they perform. Four different sets of configurations were derived from a DFD. The scenario describes a simple system where users are able to interact with a web service through a web application. The web service handles two databases, a user management database and a user information database. In this scenario, users are able to sign into the application with credentials and then extract their personal information from the user information database. Administrators have direct access to the two database through a database management system (DBMS).

3.4.1 Assets

Three different assets are identified in the DFD, which are admin credentials, user credentials, and user Personal Identifiable Information (PII), which refers to some type of sensitive data. By configuring three different assets, it is possible to describe them with varying sets of severity, which should generate interesting results for the evaluation.

According to the OWASP Risk Rating methodology, business impact factors should be used when available, as they more accurately reflect real consequences. However, in practice, assessing business impact often involves input from stakeholders, contextual business knowledge, and subjective judgments that may not be readily accessible or consistently quantifiable in this research setting.

For this reason, this threat model focuses solely on technical impact factors, which are more straightforward to assess and apply. The technical impact and threat impact configured in Thalaura used the information in Table 3.10 and Table 3.11. In Threat Agile, these assets are configured as data assets. The technical impacts in Threat Agile were configured according to Table 3.12 from which the values were chosen to be similar to the values in Table 3.10 in addition to the quantity which is required for the configuration.

Asset	C	I	A
User Credentials	2	1	1
User PII	6	5	5
Admin Credentials	9	9	9

Table 3.10: Technical impact of assets in Thalaura

Asset	Confidentiality	Integrity	Availability
User Credentials	Minimal sensitive data disclosed	non-sensitive data disrupted	Minimal slightly corrupt data Minimal secondary services interrupted
User PII	Extensive sensitive data disclosed	non-sensitive data disrupted	Extensive slightly corrupt data Minimal primary services interrupted
Admin Credentials	All data disclosed	All data totally corrupt	All services completely lost

Table 3.11: Threat impact of assets in Thalaura

Asset	Quantity	Confidentiality	Integrity	Availability
User Credentials	many	internal	archive	archive
User PII	many	confidential	important	important
Admin Credentials	very-few	strictly-confidential	mission-critical	mission-critical

Table 3.12: Quantity and technical impact of data assets in Threat Agile

3.4.2 Data Flow Diagram

An initial DFD was drawn using draw.io's threat modeling extension. This gives a clear overview of how assets are transported, processed, and stored in the system, which can be seen in Figure 3.6.

Additionally, the threat model was extended to include security controls, which are illustrated in Figure 3.7. It should be noted that this figure is a simplified representation showing only four controls for illustrative purposes. In practice, System B was configured to apply security controls wherever necessary to mitigate the risks identified as originating from System A. The overarching strategy was to implement all possible security controls suggested by the specific threat modeling tools in an attempt to mitigate all generated risks. The specific behavior of these security controls varied depending on the tool used.

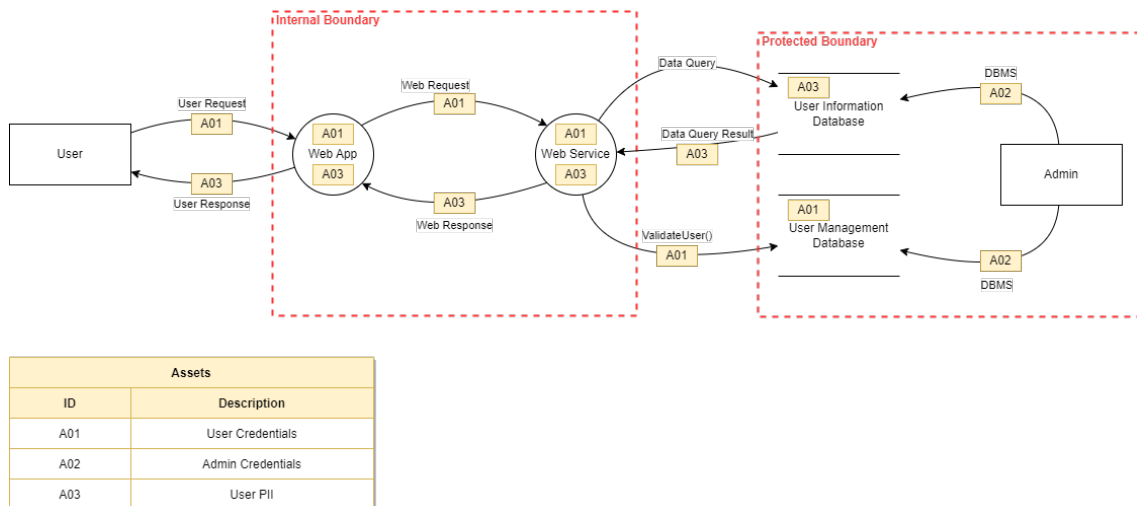


Figure 3.6: Data flow diagram over the threat model

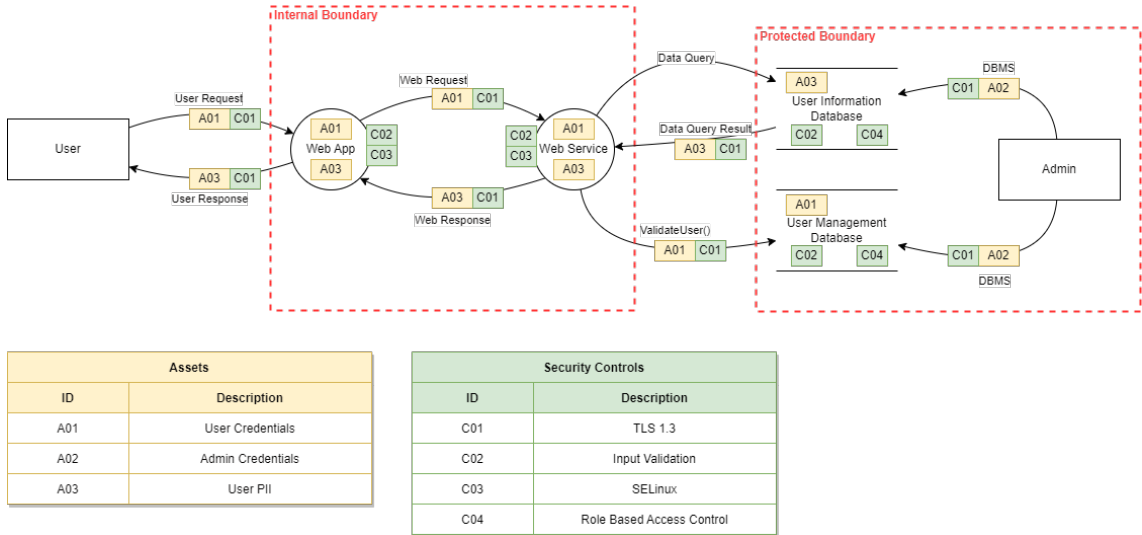


Figure 3.7: Data flow diagram over the threat model with security controls

3.5 Monte Carlo Simulation

A Monte Carlo sensitivity analysis was performed to explore the influence of various input parameters on the resulting risk scores in Thalaura. Instead of relying on a limited set of static scenarios, a script was developed to generate 100,000 simulations with unique parameter combinations. Each simulation was created by randomly assigning values to key input factors, including confidentiality, integrity, availability, STRIDE categories, zone values, discovery values, and exploit values. This process allowed for an exhaustive exploration of the parameter space, providing a robust analysis of how changes across a wide range of plausible configurations affected the overall risk rating.

The goal of using this method is to evaluate the performance of the risk rating approach across a wide range of scenarios. By observing how the results change as the inputs vary, we can gather evidence on the consistency of the methodology, which would be difficult using purely analytic methods [15]. In the simulation, input values were sampled from the ranges and sets shown in Figure 3.4 (e.g., most parameters in the interval 0–9 except for the zone value, which has the discrete set [1, 5, 9]). Since the presented methodology to generate risks uses a deterministic algorithm, the simulation is straightforward.



4 Results

The following chapter presents the evaluation results of the developed semi-automated tool, Thalaura. The chapter is structured into three main sections. First, a qualitative analysis compares the identified risks and generated threat models between Thalaura and existing tools. Secondly, a quantitative evaluation is conducted to analyze the relationships between input parameters and the resulting risk scores, using correlation analysis and severity distributions based on the method implemented in Thalaura.

4.1 Qualitative results

Firstly, the resulting data flow diagrams for Microsoft Threat Modeling Tool, Threat Agile, and Thalaura are shown in Figure 4.1, Figure 4.2, and Figure 4.3, respectively. Although the threat models were generated for both System A (the system without security controls) and System B (the system with security controls), the resulting diagrams are practically identical in their visual representation. The only difference is a small detail in Threat Agile, such as HTTP being replaced by HTTPS in System B. These protocol-level changes, while important from a security perspective, do not alter the overall system architecture shown in the diagrams. Therefore, to avoid redundancy, only the visualization for System B is included in the figures. The minor visual differences stem from the fact that configuration-specific details are managed within the component configuration windows rather than reflected in the diagrams themselves.

