Department of Informatics

Threat-Oriented Cyber Resilience: Balancing Risk and Reward Through Prevention, Detection, and Mitigation

Dissertation submitted to the
Faculty of Business, Economics and Informatics
of the University of Zurich

to obtain the degree of
Doktor der Wissenschaften, Dr. sc.
(corresponds to Doctor of Science, PhD)

presented by
Jan von der Assen
from
Degersheim, St. Gallen, Switzerland

approved in July 2025

at the request of
Prof. Dr. Burkhard Stiller
Prof. Dr. Abhishta Abhishta
Prof. Dr. Alberto Huertas Celdran

The Faculty of Business, Economics and Informatics of the University of Zurich hereby authorizes the printing of this dissertation, without indicating an opinion of the views expressed in the work.

Zürich, July 16, 2025

The Chairperson of the Doctoral Board: Prof. Elaine Huang, Ph.D.

© 2025 - Jan von der Assen
All rights reserved.

Abstract

Cyber attacks have remained a key concern for organizations, nations, and individuals due to their increasing frequency and impact. Establishing a perfect posture against such attacks is non-trivial from a technical perspective. In addition, economic considerations only justify certain expenses to thwart attacks. As a result, organizations strive for optimal resilience, *i.e.*, an ability to withstand attacks while continuing to do their business.

Practices to improve cyber resilience can be categorized as either being *preventive*, *detective*, or *responsive*. Prevention is a critical pillar since it shifts security efforts into a stage where changes are cheaper to implement. Detective measures are implemented, acknowledging that prevention may not be perfect. Such measures reduce the time for response, minimizing the impact of an attack. A systematically executed response then aids an organization in limiting damage and returning operations to a manageable state.

Establishing a cybersecurity program framed through preventive, detective, and responsive measures requires navigating numerous practices, models, and solutions. A risk-based perspective abandons a one-size-fits-all approach, where a specific set of security controls is considered optimal for any scenario. Instead, efforts are focused on assets that are sensitive to attacks. The conceptual frame for running a cyber risk program is well-established, and multiple methodologies, frameworks, and standards exist. Implementing such practices is nevertheless challenging, and numerous limitations exist in practice.

This PhD thesis considers those challenges within a risk-based security program as research opportunities. Through a set of design studies, novel solutions are proposed and evaluated in the context of specific challenges in prevention, detection, and response. To prevent risks, the proposed threat modeling solutions leverage visual architectural modeling to address the role of collaboration, the significance of insider attacks, and the emergence of AI-related attacks. In addition, the problem of

quantification is addressed through solutions exploiting continuity measures. The results indicate improved effectiveness and efficiency in those problem domains while acknowledging and raising attention to underlying limitations, such as the lack of data available or the infeasibility of achieving full automation. Detection and response are considered closely related practices, and novel solutions are proposed to gather attack data and use it in the scope of proactive and reactive defense against sets of threats. Testing the developed solutions attests to the viability of both defense paradigms while being closely coupled to the specific attack and defense scenarios.

Kurzfassung

Cyberangriffe sind aufgrund ihrer zunehmenden Häufigkeit und Auswirkungen nach wie vor ein relevantes Anliegen für Organisationen, Staaten und Einzelpersonen. Der Aufbau einer perfekten Abwehr gegen solche Angriffe ist aus technischer Sicht unrealistisch. Zudem rechtfertigen wirtschaftliche Betrachtungen nur gewisse Investitionen zur Abwehr. Infolgedessen streben Unternehmen ein optimales Mass an Resilienz an – die Fähigkeit, Angriffen zu widerstehen und gleichzeitig ihre Geschäfte weiterzuführen.

Praktiken zur Verbesserung der Cyber-Resilienz können entweder als *präventiv*, *detektivisch* oder *reaktiv* kategorisiert werden. Die Prävention ist eine wichtige Komponente, da sie die Sicherheitsbemühungen in ein Stadium verlagert, in dem Änderungen kostengünstiger umzusetzen sind. Es werden überwachende Massnahmen, die anerkennen, dass vollständige Prävention nicht realisierbar ist. Solche Massnahmen verkürzen die Reaktionszeit und minimieren die Auswirkungen eines Angriffs. Eine systematisch durchgeführte Reaktion hilft einer Organisation, den Schaden zu begrenzen und den Betrieb wieder in einen operativen Zustand zu bringen.

Der Aufbau eines Cybersicherheitsprogramms mit präventiven, detektiven und reaktiven Massnahmen erfordert zahlreiche Praktiken, Modelle und Lösungen anzuwenden. Eine risikobasierte Perspektive verzichtet auf einen pauschalen Ansatz und konzentriert sich auf die für Angriffe anfälligen Unternehmenswerte. Der konzeptionelle Rahmen für die Durchführung eines Cyber-Risiko-Programms ist etabliert, und es gibt zahlreiche Methoden, Rahmenwerke und Standards. Die Umsetzung solcher Verfahren ist jedoch eine Herausforderung, und in der Praxis werden zahlreiche Einschränkungen beobachtet.