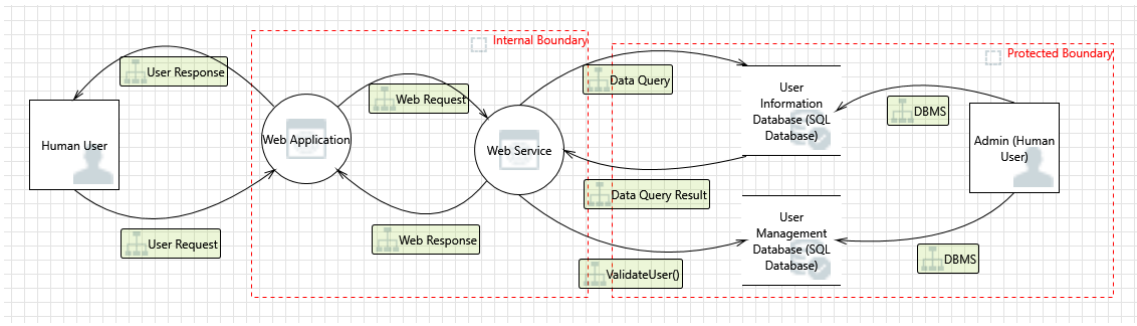


Figure 4.1: Data Flow Diagram in Microsoft Threat Modeling Tool

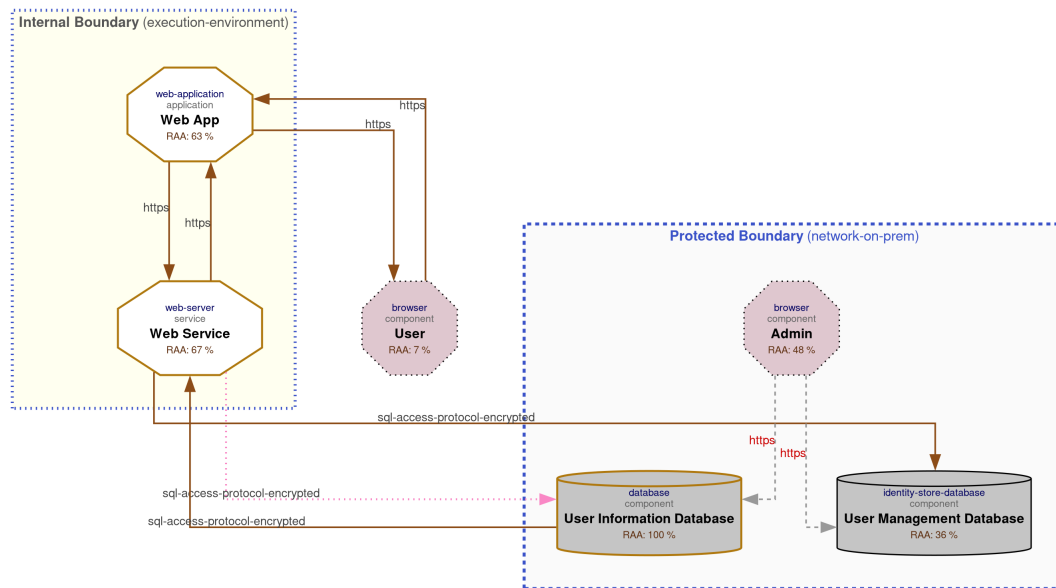


Figure 4.2: Data Flow Diagram in Threat Agile

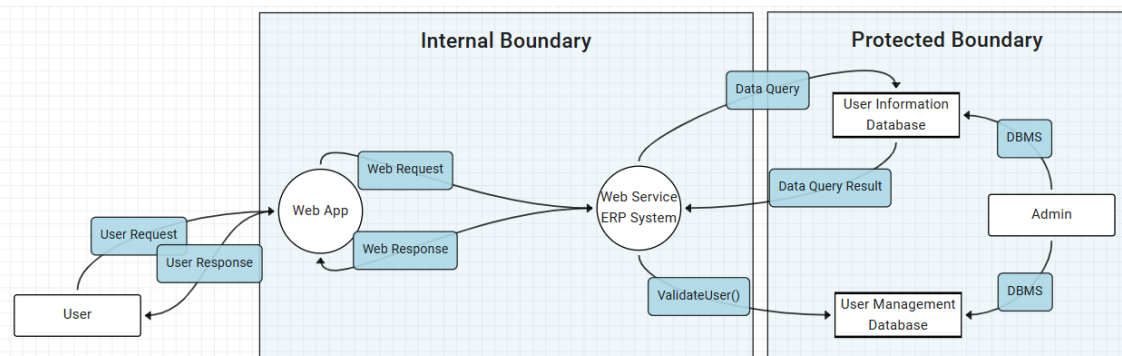


Figure 4.3: Data Flow Diagram in Thalaura

4.1.1 Risk and threat identification

Microsoft Threat Modeling Tool does not identify or calculate risks. Instead, it focuses on threats, which is still interesting to compare against, hence both concepts are analyzed. Note that the risks removed in Figure 4.5 should be interpreted as mitigated.

For system A, MSTMT identified a total of 51 threats. Among these, spoofing and tampering each accounted for 11 threats, followed by repudiation with 5, and information disclosure with 4. DoS had the highest number, with 12 threats, while elevation of privilege accounted for 8. However, in system B slightly less threats were identified as the amount of spoofing threats were reduced by 5 and one less threat against elevation of privilege was identified. Both tampering and information disclosure saw an increase of threats. Four additional tampering threats were identified, though three threats from system A were not present in system B. Similarly, the number of information disclosure threats rose by two, with five new threats and three removed which results in 48 total threats in system B.

Threat Agile identified a total of 42 risks against system A, slightly less than the number of threats MSTMT identified for the same system. Only 3 risks against spoofing were

identified, though for tampering and information disclosure, 10 and 12 risks were found, respectively. The most amount of risks were categorized as elevation of privilege, amounting to 17. Most notably, no risks were identified by repudiation or denial of service, neither for system A nor system B. Threat Agile was able to address a lot of risks when configured, especially against information disclosure and elevation of privilege, where 11 and 10 risks were removed respectively, resulting in a total of 30 different risks disappearing in system B. However, an additional 19 risks were identified in system B spread quite evenly across the STRIDE categories, with the notable exception of 8 new tampering risks.

The tool that identified the most amount of risks was Thalaura, with a total of 70 risks in system A. The distribution of risks is very even, where tampering, repudiation, information disclosure, and denial of service all have 14 risks associated with them. Spoofing had 8 identified risks, while elevation of privilege had the fewest, with only 6. Furthermore, in system B Thalaura was able to identify an additional 41 risks resulting in a total of 111 risks. Spoofing, tampering, information disclosure, and elevation of privilege all saw an increase in the number of risks, with tampering having the highest increase at 20 risks. No additional risks were identified against repudiation or denial of service. Additionally, Thalaura was able to remove or mitigate 95 of the identified risks, leaving only 12 risks that were unable to be mitigated.

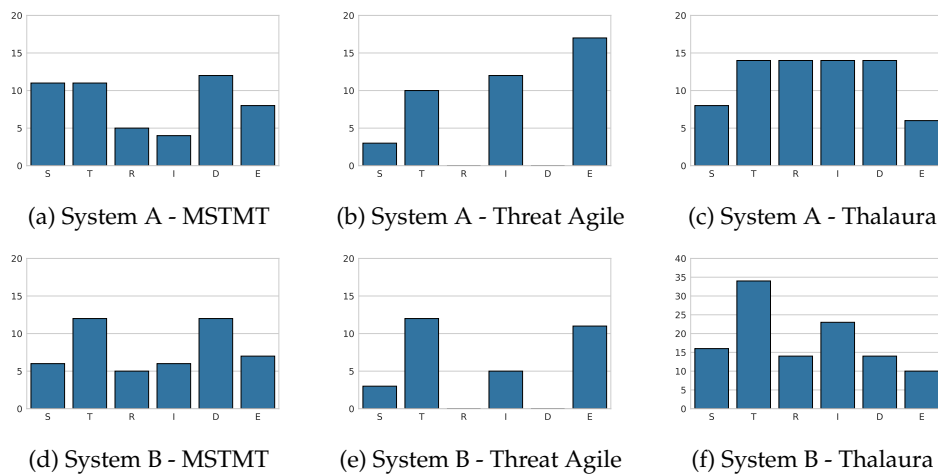


Figure 4.4: Threat and risk identification distributions

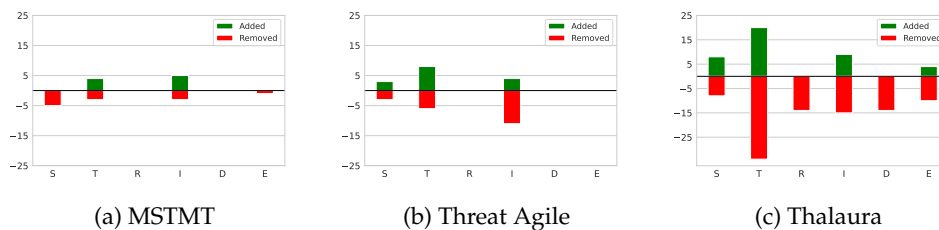


Figure 4.5: Difference of risks between system A and B

4.1.2 Identified risks in Thalaura

In Table 4.1 a subset of risks are presented that were identified using Thalaura during threat modeling. Each risk is categorized using the STRIDE framework, outlining the type of threat, its risk rating, and an appropriate mitigation strategy. These risks range in severity from low