Diese Arbeit betrachtet diese genannten Herausforderungen innerhalb eines risikobasierten Sicherheitsprogramms als Forschungsmöglichkeiten. Durch eine Reihe von Designstudien werden neuartige Lösungen präsentiert und im Kontext spezifischer Herausforderungen bei Prävention, Erkennung und Reaktion bewertet. Für die Risikoprävention nutzen die vorgeschlagenen Lösungen zur

Bedrohungsmodellierung die visuelle Architekturmodellierung, um die Rolle der Kollaboration, die Bedeutung von Insider-Angriffen und das Aufkommen von KI-bezogenen Angriffen zu berücksichtigen. Darüber hinaus wird das Problem der Quantifizierung durch Lösungen angegangen, die Konzepte aus der Kontinuitätsmodellierung nutzen. Die Ergebnisse deuten auf eine verbesserte Effektivität und Effizienz in diesen Problembereichen hin, wobei gleichzeitig auf die zugrundeliegenden Beschränkungen hingewiesen wird, wie z. B. der Mangel an verfügbaren Daten oder die Unmöglichkeit, eine vollständige Automatisierung zu erreichen. Erkennung und Reaktion werden als eng miteinander verbundene Verfahren betrachtet. Es werden neuartige Lösungen vorgeschlagen, um Angriffsdaten zu sammeln und sie im Rahmen der proaktiven und reaktiven Verteidigung gegen verschiedene Bedrohungen zu nutzen. Die an den entiwckelten Lösungen durchgeführten Tests bestätigen die Realisierbarkeit beider Verteidigungsparadigmen, wobei die Resultate eng an die spezifischen Angriffs- und Verteidigungsszenarien gekoppelt sind.

Acknowledgments

The Road goes ever on and on,
Down from the door where it began.

— J. R. R. Tolkien (1954)

This document is likely the only one produced throughout this degree that sees only a single author on its cover page, seemingly giving the impression that it is the contribution of a single person. On the contrary, this work would not have been possible without the tremendous support of my surroundings. I would especially like to thank the following people:
Thanks to Prof. Dr. Burkhard Stiller, who supported and guided my studies since the B.Sc. in 2017.
Thanks to Prof. Dr. Abhishta Abhishta for supporting this thesis with his expertise.
Thanks to Prof. Dr. Muriel Franco for sparking my initial interest in research and sharing his methodological expertise.
Thanks to Prof. Dr. Alberto Huertas Celdrán, who was not only a friend but a dedicated supervisor and research collaborator over multiple years.
Thanks to Dr. Christian Killer, who took the role of a reliable and loyal friend the first time we met.
Thanks to Prof. Dr. Eder Scheid, who shaped this thesis from the first sentence of my first publication.
Thanks to Chao Feng, whose kind nature accompanied me until the end of this journey.
Thanks to all collaborators and partners in scientific projects – especially the Cyber-Defence Campus – for the interesting discussions and support provided.
Thanks to the 40 students who spent half a year of their lives working on a thesis under my supervision, I can genuinely say that I learned something from each of them.
Thanks to all members of the Communication Systems Group for their openness to collaborate and share their expertise.

Thanks to Isabel, Angelika, and Marcel for supporting me and my education in many aspects.
Thanks to my friends, who always enable me to get a fresh perspective on all matters.
Thanks to my wife, Alma; having you by my side was the most valuable thing to me.

Contents

1 Introduction **3**

1.1 Problem Statement . . . 6
1.2 Research Questions . . . 7
1.3 Thesis Contributions . . . 9
1.4 Outline . . . 10

2 Fundamentals and Definitions **13**

2.1 Cybersecurity Risk . . . 13
2.1.1 Terminology . . . 14
2.1.2 Risk Factors . . . 15
2.1.3 Standards and Frameworks to Manage Risk . . . 17
2.1.4 Threat Modeling and Secure System Design . . . 20
2.1.5 Threat Modeling Methodologies . . . 21
2.1.6 Representations . . . 24
2.1.7 Tools . . . 25
2.2 Preventing Threats, Detecting Attacks, and Responding to Breaches . . . 27
2.2.1 Prevention . . . 27
2.2.2 Detection . . . 28
2.2.3 Response . . . 30
2.3 The Cyber Threat Landscape . . . 32
2.3.1 Threats and Risks . . . 32
2.3.2 Problem space Domains . . . 38
2.4 Research Foundations and Methodologies . . . 48
2.4.1 Design Science Research . . . 48
2.4.2 Research Foundations . . . 51

2.4.3 Research Methods 60

3 Related Work **67**
3.1 Threat Modeling 68
3.1.1 Threat Modeling Methodologies 68
3.1.2 Threat Modeling Tools 70
3.1.3 Collaboration 74
3.1.4 AI-based Systems 77
3.1.5 Insider Threats 80
3.1.6 Threat Quantification 83
3.2 Threat Detection and Mitigation 85
3.2.1 Malware Data Collection 85
3.2.2 Reactive Malware Defense and Offense 88
3.2.3 Proactive Defense 93
3.2.4 Limitations and Opportunities 95

4 Modeling and Prioritizing Threats **97**
4.1 Collaborative Threat Modeling 97
4.1.1 Design and Architecture 98
4.1.2 Prototypical Implementation 103
4.1.3 Evaluation 105
4.1.4 Key Observations and Discussion 109
4.2 Modeling Insider Threats 110
4.2.1 Approach and Architecture 110
4.2.2 Web-based Implementation 112
4.2.3 Evaluations 113
4.2.4 Key Observations and Discussion 119
4.3 Quantifying Threats 120
4.3.1 The *QuantTM* Approach 121
4.3.2 Prototypical Implementation 126
4.3.3 Evaluations 133
4.3.4 Expert-based Experiments 143
4.3.5 Key Observations and Discussion 145
4.4 Modeling and Quantifying Threats to AI 147
4.4.1 The *ThreatFinderAI* Approach 148

4.4.2 Evaluations . . . 154
4.4.3 Key Observations and Discussion . . . 166

5 Detecting and Mitigating Threats **169**
5.1 Lightweight Moving Target Defense . . . 170
5.1.1 Framework Architecture . . . 171
5.1.2 Evaluations . . . 175
5.1.3 Key Observations and Discussion . . . 182
5.2 Rule-based Reactive Moving Target Defense . . . 183
5.2.1 The *R-MTD* Approach . . . 183
5.2.2 Prototypical Implementation . . . 185
5.2.3 Evaluations . . . 188
5.2.4 Key Observations and Discussion . . . 191
5.3 Obtaining Data Through Dynamic Execution . . . 193
5.3.1 Architecture and Implementation . . . 193
5.3.2 Evaluations . . . 197
5.3.3 Key Observations and Discussion . . . 204
5.4 File System Defense . . . 205
5.4.1 File System and Testbed Architecture . . . 206
5.4.2 Proactive Techniques . . . 208
5.4.3 Reactive Techniques . . . 208
5.4.4 Evaluations . . . 212
5.4.5 Key Observations and Discussion . . . 219
5.5 Conceptual Integration . . . 220

6 Deployments and Pilots **221**
6.1 Adopting Secure Development Practices . . . 222
6.2 Service-based Threat Modeling . . . 225

7 Summary and Future Research **229**
7.1 Summary . . . 230
7.2 Research Questions Revisited . . . 232
7.3 Limitations and Avenues for Future Research . . . 235

References **260**

A Publications **261**
A.1 Contributing Manuscripts 261
A.2 Related Manuscripts 264

B Keyword Search in Google Scholar **269**

List of Figures **270**

List of Listings **274**

List of Tables **276**

Curriculum Vitae **279**

Most companies live below the "security poverty line".

— Travis McPeak (2024)

1 Introduction

Cybersecurity has received widespread attention from various angles. On the level of the European Union, the *Network and Information Security Directive* (NIS2) directive requires member states to harmonize and ensure the cybersecurity of critical services, warranting execution until October 2024 [74]. Almost at the same time, the United States Securities and Exchange Commission implemented new regulations mandating the disclosure of cybersecurity incidents [261]. Evidently, governing bodies and regulators have acknowledged the importance of cybersecurity. Enterprises have met this trend by increasing their cybersecurity budget. For example, one study revealed that cybersecurity spending increased by 70% from 2019 to 2023 [173]. Many of these factors likely contribute to the continuous growth of the cybersecurity market and worldwide end-user spending of 214,953 billion USD across segments [96]. Figure 1.1 emphasizes the increasing attention given to the topic, as highlighted through the thousands of doc-

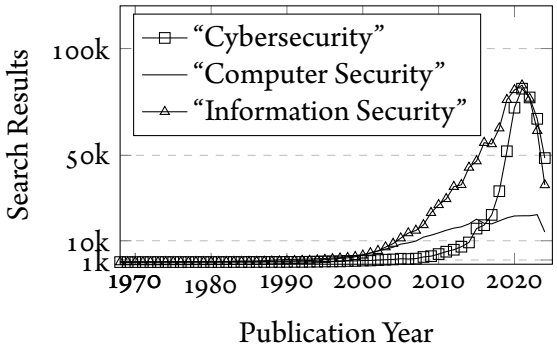

Figure 1.1: Cybersecurity Publications

uments (*e.g.*, publications, books) on the topic. Thus, cybersecurity is an unignored aspect of today's society. As such, protecting information during computerized processing is subsumed into an age-old problem. For example, the Roman historian [94] describes a practice of obscuring messages before transmitting them to the senate. The algorithm, which involved the shifting of letters to ensure secrecy, has later become known as the *caesar cipher* encryption technique.

The role of cybersecurity as an operational risk [120] to information systems has stimulated not only the IT security market but also the research community. Based on a keyword search in the database for academic literature Scopus, the information security research activity is plotted in Figure 1.2. As can be inferred, the first scientific publications on cybersecurity were published in the late 1970s, with an increased number of publications appearing after the year 2000. Furthermore, there have been nearly exponential studies in this area in recent years.

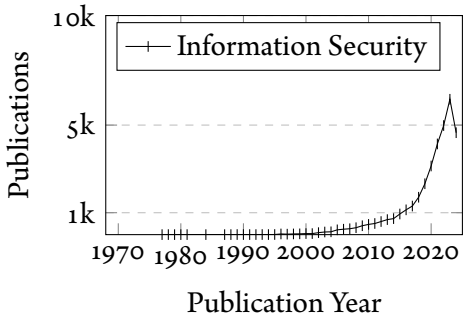

Figure 1.2: Academic Publications

Based on these trends, one could assume that the increased attention, discussion, and investment in security-related topics would have led to cybersecurity becoming a well-controlled risk. However, data from the industry points in a different direction. On average, a data breach incurs losses of 4.88M USD – a 10% increase over the previous year and the highest according to 19 consecutive analyses performed by [125]. Even in non-commercial settings, cyberattacks are a key risk that individuals, organizations, and governments must address. For example, the *World Economic Forum* (WEF) [290] contextualizes cyber attacks as a risk to democratic elections.