to critical and include issues such as spoofing via missing authentication, elevation of privilege through absent authorization controls, and data leakage via insecure communication and side-channel attacks. The proposed mitigations emphasize established countermeasures like TLS 1.3 and role-based access control to reduce the platform’s overall attack surface. Note that the final risk cannot be mitigated through a security control.

Risk	Mitigation	STRIDE	Rating
Missing authentication allows adversary to impersonate as admin by bypassing authentication mechanisms with the intention to compromise the confidentiality and integrity of admin credentials. In worst case scenario, all data disclosed and all data totally corrupt.	TLS 1.3 Authentication	Spoofing	Critical
Missing authorization allows adversary to escalate its privileges by bypassing authorization mechanics with the intention to compromise the confidentiality, integrity and availability of user pii. In worst case scenario, extensive non-sensitive data disclosed, extensive slightly corrupt data and minimal primary services interrupted.	Role Based Access Control 2	Elevation of privilege	High
Unsecured communication channel allows adversary to compromise the confidentiality of user pii by data flow sniffing. In worst case scenario, extensive non-sensitive data disclosed.	TLS 1.3	Information disclosure	Medium
Cryptographic operations take different amounts of time allows adversary to compromise the confidentiality of user pii by side-channel attack. In worst case scenario, extensive non-sensitive data disclosed.		Information disclosure	Low

Table 4.1: Sample of detected risks in Thalaura

4.1.3 Identified risks in Threat Agile

The risks shown in Table 4.2 show risks identified using Threat Agile. One major issue identified is missing authentication on a communication link between the admin and the user information database, which could allow unauthorized access to sensitive user data and administrative interfaces, leading to an elevated privilege escalation risk. The presence of Cross-Site Request Forgery (CSRF) vulnerabilities in the web application exposes the system to spoofing attacks, where countermeasures are suggested for logged-in requests, cookies, and third-party products. Additionally, Threat Agile identifies unnecessary transfer of PII between the web service and web app. Each of these risks is addressed with targeted mitigation strategies and varies in severity.

Risk	Mitigation	STRIDE	Severity
Missing Authentication covering communication link User Database Information System Access from Admin to User Information Database	Apply an authentication method to the technical asset. To protect highly sensitive data consider the use of two-factor authentication for human users.	Elevation of privilege	Elevated
Cross-Site Request Forgery (CSRF) risk at Web App via User Request from User	Try to use anti-CSRF tokens of the double-submit patterns (at least for logged-in requests). When your authentication scheme depends on cookies (like session or token cookies), consider marking them with the same-site flag. When a third-party product is used instead of custom developed software, check if the product applies the proper mitigation and ensure a reasonable patch-level.	Spoofing	Medium
Unnecessary Data Transfer of User PII data at Web Service from/to Web App	Try to avoid sending or receiving sensitive data assets which are not required (i.e. neither processed or stored) by the involved technical asset.	Elevation of privilege	Low

Table 4.2: Sample of detected risks in Threat Agile

4.1.4 Identified risks in Microsoft Threat Modeling Tool

A small set of threats identified using MSTMT can be seen in Table 4.3. Firstly, a threat of remote code execution is found in the web application, where a human user may be able to achieve remote code execution. Secondly, MSTMT also identified weak credential storage, where it describes some different measures that can be taken to secure credentials. Finally, a denial of service threat is presented, indicating a potential for excessive resource consumption in the web service or user information database. This description explains that resource consumption may be difficult to deal with, but leveraging the operating system can aid in this problem.

Threat	Description	STRIDE
Web Application May be Subject to Elevation of Privilege Using Remote Code Execution	Human User may be able to remotely execute code for Web Application.	Elevation of privilege
Weak Credential Storage	Credentials held at the server are often disclosed or tampered with and credentials stored on the client are often stolen. For server side, consider storing a salted hash of the credentials instead of storing the credentials themselves. If this is not possible due to business requirements, be sure to encrypt the credentials before storage, using an SDL-approved mechanism. For the client side, if storing credentials is required, encrypt them and protect the data store in which they're stored	Information Disclosure
Potential Excessive Resource Consumption for Web Service or User Information Database (SQL Database)	Does Web Service or User Information Database (SQL Database) take explicit steps to control resource consumption? Resource consumption attacks can be hard to deal with, and there are times that it makes sense to let the OS do the job. Be careful that your resource requests don't deadlock, and that they do timeout.	Denial Of Service

Table 4.3: Sample of detected threats in Microsoft Threat Modeling Tool

4.1.5 View over risks in Thalaura

In Figure 4.6 three generated risks from Thalaura are presented. Once a risk is generated in the application the card appears containing all relevant information concerning the risk. From the card, it is possible to locate the component the risk stems from, the vulnerability, and the description of the risk, in addition to possible security controls that can be implemented to mitigate the risk.

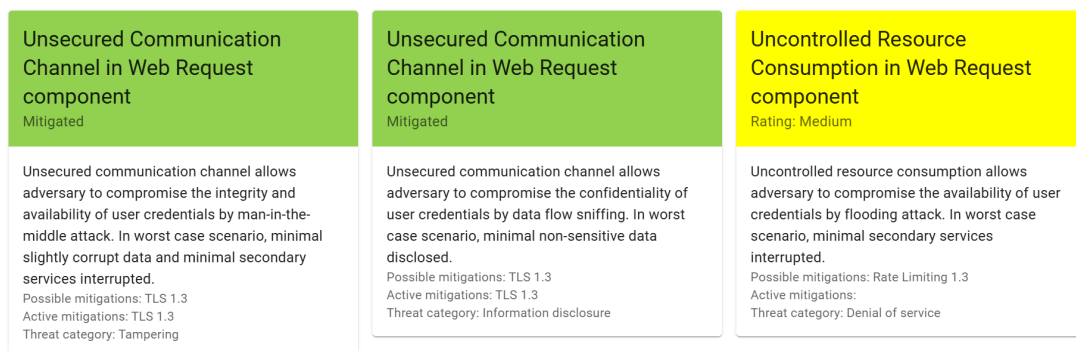


Figure 4.6: Risks as presented in Thalaura

4.2 Quantitative results

A Monte Carlo simulation was conducted to limit-test the methodology implemented in Thalaura by exploring a wide range of possible parameter combinations. The purpose of the simulation is to better understand how sensitive the risk rating is to changes in different input factors. This provides insight into which parameters have the most influence on the final risk score, helping to evaluate the consistency of the methodology under various conditions.

The result of the Monte Carlo simulation of 100,000 risks can be seen in Figure 4.7 and resulted in the following distribution: **Low** (21.7%), **Medium** (50.5%), **High** (25.0%), and **Critical** (3.2%). This suggests that most potential risks fall into the medium and high categories, which together account for over 75% of all outcomes. The most notable finding is the low proportion of critical risks, which suggests the model may be skewed toward lower-risk outcomes.

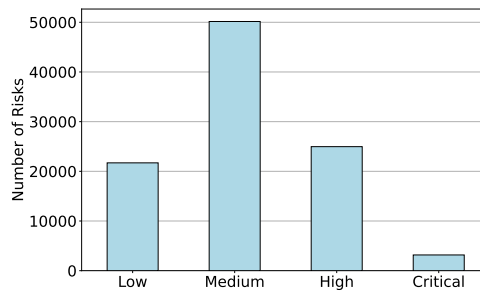


Figure 4.7: Risk Distribution of Monte Carlo Simulation

A closer look at the specific risk distributions across different STRIDE categories reveals slight variations, although the overall distribution remains very even. This is expected, as each category had an equal likelihood of being selected in the simulation. Figure 4.8 illustrates these distributions graphically, and Table 4.4 provides the exact counts and proportions for each risk rating level.