Different philosophies exist to combat cyber attacks. Maturity-based approaches postulate that organizations should achieve certain pre-defined capabilities. Alternatively, the risk-based abstraction framework provides a language to discuss why and how a particular information security program can fail: Potential threats can materialize into attacks that exploit systematic weaknesses in systems that are covered by inadequate controls of system assets – leading to negative impacts for stakeholders [137]. Unfortunately, an information security program can be deficient in any of these aspects. First, an organization may be unaware of the potential attacks (*i.e.*, *threats*) that an adversary may execute. For example, ransomware comprises several variants to execute software-based attacks to incur damage and request the victim to pay a ransom for recovery. In *crypto-ransomware*, a subset of files is made unavailable through encryption. An organization may decide to pay for the ransom, potentially resulting in a decryption key [6]. However, this victim may be unaware that the attacker may have exfiltrated that data – enabling a second round of extortion under the looming threat of publishing the

data [9]. Secondly, organizations may be aware of a threat, but not about the specific system flaw (*i.e.*, *vulnerability*) to be exploited. For example, *Heartbleed* was a flaw within the most popular *Transport Layer Security* (TLS) library. The vulnerability was not an implementation and not a protocol design flaw, and remained undetected for years [141]. Thirdly, an organization may secure an asset with inadequate controls. For example, sensitive user information of [140] was obtained after the user database was exfiltrated. Although the company was likely aware of an exfiltration threat, the user passwords were only secured through a weak hashing mechanism. Researchers demonstrated that thousands of these passwords can be obtained by locally executing password cracking software [140]. Thus, if these fundamental aspects are not carefully aligned, the overall information security strategy may be ineffective.

In brief, securing information is non-trivial since many relevant assets must be known, threats and weaknesses identified, and effective controls implemented. Furthermore, non-technical and economic aspects (*e.g.*, information asymmetries, lacking incentives) present additional burdens [175]. As such, an organization may unlikely reach "perfect protection". Instead, organizations may strive for cyber resilience, as defined by [27]: "(..) refers to the ability to continuously deliver the intended outcome despite adverse cyber events." From this definition, multiple principles are derived. First, information systems must keep business goals – and not technical goals – intact. In addition, information security shall be addressed by the organization as a whole instead of a dedicated unit. Architecturally, multiple control mechanisms should enhance this resilience, achieving both protection and recovery. In modern ERM, many of those principles may be considered [120]. However, when looking at them from a technical, systems-engineering perspective, the key activities of a risk management program (*cf.*, Figure 1.3) are faced with specific challenges (*i.e.*, identifying emerging attacks, modeling system assets). These challenges are the focus of this PhD thesis.

Figure 1.3: Macro-level Risk Management Steps Defined by NIST [185]

1.1 Problem Statement

Risk refers to the uncertain deviation from an expected outcome (*cf.*, Section 2.1.1). Throughout this thesis, a risk-based perspective is adopted to foster the cyber resilience of organizations, governments, and individuals. However, although this thesis follows a risk-based viewpoint, such as the one followed in *Enterprise Risk Management* (ERM), this perspective is taken from a technical system's engineering position. For example, while ERM focuses on strategic risks to an enterprise, the system's engineering context applied throughout this thesis may involve the management of more granular, risk-based, system-centric decision-making. This involves numerous decisions with varying timelines (*e.g.*, a developer deciding which software library to depend on to implement a system component or an architect deciding to add an additional security control). Thus, different problems are encountered across several activities shown in Figure 1.3, which are subsequently summarized.

To *identify risks* and protect assets through threat modeling, academia and industry agree on several problems. First, this activity is often carried out in an unsystematic, ad-hoc manner. It may be conducted without representing non-technical aspects due to the fact that only a small number of security experts are involved [253]. As such, it is argued that more varied viewpoints should be adopted. However, this challenge may oppose another critical problem of threat modeling: due to the complexity of assets and threats being modeled by subject-matter and domain experts, the overall activity carries substantial cost [98]. These two problems are further complicated by new technological or environmental trends. Hence, novel methods should address domain-oriented challenges and guide and include cross-functional involvement while providing practical automation to reduce costs.

Quantifying cyber threats is inherently challenging since most threat modeling tools, methods, and executions follow a *qualitative* approach. In addition, many approaches encompass neither a quantification nor qualification – or do so in an unsystematic manner [273]. This may contribute to the overall "art" of threat modeling [108] being viewed as non-systematic since an outsider might not understand how decisions were made. As such, they may also only intrinsically be driven by business value – lacking explicit evidence that the output of such a qualification is well-aligned with the strategic or procedural business goals. For example, approaches use technical metrics, such as the number of system interfaces, to denote its sensitivity against threats [273]. This challenge underlines the observation that threat modeling is executed in an isolated, technology-focused manner.

Two questions must be addressed for *attack detection*: How is data gathered, and how is that data used for classifying behavior? A threat-oriented data collection approach would require obtaining data that captures the threat's behavior while not overlapping with benign scenarios. Signature-based approaches are proliferated, which are neither generic enough to capture classes of threats nor robust

enough to outlive their polymorphic nature [272]. However, obtaining behavioral data is challenging [47], requiring novel approaches to simplify and secure data collection. To demonstrate the value of that data, the resulting models would need to be tested in several configurations (*e.g.*, against seen and unseen threats, with different levels of benign behavior). Practically, the body of research must be enriched by realistic deployments. Furthermore, attack detection is only useful if it enables effective mitigation. Hence, challenges at the intersection between detection and mitigation arise.

In practice, no preventive control measure is perfect, necessitating several layers of defense. For *response and recovery*, several limitations exist. First, the possibility of developing dynamic controls against classes of threats is not well-established – experiments and real-world deployments would be needed to demonstrate that it is possible to mitigate not just a specific implementation of an attack vector [267, 272]. Proactive defense methods suffer from the limitation that they may not be deployed at the right time or that they waste unnecessary resources in the absence of an attack [268]. On the other hand, reactive approaches are limited by their detection systems, such as wasting resources due to their continuous operation and reacting to false alarms [267].

This PhD thesis addresses the limitations of costly, siloed, and qualitative threat modeling. It encompasses challenges such as establishing behavioral threat detection through data collection and classification. Finally, problems such as the effectiveness and efficiency of attack defense are approximated through the proposal of threat-oriented defenses in both proactive and reactive scenarios.

1.2 Research Questions

In this thesis, the challenges (*e.g.*, identifying or quantifying threats) of several areas in cybersecurity (*cf.*, Chapter 1.1) are analyzed, and new solutions to improve their execution are proposed, implemented, and evaluated. To determine the study design of the selected areas, a more granular view of this PhD thesis' scope is expressed through the following research questions.

RQ1) Which limitations are faced when modeling cybersecurity threats, and how can they be overcome through novel methods and tooling?

RQ2) Is it possible to formulate quantitative risk-based representations of threats, and how sensible–considering expressiveness and usability–is it to do so in the context of threat modeling?

RQ3) What novel ways exist to extract cyberattack data, and how useful is that data to represent threat abstractions?

RQ4) How effective and efficient is it to mitigate cyberattacks by developing proactive and reactive mitigation strategies that tackle families of threats?

RQ1 forms the core of the thesis, as the underlying abstraction of threat modeling is used for all the other solutions across activities (*e.g.*, solutions on *RQ4)* aim to mitigate classes of threats). To understand the research gaps, limitations are surveyed. This is achieved by performing a semi-systematic survey of the literature. Aside from descriptively establishing the state of threat modeling in terms of tools and methods, practical insights and learnings are collected, primarily focusing on the role of industry practitioners. Furthermore, observations from the author's involvement in research projects (*e.g.*, applying threat modeling in the public sector or securing the development of a data sharing platform for the healthcare sector) serve as problem instances to establish problem constructs (*e.g.*, *how well do currently available threat modeling tools support the inclusion of a remote participant?*). These theoretical findings are enriched by an industry analysis and performing dynamic analysis against the previous questions. The resulting limitations are then used as an input for several design studies that propose novel methods and tooling. Evidence of their effectiveness is then gathered through several methods, such as case studies, field experiments, and illustrations.

To study *RQ2*, a literature review of the threat modeling methods and tools with respect to their resulting numerical output is performed. Additionally, dynamic analysis and a questionnaire-based vendor inquiry are performed. Due to a lack of quantitative methods, the field of cyber risk management is surveyed – discussing how to possibly implement quantitative threat analysis. Generic and domain-specific threat quantification methods and tools are developed. These are analyzed through focus group experiments and case studies, illustrating the usability of the methods and the expressiveness of their results. *RQ3* is guided by the ML development pipeline. For the data collection phase in the model, a literature review is performed, highlighting the research gaps, which are addressed through the development of frameworks. Their evaluation comprises both quantitative (*e.g.*, simulations, performance analyses) and qualitative evaluations (*e.g.*, usability tests, case studies), indicating the efficacy of these monitoring approaches. In *RQ4*, the moving target defense paradigm is adopted as a framing concept. Here, two research lines are followed. First, regarding proactive defense, approaches that do not rely on a signal are identified. In contrast, studies presenting reactive defenses are surveyed based on their usage of an "intelligent" defense signal. In this category, the trends observed from *RQ3*, such as behavioral fingerprinting or ML-based detection, are used to elicit research directions. Several proactive and reactive defense frameworks are then developed. Their evaluation considers the effectiveness and efficiency, which is established from defensive and offensive viewpoints. To do so, several experimental methods are used, such as simulations, performance analyses, or quantitative model performance testing.

1.3 Thesis Contributions

This thesis contributes to the growing body of cybersecurity research. Taking a holistic perspective on the field while considering both technical and non-technical problems yields advancements in the following areas of cybersecurity:

Threat Modeling is a well-estbalished discipline within cybersecurity. Herein, several limitations (*e.g.*, cross-functional, domain-specific modeling) are identified based on the literature, practical experiments, and interactions with industry practitioners. Consecutively, a set of practical approaches and solutions is proposed and prototypically implemented. These solutions are deployed and evaluated in different settings – including illustrative scenarios, case studies, and user experiments.

Threat Quantification lies at the intersection of practical threat modeling and strategic risk management, focusing on the lack of quantitative methods in threat modeling. Due to the novelty of the field, a generic, quantitative threat assessment method and model are proposed. Prototypical implementations are evaluated within a Swiss SME, an expert-based field experiment, yielding discussions on the applicability of this method from the threat modeling perspective. These generic approaches are instantiated for specific problem domains (*i.e.*, ML and LLM security).