The **Spoofing** and **Tampering** categories have fairly similar distributions, with medium and high risks dominating both. In contrast, the categories **Repudiation**, **Information Disclosure**, and **Denial of Service** show no critical risks at all, and have a higher proportion of low and medium risks. Notably, the **Elevation of Privilege** category stands out with the highest proportion of severe risks, including 15.6% critical and 44.6% high-rated risks.

STRIDE Category	Total Risks	Low (%)	Medium (%)	High (%)	Critical (%)
Spoofing	16,931	14.8	47.0	36.6	1.7
Tampering	16,745	18.1	48.6	31.6	1.8
Repudiation	16,562	29.7	57.8	36.6	0.0
Information Disclosure	16,505	30.2	57.6	12.1	0.0
Denial of Service	16,931	30.3	57.3	12.5	0.0
Elevation of Privilege	16,665	7.1	32.8	44.6	15.6

Table 4.4: Risk Distribution Across STRIDE Categories

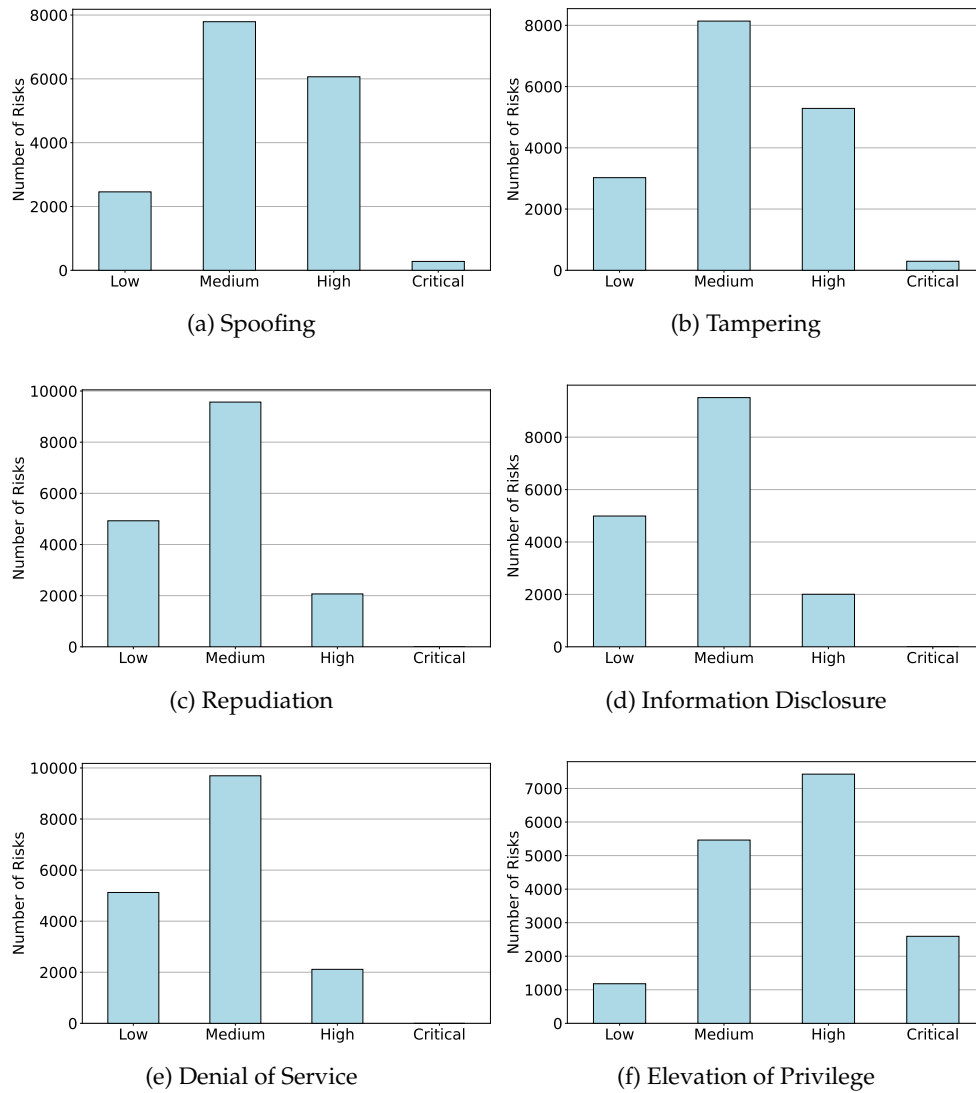


Figure 4.8: Distribution of the Amount of Risk Ratings Across STRIDE

Several correlations were identified by analyzing relationships between parameters and the computed impact and probability factors. The complete correlation matrix is presented in Appendix A.1, while Tables 4.5 and 4.6 summarize the most relevant findings.

As shown in Table 4.5, all three security properties have relatively balanced correlations with the impact factor. This suggests that each security property contributes to the overall impact factor independently and maintains a fair correlation across the CIA triad.

In contrast, the influence of STRIDE categories varies in its impact factor. Notably, *Elevation of Privilege* shows a strong correlation of **0.52**, which indicates that it has a strong effect on the expected impact factor. *Information disclosure* (**-0.21**), *Repudiation* (**-0.28**), and *Denial of Service* (**-0.26**) exhibit similar negative correlations, indicating a higher likelihood of negatively impacting the impact factor, whereas *Spoofing* (**0.13**) and *Tampering* (**0.08**) contribute little to the overall impact factor.

When it comes to the probability factors shown in Table 4.6, it is clear that the *Zone Value* (**0.92**) has the most influential effect on the probability factor. While the *Discovery Value* (**0.3**) and *Exploit Value* (**0.28**) do contribute, their influence is noticeably weaker in comparison to the zone value. This indicates that the location of the asset plays an important role when calculating the probability factor using this model.

Parameter	Correlation
Confidentiality	0.46
Integrity	0.56
Availability	0.43
Spoofing	0.13
Tampering	0.08
Repudiation	-0.28
Information Disclosure	-0.21
Denial of Service	-0.26
Elevation of Privilege	0.52

Table 4.5: Correlation between a set of parameters against the impact factor

Parameter	Correlation
Zone Value	0.92
Discovery Value	0.3
Exploit Value	0.28

Table 4.6: Correlation between a set of parameters against the probability factor

Figure 4.9 provides a more detailed view of how the probability and impact factors contribute to a risk's final classification. The five-number summary can be seen in Table 4.7 for the probability factors and Table 4.8 for impact factors.

The median **probability factor** increases consistently with the severity of the risk rating, rising from lower values for *Low* risk ratings to a median of **7.25** for both high and critical risks, where it then plateaus. They also exhibit a narrow and concentrated distribution, with most values falling between **4.5** and **5.5**, indicating relatively uniform probability. In contrast, *Medium* and *High* risks display broader variability, spanning values from **4.0** to **9.0**. The distribution for *High* risks skews slightly toward higher probabilities, as shown by a tighter interquartile range centered near the upper end of the scale. Finally, *Critical* risks show a tightly clustered distribution between **6.0** and **9.0**, suggesting consistently high probability values for *Critical* risks.

The median **impact factor** also increases steadily with the risk rating, starting from **1.67** for *Low* risks and rising to **6.33** for critical ones. *Low* risks display a compact distribution, with most values falling between **1.0** and **2.33**, with no outliers. *Medium* risks are more dispersed, ranging from **0.33** to over **5.5**, but still centered around a slightly higher median of **2.0**. A noticeable event occurs for *High* risk ratings where the minimum value is **3.0**, which is within the interquartile range. The *High* risk rating also shows diversity where many outliers reside, up to a maximum value of **9.0**. This indicates that despite having a variation of impact factors, *High* risk ratings tend towards lower impact factors, which is quite the opposite of the *High* risk rating for the probability factor. Similarly, *Critical* risk ratings also have a tight cluster where the minimum value is in the interquartile range, which further aligns with the general risk distribution of the Monte Carlo simulation, where it is difficult to achieve high impact factors that affect the final risk rating score.

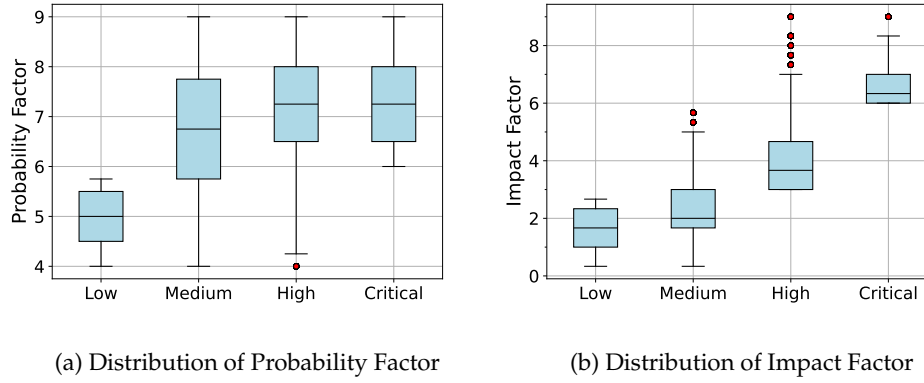


Figure 4.9: Distributions of Factors by Risk Ratings

Risk Rating	Min	Q1 (25%)	Median	Q3 (75%)	Max	Lower Bound	Upper Bound
Low	4.00	4.50	5.00	5.50	5.75	3.00	7.00
Medium	4.00	5.75	6.75	7.75	9.00	2.75	10.75
High	4.00	6.50	7.25	8.00	9.00	4.25	10.25
Critical	6.00	6.50	7.25	8.00	9.00	4.25	10.25

Table 4.7: Five-number summary for Probability Factor by Risk Rating.

Risk Rating	Min	Q1 (25%)	Median	Q3 (75%)	Max	Lower Bound	Upper Bound
Low	0.333	1.000	1.667	2.333	2.667	-1.000	4.333
Medium	0.333	1.667	2.000	3.000	5.667	-0.333	5.000
High	3.000	3.000	3.667	4.667	9.000	0.500	7.167
Critical	6.000	6.000	6.333	7.000	9.000	4.500	8.500

Table 4.8: Five-number summary for Impact Factor grouped by Risk Rating.



5 Discussion

This chapter discusses the qualitative and quantitative results, provides a critical evaluation of the applied methodology, considers the work in a broader context, and concludes by addressing the research questions.

5.1 Qualitative results

Two key features distinguish the Thalaura tool from others. First, it provides the capability to mitigate identified risks. As illustrated in Figure 4.5, Thalaura both introduces and eliminates significantly more threats or risks than the other tools. This behavior is by design. As users model a system and add details such as security controls, the tool responds immediately by introducing new risks associated with the added controls while mitigating any risks those controls are configured to address. The reason behind this is to make the threat modeling experience more responsive. Furthermore, it also creates a clear picture of what steps the threat model has taken. In MSTMT and Threat Agile it is unclear what the implemented security controls mitigate. Once a security control is implemented, the risk will no longer appear, as opposed to setting some status in the threat or risk as addressed or mitigated. In Thalaura the risks are directly addressed as mitigated, which makes it easy to understand the consequences of implementing security controls in terms of what they mitigate, but also the risks that come with them. This is possible to further nuance by calculating the impact a security control has on the risk. The fact that Thalaura can suggest security controls to implement to mitigate a threat makes the threat modeling experience more intuitive. Instead of debating what encryption standard to use, Thalaura can suggest a solution. Since security professionals configure this information, the developers can take confidence that the solution is sufficient and safe. It is the responsibility of the security professionals to ensure the data in the tool is correct and of high quality, so that the developers do not need to concern themselves with the risks except for mitigating them and communicating them.

Furthermore, another difference between the tools is the semantics of the risks and threats. This difference is observable by firstly comparing the first row in the Tables 4.1-4.3. The description of the first risk in Thalaura points out the vulnerability: missing authentication, the threat: adversary impersonate as admin, the threat vector: bypassing authentication mechanisms, the compromised security properties of the asset: confidentiality and integrity, and finally, the worst threat impacts which are data disclosed and all data totally corrupt. This is

hopefully enough information for the stakeholders to realize; we should probably implement some security control for authentication.

If we compare the semantics of the risk in Threat Agile it states the vulnerability: missing authentication, and the location which is a communication link between Admin and User Information Database. The impact of a realized risk is not implied here, however the description under mitigation suggests to apply an authenticated method on the dataflow. While it is an obvious example to implement authentication over dataflows, the tool leaves us wondering why this is important to implement. The reason why it is important to highlight is because we need insights to understand what mitigations are more important to implement first. In software, it is not reasonable to implement all security controls and therefore it is important to prioritize different risks. Finally, looking at MSTMT which generates threats, it says that the web application may be subject to elevation of privilege using remote code execution. MSTMT follows a software-centric approach and does consider specific components as assets. MSTMT generates quite straightforward descriptions of the threats which are simple to understand, however it does not consider security properties of assets.

Regarding the mitigation suggestions Threat Agile gives a more descriptive suggestion compared with Thalaura. However, during the risk mitigation process it was unclear how to make the threat model actually mitigate the risk. For example, the second risk in Table 4.2 suggest to use anti-CSRF tokens or marking cookies with the same-site flag. During the threat modeling the option to add this was not found thus the risk remains. This could obviously be fixed by addressing it with a comment, but the fact that it is not possible to do within the scope of the tool must be highlighted and is an important difference between the tools. It should be intuitive for someone threat modeling to address a risk through the tool. An issue with having security controls to mitigate all risks however is the sheer amount of data required. It is also not trivial to decide what a security control mitigates. In reality the relationship between security controls, vulnerabilities, threats and risks is very complex. Currently, in the implementation of Thalaura a vulnerability may only be mitigated by one security control, however, this should be expanded so that there are many options for users to give them a list of options that they may work with.

5.2 Quantitative results

The correlation matrix shows some interesting results of the method that Thalaura use and the limitations that come with it. Firstly, concerning the correlation between the impact factor and the several parameters shown in Figure 4.5, the results closely align with what should be expected. As for the security properties confidentiality, integrity, and availability, their correlation are very close to each other, ranging from 0.43 to 0.56. This shows that the security properties of assets are equally important when it comes to deciding the impact factor. As for the STRIDE categories the correlation varies by a lot. This is also reasonable and expected considering Table 3.4. Since a threat against elevation of privilege may compromise all three security properties it naturally has a higher impact on the impact factor. This also explains why repudiation, information disclosure and denial of service has a lower correlation, which is because they only concern one security property. The probability factor however has an extremely high correlation to the zone value. This might also not be surprising when looking at how the probability factors are calculated. The zone value is heavily weighted by the fact that two vulnerability factors awareness and intrusion detection are set by the zone value, but also the opportunity factor in the threat agent factors. It can make sense that the zone value has a high correlation to the probability factor. However, this number is currently slightly too high. The reason being that since the zone value is set by the threat modelers, they have a higher impact on the risk calculation than the security experts for the probability factor. Also, it does not make much sense that the intrusion detection factor is decided by the zone value. Instead, it would be reasonable to allow the threat modelers to configure what intru-

sion detection system they have in their component, which could positively impact the risks in that specific component. Furthermore, the threat agent factors in this model will always be critically high since the skill level, motive, and size of the threat agent were statically set to 9. This was initially due to the nature of the domain in the telecom industry. However, the consequence of this is visible in Figure 4.9 where the probability factor can never go below a value of 4. Since all risks are compared to each other in the same domain, it does not make sense to have this limitation. It becomes more challenging to assess which risks to prioritize since the median risk will lean towards a higher risk rating, and risks that would typically be put into a low rating are bumped up to medium. While it is understandable to mark risks as more severe because threat agents in this domain have more resources, it should also be possible to make a distinction between different domains. In order to address this problem, threat agents should also be configurable like some of the other parameters used in calculating the risk rating. For example, a developer could select potential threat agents where the tool would calculate the risk rating with the worst-case threat agent value.

Finally, it is important to consider that the quantitative results reflect random sampling from input distributions. In practice, the data input into the tool should be well-considered by the security experts, but it may introduce its own bias. The analysis of the quantitative results aims to serve as a guide for security experts to understand the consequences of their decisions when setting values for vulnerabilities and threat vectors. For example, setting a threat vector that threatens elevation of privilege will have a higher impact on the final risk than setting it to spoofing, as it also considers the assets' integrity and availability. By understanding the bias of the system, it can be configured to counteract it.

5.3 Critique of Method

The method used in this thesis combines the OWASP Risk Rating methodology with STRIDE and an asset-centric approach to automated threat modeling. The goal was to design a flexible yet structured process for identifying and evaluating security risks based on system architecture. This combination enables risk assessment from a technical perspective while maintaining clear traceability between system components, threats, and their associated risks. The following subsections discuss and critique how each of these methods was implemented and adapted within the tool.

5.3.1 OWASP Risk Rating

The OWASP risk rating method worked really well when building the application and to evaluate the risks. Since the method relies on setting different parameters, all of the parameters can be tied to different aspects of the threat model. Incorporating different pieces of the system is easy using this method for example when setting the zone value or when configuring new threat vectors. The concept of calculating the risk of different parameters both in the architecture of the system and also from the vulnerabilities themselves is a powerful combination and allows for interplay between threat modelers and security experts, incorporating both perspectives into a single risk rating. The benefit of the method is also that it is easy to expand upon and the parameters themselves can be adjusted for whatever you would like. In this case it was probably unnecessary to incorporate both awareness and intrusion detection in the vulnerability factors since they are both set by the zone value. Instead, it could be possible to have a new vulnerability factor tied to the zone value. The intrusion detection parameter could be set by threat modelers explaining what intrusion detection system they currently have in their application. This is a very concrete example of the possibilities with this method and is very simple to incorporate in a tool and great for automation of risk calculation.