Threat Detection focuses on detecting cyberattacks while relying on the abstraction of cyber threats. Thus, this abstraction is mapped to the metamodel of classification and anomaly detection. This category presents novel approaches to gathering cyber attack data and using it to classify attack families, instances, or abnormal behavior. The resulting prototypes are assessed based on their effectiveness and efficiency in experimental settings, presenting evidence of the usefulness of detecting attacks based on an abstraction of threats.

Threat Mitigation comprises work on either proactively preventing or reactively responding to cyber threats. Defense concepts that fit under this abstraction, such as attack deception and *Moving Target Defense* (MTD), are explored, and new strategies are proposed and tested.

Deployments and Pilots contrast the previous approaches, which are strictly executed in an academic research setting. As such, the previously described contributions of this thesis have been deployed and tested in practical settings, presenting additional insight into their practicality in real-life settings.

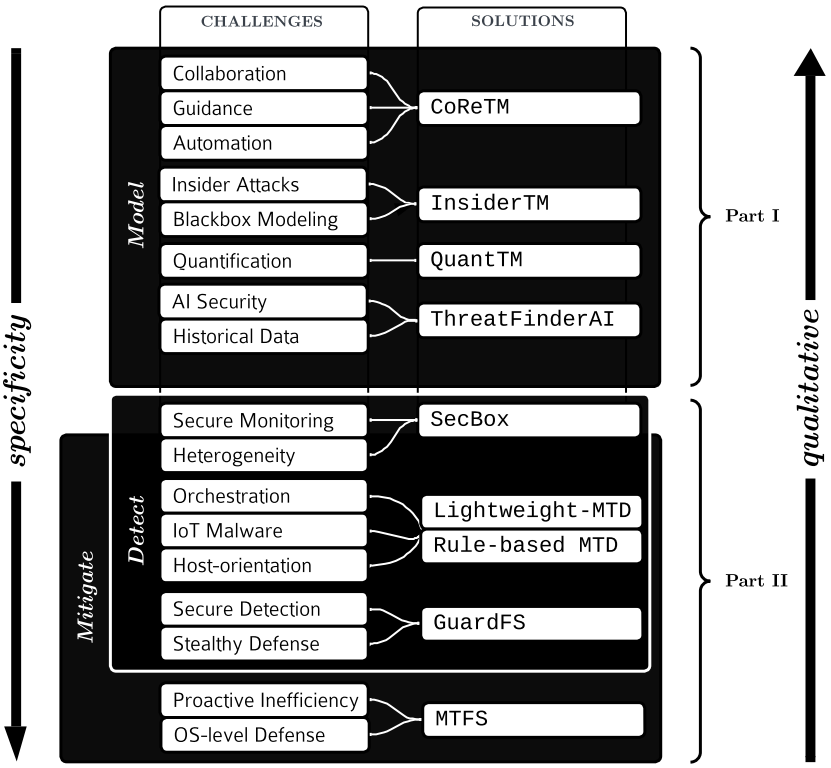

Figure 1.4: Solutions Contributed per Problem Area

These contributions address four areas of cybersecurity: threat modeling and prioritization, attack prevention, detection, and response. Figure 1.4 summarizes the contributions and contextualizes them according to problems and solutions. As indicated, the solutions are presented in two chapters – the latter solutions exhibit higher specificity (*i.e.*, the proximity between solutions and problem scenario of each study) and use quantitative methods.

1.4 Outline

This PhD thesis is structured into seven chapters, with the first chapter introducing the problem statement and outlining the scope of the work performed. **Chapter 2** introduces the reader to important concepts upon which this work is based. The cybersecurity field is analyzed horizontally (*i.e.*, in terms of different activities) and vertically (*i.e.*, in terms of technological trends and domains). Here, the key terms of the risk management abstraction model, followed by practical, risk-based approaches, are introduced. Furthermore, technological concepts such as Artificial Intelligence or the Internet-of-Things and security domains, including e-government, are described. Finally, methodological de-

tails are described by iterating through the key methodologies employed throughout the thesis. Each academic study begins with a literature review to understand the existing state of the art. Additional theoretical aspects are presented to the reader in **Chapter 3** – introducing the methodology and synthesized findings of several literature reviews. First, the state of research concerning threat modeling is analyzed, highlighting trends, limitations, and future directions for research. As a result, threat quantification is investigated by different disciplines. Similarly, existing approaches in threat detection are presented, focusing on threat data collection and detection, and certain studies are shared with those presented in threat response.

The following two chapters then present the main body of novel work – each presenting an overview of the solutions proposed for the areas of threat modeling, detection, and mitigation. Within each chapter, an introduction to the approach and selected evaluations is provided. Since this thesis does not contribute a new and overarching framework, the design science conceptual model (*cf.*, Section 2.4) is used to introduce how the solutions relate. **Chapter 4** presents the solutions *CoReTM, QuantTM, InsideTM* and *ThreatFinderAI* for threat modeling and quantification. With respect to attack detection, **Chapter 5** introduces four solutions (*i.e., SecBox, Lightweight MTD, Rule-based MTD, GuardFS*) to collect, use, and evaluate threat data from offensive and defensive viewpoints. In addition, *MTFS*, presents a solution on how to defend attacks from a completely proactive (*i.e.*, without any detection) viewpoint.