5.3.2 STRIDE

A difficult aspect of implementing the method was combining the OWASP risk rating method with STRIDE and the asset-centric threat modeling approach. First, a mapping between the security properties of an asset was made to the STRIDE category. Then, the risk calculation was slightly adjusted to make it work for assets that are not compromised by a threat vector with a certain STRIDE category. In practice, the STRIDE category only decides what security properties are compromised through a certain threat vector. It is unclear if the STRIDE mapping to security properties is always correct or if there are exceptions for this mapping. Another solution could be to set the security properties directly in the threat vector, or through some other method decide what security properties are compromised by a threat. STRIDE added another complexity to the method and eventually all it did was to categorize certain threat vectors labeling them with the threat type of spoofing for example. It is a useful method when manually threat modeling to guide the threat modelers towards certain threats. However, when designing a tool to automatically identify risks, the benefit of using STRIDE diminishes. It is still useful for security experts to feed data into the tool but it does not necessarily need to be categorized into STRIDE categories. The main focus should be the security properties of the assets, and to keep the domain easy to understand it is important to not add additional complexity. There might be some other advantage of using STRIDE for integration purpose, but for this method it is not evident.

5.3.3 Data source

The tool lacks integration with a common vulnerability data source, which is disadvantageous. However, it is not apparent how to integrate open data sources into a threat modeling tool. Given the variety of databases containing information about vulnerabilities and threats, selecting the most appropriate database poses a challenging decision. Not only is the information different on these databases, but there are also several different methods used when calculating risks or defining vulnerabilities. Furthermore, the lists are not entirely defined to perform specific calculations. Vulnerabilities are generally quite ambiguous, which makes them very difficult to define in a way that is suitable for several different methods. Therefore, when building a threat modeling tool, it is crucial to consider the data source and perhaps start the process from this direction.

Another limitation encountered was the inability to evaluate the business impact metric. This metric is particularly valuable for risk analysis, as it captures the potential financial and reputational damage of a given threat. From a high-level business perspective, the cost of implementing a security control (or the cost of failing to do so) is often a key factor in deciding appropriate risk response. Unfortunately, quantifying these costs is difficult, as it requires reliable data on security incidents and control effectiveness. Calculating return on investment in this context is especially complex, involving variables that are often speculative or context-dependent. In fact, accurately defining the metrics and methodologies for risk prioritization and automated mitigation is a research field in its own right. Nevertheless, the lack of reliable data made it extremely difficult to implement meaningful mitigation suggestions or to prioritize between risks that fall into similar categories. Without concrete metrics, distinguishing which risks deserve higher attention becomes highly subjective.

5.3.4 Data Flow Diagrams

Using DFDs when threat modeling is essential to gain an understanding of the architecture of the system. It is interesting to see the difference between the tools in regards to not only how the process of drawing the architecture is, but also how different concepts are presented and the difference in how the components function and generate risks. In Thalaura we deploy a Zero Trust mindset when generating the risks. Regardless if the boundary is external, internal or protected all risks are generated in the same way but with a lower risk score if the

component is in a more protected boundary. When working with MSTMT there were discoveries not evident from the results, but the tool has a different view on the risk generation in regards to the trust boundaries. In MSTMT, it can actually avoid to generate risks if they belong in a component under a trust boundary. Interestingly enough, this does not apply for all risks. Even though some components are located under a trust boundary risks are generated, but it was noticeable when removing the trust boundaries in MSTMT that even more risks appeared. This became an unpredictable process where the tool is not transparent with the selection of risks. Most likely, there is some functionality behind the scenes that decide which risks are too severe to hide even though behind a trust boundary. But considering that MSTMT does not calculate the risk, but only threats, this becomes even more confusing on how the decisions are made. This an important difference to mention because in Thalaura these risks are generated and shown. Even risks that are mitigated are possible to see in Thalaura but instead marked with a status showing that is has been addressed and mitigated. These are core concepts for maintaining threat models in the SDLC and tracing progress is important to understand why certain decisions has been made, but also understand how the tool works.

Furthermore, traceability in general when working with security is extremely important. It is therefore important to understand how automatic threat modeling tool works to completely rely on them, and this begins with transparency on how the tool generates risks and how said risks are generated. This brings us to another interesting topic, considering Threat Agile. It is an open-source application, which means that it has great transparency, which deserves recognition. However, some of the methods used to generate risks in Threat Agile are not simple to understand and functionality is separately divided across classes. It can make sense implementation wise and does make it easier to treat risks individually, however, the calculation of the risk does not follow a clear methodology. The reason behind this is that Threat Agile base their information from external databases such as the CWE. In this database weaknesses are labeled with some likelihood factor, however, it over-simplifies the problem when approaching with the asset-centric point of view. When applying the asset-centric methodology, the assets are important. It falls short on this promise as the risk descriptions do not directly correlate to the de facto risks against the asset. What is going to happen to this asset if a risk is realized? This is where Thalaura has an edge, as the risk descriptions directly tell you exactly how the asset can be compromised and what the consequences are, highlighting the real issues which are easier to grasp for stakeholders.

5.3.5 Risk Generation Algorithm

While there are several areas where Thalaura can be improved, the current version of the tool provides a strong foundation for further development and experimentation. One of the most successful outcomes of this thesis is the design and implementation of the risk generation algorithm, which functioned as intended throughout testing. It consistently produced relevant and descriptive risks based on the defined input parameters, demonstrating both reliability and consistency. This algorithm lays the groundwork for more advanced features, such as automated risk prioritization or integration with external data sources, which could significantly enhance the tool's capabilities in future iterations. Furthermore, this thesis has so far focused on addressing risks through mitigation, which is one of the four strategies for risk response. In practice, some risks may be necessary to transfer, avoid, or accept, which is currently not supported in Thalaura. Avoiding risks may be possible by removing a feature (such as a security control). This would, however, go undocumented. As mentioned, many improvements can be made in the tool, and allowing features for addressing risks via all risk response strategies is essential in cases where security controls may be unavailable or if other solutions are better in that specific case.

5.4 The work in a wider context

In general, it is pretty challenging to grasp the complexity of threat modeling since many different aspects affect the results. This thesis explains the effort and process of creating a tool to generate and analyze real risks based on a DFD. While there are many areas of improvement, the thesis highlights the risk generation process and the risks' evaluation. The results show that it is possible to improve and create a threat modeling application to allow threat modelers to mitigate risks in their system. This can be expanded upon by following up on security bugs and integrating a threat modeling tool with a project tracking software. If integrated well enough, the threat modeling process can become closer to the development process through issue tracking. The threat modeling process would become simpler and further integrate into the SDLC, where a developer could either add their changes through the ticket system or by loading the threat model project and adding new security controls into the threat model.

The automation of threat modeling presents both opportunities and ethical challenges. The impact of such a tool, which can automatically assess potential security risks, can contribute to societal resilience against threat agents and adversaries in critical infrastructure. However, there is a possible danger revolving around the accuracy of automated approaches, and there is a risk of a false sense of security when relying solely on automated threat modeling tools. Therefore, automatic threat modeling tools must be transparent and easily understood. Furthermore, assessment and validation of these tools are also critical. There has not been much research on how to assess automatic threat modeling tools, and there is definitely an area of improvement that can be made to validate tools through quantitative results.

Another crucial factor that is important to consider when threat modeling is the aspect of human interaction. Even the most secure systems can be compromised through social engineering where adversaries manipulate individuals rather than exploiting technical vulnerabilities. These factors can be challenging to model since human behavior can be unpredictable. This is especially true for any personnel who have direct access to sensitive information or critical infrastructure, such as system administrators, developers with production access, or customer support staff. These individuals become high-value targets for adversaries, and it is therefore important to model potential threat vectors against them. While these types of vulnerabilities may fall in the broader variation of information security, it is still possible to highlight areas with elevated access or insufficient audit trails.

5.5 Answering the Research Questions

Research Question 1: *What data and metadata from data flow diagrams (DFDs) are necessary for developing an asset-centric risk algorithm using STRIDE threat modeling?*

Throughout the research of this thesis and the development of the risk algorithm, many different types of data were discovered to aid in the development of the risk algorithm. Firstly, data concerning the asset is essential for the asset-centric approach. This includes the severity or technical impact if the asset is compromised regarding confidentiality, integrity, and availability. It is also necessary to understand the environment of the asset. In this case, an asset is seen as digital data that may exist in three states: in transit, in use, and at rest. This is very useful since it can be combined with components in a DFD. An asset in transit can be correlated to the data flows, and an asset in use can be associated with processes. Finally, an asset at rest is the equivalent of storing an asset in data storage.

Furthermore, additional information can be provided in the DFDs. In Thalaura, further details can be provided for components concerning security controls and where the assets are located. Standard vulnerabilities were provided for data flows, processes, and storage to enable the risk generation algorithm. The idea is to provide generic and expected vulnera-

bilities that must always be mitigated. When users attempt to mitigate these vulnerabilities through security controls, they feed more information into the DFDs, which may generate more risks. This gives Thalaura the possibility of layered risk generation.

Research Question 2: *How can the identified data and metadata be applied to assess and prioritize security risks?*

This thesis introduces a novel risk generation algorithm that maps STRIDE threats to compromised security properties of assets based on the technical and business impact. Additionally, the risks were quantified with likelihood factors using the OWASP Risk Rating methodology by deriving vulnerability factors from the DFD through the zone value of trust boundaries. To assess the risks, the risk value and semantics can be used to understand their impact. Currently, the most straightforward method to prioritize security risk is by looking at the rating and prioritizing the most critical risks.

From a security perspective, all identified risks should be addressed, which might not always be through mitigation. The addressment of risks is not a topic brought up in this thesis. However, a risk may be avoided, transferred, mitigated, or accepted. Since all risks must go through this process, the necessity of prioritizing risks is not so evident, and as mentioned, all risks should be seen as security bugs and finally be resolved.

Research Question 3: *How does such an algorithm compare to manual or other methods of risk assessments in terms of accuracy, coverage, and consistency?*

The evaluation results demonstrate that Thalaura outperformed both MSTMT and Threat Agile regarding total risk coverage for systems A and B. Thalaura successfully mitigated 99 out of 111 risks, a clear majority of all risks. In contrast, the other tools do not support dynamic risk mitigation. For instance, with Threat Agile, re-analysis of the system is required, and previously identified risks may not be regenerated, limiting traceability.

A key advantage of Thalaura is its ability to maintain a transparent and traceable mitigation process. All mitigated risks remain accessible for review, ensuring accountability and clarity. Additionally, Thalaura excels in accuracy, generating risks that directly reflect the user-provided system specifications, including DFD inputs, vulnerabilities, and applied security controls. The accuracy of the risks is difficult to measure due to the simplicity of the test case. A better measurement would be to compare the results from a threat model by security experts on a real application against the performance of Thalaura on the same application.

Regarding the consistency of Thalaura, it is deterministic, which means that the risk generation is predictable. All risks are generated the same way through the risk generation algorithm, and the risk values are always determined using the same method. These factors ensure that Thalaura remains consistent and transparent. Furthermore, Thalaura employs a Zero-Trust principle and treats all components equally regardless of their trust boundaries. This is also great for consistency and ensures comprehensive risk coverage. In contrast, MSTMT may not generate threats between components within the same trust boundary, and the criteria for when specific threats are identified versus when they are not lack clarity. This inconsistency can hinder risk analysis and decision-making.



6 Conclusion

This final chapter provides a summary of the work presented in this thesis and draws a conclusion based on the research findings. The chapter begins by highlighting the main achievements and key contributions of the Thalaura tool. It concludes by outlining potential directions for future research, discussing how the work can be extended and further validated.

6.1 Conclusion

This thesis has presented the design, implementation, and evaluation of Thalaura, an automated threat modeling tool that supports security risk assessment using Data Flow Diagrams (DFDs). The work demonstrates that it is possible to generate structured and meaningful security threats and associated risks based on technical impact factors, known vulnerabilities, and existing or missing security controls. By automating key parts of the threat modeling process, Thalaura reduces manual effort and promotes consistency across assessments, while still allowing for flexibility in defining assets and security controls.

The evaluation of Thalaura against existing tools and theoretical frameworks shows promising results, particularly in its ability to produce high-granularity risks with a clear mapping to system components. Unlike some traditional approaches that rely heavily on manual input or generic threat libraries, Thalaura enables a contextualized approach, which is especially valuable in domains like telecom, where system complexity and operational risk are inherently high. While the tool focuses on technical aspects of threat modeling, it lays the groundwork for future extensions, such as integrating risk tracking systems, refining threat agents, or expanding to include permission levels for specific assets. Overall, this work contributes to the development of more practical, efficient, and transparent threat modeling practices suitable for integration into real-world software development lifecycles.

6.2 Future work

While comparing different sets of tools is a good starting point for evaluating them, such comparisons present challenges. An issue is that each tool relies on different underlying datasets, methodologies, and assumptions, making direct comparison difficult. To gain more meaningful insights into the real-world applicability and effectiveness of the Thalaura tool,

it would be beneficial to benchmark it against established threat models created by experienced security professionals. This would allow for a more grounded assessment of the tool's accuracy and usability.

It would also be interesting to analyze different approaches to automatic threat modeling, such as pattern matching or algorithmic approaches, to help understand if the choice of underlying methodology has any meaningful difference in impact when it comes to risk generation.

Furthermore, incorporating feedback from both developers and security experts through hands-on testing could offer valuable perspectives on the tool's usability. Developers could evaluate how intuitive and supportive the tool is during the design and development process, while security professionals could assess the quality and relevance of the generated risks. This perspective would not only help validate the technical capabilities of the tool but also reveal how well it integrates into existing workflows. Such a comprehensive evaluation approach could guide further improvements and increase confidence in Thalaura's effectiveness for real-life threat modeling scenarios.



Bibliography

- [1] Ijaz Ahmad, Shahriar Shahabuddin, Tanesh Kumar, Jude Okwuibe, Andrei Gurtoy, and Mika Ylianttila. “Security for 5G and Beyond”. In: *IEEE Communications Surveys Tutorials* 21.4 (2019), pp. 3682–3722. DOI: 10.1109/COMST.2019.2916180.
- [2] Karin Bernsmed, Daniela Soares Cruzes, Martin Gilje Jaatun, and Monica Iovan. “Adopting threat modelling in agile software development projects”. In: *Journal of Systems and Software* 183 (2022), p. 111090. ISSN: 0164-1212. DOI: <https://doi.org/10.1016/j.jss.2021.111090>.
- [3] Jin Cao, Maode Ma, Hui Li, Ruhui Ma, Yunqing Sun, Pu Yu, and Lihui Xiong. “A Survey on Security Aspects for 3GPP 5G Networks”. In: *Commun. Surveys Tuts.* 22.1 (Jan. 2020), pp. 170–195. ISSN: 1553-877X. DOI: 10.1109/COMST.2019.2951818.
- [4] Dayna Eidle, Si Ya Ni, Casimer DeCusatis, and Anthony Sager. “Autonomic security for zero trust networks”. In: *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*. 2017, pp. 288–293. DOI: 10.1109/UEMCON.2017.8249053.
- [5] Michael A. Enright, Eman Hammad, and Ashutosh Dutta. “A Learning-Based Zero-Trust Architecture for 6G and Future Networks”. In: *2022 IEEE Future Networks World Forum (FNWF)*. 2022, pp. 64–71. DOI: 10.1109/FNWF55208.2022.00020.
- [6] OWASP Foundation. *OWASP Top 10 Proactive Controls 2024 (Archive)*. Accessed: 2025-05-26. 2024. URL: <https://top10proactive.owasp.org/archive/2024/>.
- [7] OWASP Foundation. *Threat Modeling Process*. https://owasp.org/www-community/Threat_Modeling_Process. Accessed: 2024-10-09. 2024.
- [8] Daniele Granata and Massimiliano Rak. “Systematic analysis of automated threat modelling techniques: Comparison of open-source tools”. In: *Software Quality Journal* 32.1 (2023), pp. 125–161. ISSN: 0963-9314. DOI: 10.1007/s11219-023-09634-4.
- [9] Anne Honkaranta, Tiina Leppänen, and Andrei Costin. “Towards Practical Cybersecurity Mapping of STRIDE and CWE — a Multi-perspective Approach”. In: *2021 29th Conference of Open Innovations Association (FRUCT)*. 2021, pp. 150–159. DOI: 10.23919/FRUCT52173.2021.9435453.
- [10] International Organization for Standardization and International Electrotechnical Commission. *Information technology — Security techniques — Information security management systems — Overview and vocabulary*. ISO/IEC 27000:2018. 2018.