Based on the evidence presented in these studies, **Chapter 6** presents findings on the solutions' practical deployments outside of a purely academic context, highlighting additional strengths, weaknesses, and future directions. Finally, a summary of the work and its contributions and potential future work are described in **Chapter 7**, which concludes this PhD thesis.

The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards – and even then I have my doubts.

— Eugene Howard Spafford (1989)

2 Fundamentals and Definitions

This thesis takes a broad view on cybersecurity, demanding an introduction to several concepts, activities, and domains. In the following, the reader is acquainted with the concept of information security risk. Under this umbrella, several information security management approaches and their underlying methods are presented. Environmental concerns about current *Information Systems* (IS) are introduced, such as disruptive technologies, including the *Internet-of-Things* (IoT) and *Artificial Intelligence* (AI). Finally, the chapter summarizes the methodological pillars underpinning the studies presented in this thesis.

2.1 Cybersecurity Risk

Risk is intrinsically understood as the uncertainty surrounding a future outcome. As such, risk management deals with managing complex and dynamic factors that influence future outcomes in organizations. Historically, risk has been managed with a particular focus on financial risks. Later on, risk management gained breadth and depth, leading to holistic risk management. As such, other operational risks have been considered, too [120].

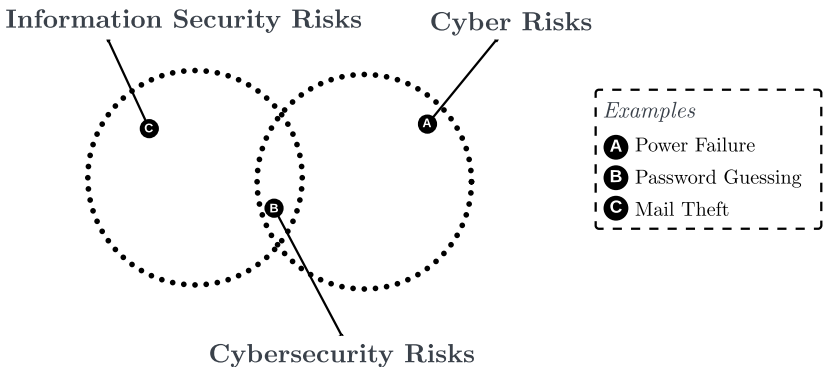

Figure 2.1: Relation of Key Terms

New methods and perspectives have shaped this paradigm. For example, a holistic perspective on risks mandates that not only adverse risks (*e.g.*, financial losses, danger to human life) but also positive scenarios are considered (*e.g.*, reduced sourcing cost, lower pricing due to currency inflation) [120]. Cybersecurity risk is defined as a purely negative operational risk. As such, cybersecurity risks can adversely affect business goals through financial losses, operational effectiveness, or reputation [29].

2.1.1 Terminology

So far, this thesis has used several terms to denote risks stemming from attacks launched on digital information systems. As such, the usage of terms resembled the disharmonious usage discovered in the literature (*cf.*, Figure 1.1). The Figure indicates the number of publications since 1970 – including academic publications, industry standards, and reports – discovered for three search terms in Google Scholar (*cf.*, Appendix B). Although being only the result of an illustrative search, the trends correspond to what is described in the literature of the field: Before the turn of the millennium, the term "information security" gained in popularity, reflecting that information has become a key asset to be protected. In recent years, "cyber space" has been used more or less synonymously for internet-enabled application domains. Hence, the term "cybersecurity" has been used for security concerns within that scope [122]. The terms relate as indicated in Figure 2.1. In this thesis, it is argued that, while established in the literature as such, there are no "pure" cybersecurity risks since it is unlikely that such attacks can be characterized with purely technical characteristics without incorporating the surrounding socio-economic information system. As such, this thesis does not distinguish between cybersecurity and information security risks. Hence, the terms are used interchangeably for the remainder of this thesis. However, pure "cyber risks" (*e.g.*, power outages, hard drive malfunctions) are not specifically considered since these describe adverse effects originating from digital environments but not relating to a malicious intent or impact.

Within information security management, the abstract notion of "risk" is often decomposed into additional concepts. The practical decomposition of a particular scenario into those aspects yields some approaches (*e.g.*, threat identification, risk analysis) that will be introduced in the following chapter. Figure 2.2 outlines the relation of the different factors that influence risk.

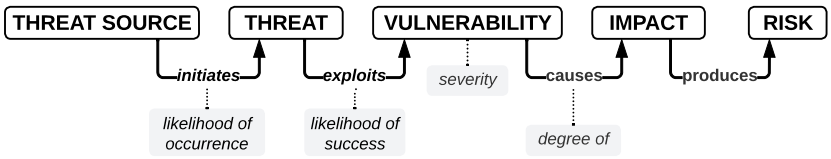

Figure 2.2: Risk Factors in the Risk Model defined by NIST [138]

In addition to this domain-specific terminology, several terms used by the research community require a clear definition since their usage is not necessarily aligned across different domains. For example, [185] introduces a risk management framework comprised of several steps, while [267] proposes an information system's architecture under this term. In this thesis, the term *framework* refers to its meaning in the computer science field. Here, the term is used to describe components that are needed to support or achieve a methodology. Commonly, it refers to layered structures making up software components or to the relationships of programs [84, 184]. In non-technical fields, a framework commonly refers to a structured process to achieve a goal [84], which can also be seen by the definition provided by Merriam-Webster, outlining two possible forms, either a concept structure of ideas or a structural frame [165]. In this thesis, this procedural view is used under the term *methodology*, *i.e.*, a set of methods used in a discipline [84, 167]. For example, the *NIST Risk Management Framework* is considered a methodology in this work, as it outlines a series of activities to manage risk in an organization [137]. *Methods* are thereby the sub-components of methodologies. They provide systematic procedures, techniques, or modes for inquiry [166]. Finally, *models* are defined as abstract constructs facilitating discussion of real-world problems and solutions [113].