- [11] Manju Khari, Vaishali, and Prabhat Kumar. "Embedding security in Software Development Life Cycle (SDLC)". In: *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. 2016, pp. 2182–2186.
- [12] Loren Kohnfelder and Praerit Garg. "The Threats to Our Products". In: *Microsoft Interface* 33 (1999), pp. 1–8.
- [13] Pedro Lohmann, Carlos Albuquerque, and Raphael Machado. "Systematic Literature Review of Threat Modeling Concepts". In: Mar. 2023. DOI: 10.5220/0000168400003405.
- [14] V Maheshwari and M Prasanna. "Integrating risk assessment and threat modeling within SDLC process". In: *2016 International Conference on Inventive Computation Technologies (ICICT)*. Vol. 1. 2016, pp. 1–5. DOI: 10.1109/INVENTIVE.2016.7823275.
- [15] Tim P. Morris, Ian R. White, and Michael J. Crowther. "Using simulation studies to evaluate statistical methods". In: *Statistics in Medicine* 38.11 (2019), pp. 2074–2102. DOI: 10.1002/sim.8086.
- [16] National Institute of Standards and Technology. *Security and Privacy Controls for Information Systems and Organizations*. NIST Special Publication 800-53 Revision 5. Includes updates as of December 10, 2020. Supersedes SP 800-53 Rev. 5 (09/23/2020). National Institute of Standards and Technology, Sept. 2020. URL: <https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final>.
- [17] NGMN Alliance. *NGMN 5G White Paper*. White Paper. Accessed: 2025-06-10. Frankfurt, Germany: Next Generation Mobile Networks (NGMN), 2015. URL: <https://www.ngmn.org/wp-content/uploads/NGMN-5G-White-Paper-V1-0.pdf>.
- [18] Livinus Nweke and Stephen Wolthusen. "A Review of Asset-Centric Threat Modelling Approaches". In: *International Journal of Advanced Computer Science and Applications* 11 (Mar. 2020), pp. 1–6. DOI: 10.14569/IJACSA.2020.0110201.
- [19] Adrian Rose;n. "Achieving Zero Trust : A Strategic Roadmap for Phased Implementation of Zero Trust Architecture". MA thesis. Linköping University, Database and information techniques, 2025, p. 72.
- [20] Zakir Ahmad Sheikh and Yashwant Singh. "A Hybrid Threat Assessment Model for Security of Cyber Physical Systems". In: *2022 Seventh International Conference on Parallel, Distributed and Grid Computing (PDGC)*. 2022, pp. 582–587. DOI: 10.1109/PDGC56933.2022.10053332.
- [21] Adam Shostack. *Threat Modeling: Designing for Security*. Crosspoint: Wiley, 2014.
- [22] Daniela Soares Cruzes, Martin Gilje Jaatun, Karin Bernsmed, and Inger Anne Tøndel. "Challenges and Experiences with Applying Microsoft Threat Modeling in Agile Development Projects". In: *2018 25th Australasian Software Engineering Conference (ASWEC)*. 2018, pp. 111–120. DOI: 10.1109/ASWEC.2018.00023.
- [23] Philipp Theurich, Jens Witt, and Sebastian Richter. "Practices and challenges of threat modelling in agile environments". In: *Informatik Spektrum* 46.3 (2023), pp. 220–229. DOI: 10.1007/s00287-023-01549-5.
- [24] Katja Tuma, Laurens Sion, Riccardo Scandariato, and Koen Yskout. "Automating the early detection of security design flaws". In: *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*. MODELS '20. Virtual Event, Canada: Association for Computing Machinery, 2020, pp. 332–342. ISBN: 9781450370196. DOI: 10.1145/3365438.3410954.
- [25] Rossouw von Solms and Johan van Niekerk. "From information security to cyber security". In: *Computers Security* 38 (2013). Cybercrime in the Digital Economy, pp. 97–102. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2013.04.004>.

- [26] Jeff Williams. *OWASP Risk Rating Methodology*. https://owasp.org/www-community/OWASP_Risk_Rating_Methodology. Accessed: 2024-10-11. 2024.
- [27] Wenjun Xiong and Robert Lagerström. "Threat modeling – A systematic literature review". In: *Computers & Security* 84 (2019), pp. 53–69. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2019.03.010>.



A

Appendix

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)
(1) Impact Factor	1.0	-0.0013	0.0003	-0.0025	-0.0032	0.4643	0.5672	0.4339	0.1251	0.0789	-0.2842	-0.191	-0.2479	0.5194	-0.3554	-0.2864	0.4878	0.4486
(2) Probability Factor	-0.0013	1.0	0.9141	0.2918	0.2791	-0.0045	-0.0006	0.0033	-0.0031	0.0018	0.0046	-0.0051	-0.0004	0.0022	-0.5627	0.1653	0.2971	0.1182
(3) Zone Value	0.0003	0.9141	1.0	-0.005	0.0007	-0.0016	-0.0006	0.0028	-0.0027	0.0014	0.0045	-0.004	-0.0008	0.0015	-0.5342	0.1564	0.2833	0.1109
(4) Discovery Value	-0.0025	0.2918	-0.005	1.0	0.0063	-0.006	0.0002	0.0024	-0.004	0.0023	0.0032	-0.0035	-0.0016	0.0036	-0.1279	0.0362	0.0674	0.0311
(5) Exploit Value	-0.0032	0.2791	0.0007	0.0063	1.0	-0.0044	-0.0004	0.0001	0.0019	-0.0007	-0.0019	-0.0013	0.003	-0.0011	-0.13	0.0416	0.0648	0.027
(6) Confidentiality	0.4643	-0.0045	-0.0016	-0.006	-0.0044	1.0	-0.0838	-0.2599	0.3305	-0.359	-0.3566	0.4413	-0.3614	0.3095	-0.1529	-0.1253	0.2101	0.1983
(7) Integrity	0.5672	-0.0006	-0.0006	0.0002	-0.0004	-0.0838	1.0	-0.0821	0.1785	0.1864	0.2791	-0.4004	-0.4066	0.1647	-0.2233	-0.1672	0.3109	0.2347
(8) Availability	0.4339	0.0033	0.0028	0.0024	0.0001	-0.2599	-0.0821	1.0	-0.3374	0.3009	-0.3371	-0.3364	0.4178	0.2863	-0.1448	-0.1274	0.194	0.2252
(9) Spoofing	0.1251	-0.0031	-0.0027	-0.004	0.0019	0.3305	0.1785	-0.3374	1.0	-0.2	-0.1987	-0.1983	-0.2014	-0.1995	-0.0746	-0.0286	0.1194	-0.038
(10) Tampering	0.0789	0.0018	0.0014	0.0023	-0.0007	-0.359	0.1864	0.3009	-0.2	1.0	-0.1998	-0.1994	-0.2025	-0.2006	-0.0396	-0.014	0.0685	-0.036
(11) Repudiation	-0.2842	0.0046	0.0045	0.0032	-0.0019	-0.3566	0.2791	-0.3371	-0.1987	-0.1998	1.0	-0.1981	-0.2011	-0.1992	0.0869	0.0678	-0.1285	-0.0806
(12) Information Disclosure	-0.191	-0.0051	-0.004	-0.0035	-0.0013	0.4413	-0.4004	-0.3364	-0.1983	-0.1994	-0.1981	1.0	-0.2007	-0.1988	0.092	0.0663	-0.1317	-0.0804
(13) Denial of Service	-0.2479	-0.0004	-0.0008	-0.0016	0.003	-0.3614	-0.4066	0.4178	-0.2014	-0.2025	-0.2011	-0.2007	1.0	-0.2019	0.0937	0.064	-0.1302	-0.0816
(14) Elevation of Privilege	0.5194	0.0022	0.0015	0.0036	-0.0011	0.3095	0.1647	0.2863	-0.1995	-0.2006	-0.1992	-0.1988	-0.2019	1.0	-0.1586	-0.1554	0.2026	0.3166
(15) Risk Rating Low	-0.3554	-0.5627	-0.5342	-0.1279	-0.13	-0.1529	-0.2233	-0.1448	-0.0746	-0.0396	0.0869	0.092	0.0937	-0.1586	1.0	-0.5282	-0.3037	-0.0952
(16) Risk Rating Medium	-0.2864	0.1653	0.1564	0.0362	0.0416	-0.1253	-0.1672	-0.1274	-0.0286	-0.014	0.0678	0.0663	0.064	-0.1554	-0.5282	1.0	-0.5787	-0.1814
(17) Risk Rating High	0.4878	0.2971	0.2833	0.0674	0.0648	0.2101	0.3109	0.194	0.1194	0.0685	-0.1285	-0.1317	-0.1302	0.2026	-0.3037	-0.5787	1.0	-0.1043
(18) Risk Rating Critical	0.4486	0.1182	0.1109	0.0311	0.027	0.1983	0.2347	0.2252	-0.038	-0.036	-0.0806	-0.0804	-0.0816	0.3166	-0.0952	-0.1814	-0.1043	1.0

Table A.1: Final Correlation Matrix with STRIDE Categories Ordered and Risk Ratings Last