2.1.2 Risk Factors

Threats are potential events or states of damage [232]. Several examples of threats can be found in humans' daily lives. Threats do not arise out of thin air. Instead, they are caused by threat sources, which characterize how the threat event could be materialized. Threat sources can include several dimensions, such as the intent or the method applied, as well as their type, including hostile attacks, human errors, structural weaknesses, or natural accidents. Aiming to enumerate all threat events may

be infeasible for an organization; hence, the usage of models simplifies this process by focusing on particular tactics, motivations, and adversarial capabilities (*i.e.*, hostile threat sources) [138].

The notion of threats is tightly coupled to the concept of *vulnerabilities* – any discussion on earthquakes likely leads to a discussion of structural weaknesses of buildings. In the information security context, a vulnerability refers to a weakness within an information system that enables exploitation by a threat source. As will be argued in this thesis, a vulnerability may relate not only to the information itself but to the procedures and organizations embodying that system. The origin of a vulnerability may stem from a lack of control or the inefficacy of the control mechanism to provide a certain function [138]. For example, the administrative portal of an application may not be secured with a password, as the developers assumed that the URL is never displayed in the UI. Alternatively, an inadequate control can involve a password that consists of only four numbers. An attacker attempting one random password per second would have exploited the system within a few hours.

Previously, the view of cyber risks in a purely technical scope was questioned. Vulnerabilities may not only exist in the technical sense (*i.e.*, structural vulnerabilities in a system). Instead, they can exist on all organizational levels [138]. For example, an information security strategy may not exist, making the organization as a whole vulnerable. Similarly, they may exist on a procedural level, *e.g.*, when a manufacturing process depends on a single supplier. This definition of a vulnerability also subsumes the definition of a *security control*, which can be defined as an (additional) safeguard to satisfy a particular requirement [136]. Threats and vulnerabilities would not receive any attention if their materialization did not lead to a particular *impact* on individuals, enterprises, or nations. Impact can thus be defined as the type of harm and its magnitude resulting from the threat's materialization. Within cybersecurity, the result is an undesired disclosure, modification, or unavailability of information directly or indirectly impacting stakeholders associated with that information [138].

These concepts suffice to describe an attack. For example, a nation-state actor could guess the password (*i.e.*, threat event and source) of a user with a weak password (*i.e.*, vulnerability), thereby impacting data confidentiality (*i.e.*, impact). However, this example constitutes an exemplary attack – and not a risk. Hence, *risk* is not characterized as a materialized incident but rather as the potential for these concepts to occur together. The differentiating factor here is the existence of *uncertainty*, which, unfortunately, applies to all previously introduced factors. The total number of relevant threats and their actors may be just as uncertain as a system's complete set of weaknesses. Similarly, the magnitude of a threat's impact and its likelihood to occur and to succeed is uncertain [138]. Various sources will characterize risk as $risk = threat \times vulnerability$ or similarly as $risk = impact \times likelihood$; however, this may convey a simplistic relation of risks, as it does not enable the communication of uncertainty if it is interpreted numerically without introducing advanced statistical concepts [119, 120, 206, 289].

2.1.3 Standards and Frameworks to Manage Risk

So far, the concept of risk within information security and its influencing factors have been introduced. However, this does not yet explain why one should care about applying this mental model. It is therefore necessary to motivate how these concepts can be used to create *value*. Traditionally, the management of information security may have been perceived as a business inhibitor. However, the goal of a modern *Information Security Risk Management* (ISRM) program is to assess an organization's risk exposure and align it with the amount of risk it is willing to take to operate its business [287]. In the strategic definition used by *Enterprise Risk Management* (ERM), risk management generates value for the company by improving the quality of decision-making beyond the cost incurred through risk management [120]. This contradicts the common understanding of risk management as a practice to strictly minimize risks. After all, these practices should maximize the tradeoff between increased quality of decision-making and cost (either direct or opportunistic). In information security, this is easily exemplified by the quote at the beginning of this chapter – a system's attack surface (and hence the risk exposure) can be reduced by removing the number of network interfaces, data points collected, and so on. However, decreasing risk exposure by reducing the system's functionality implicitly reduces the value of the system. Knowing that undertaking any business requires risk, the goal of risk management is to provide solid foundations for optimal decision-making when faced with risks. In summary, value is created by assessing the risks and balancing the number of countermeasures to be introduced.

Whereas traditional information security management has focused on achieving "best practices", risk-oriented approaches abandon the assumption of a one-size-fits-all strategy to secure information systems. To do so, several frameworks, standards, and activities have been developed – which are iteratively introduced in the following sections.

Several risk management frameworks[1] have emerged and adopted in practice [120]. In ERM, notable frameworks have been published by the *Committee of Sponsoring Organizations of the Treadway Commission* (COSO) or the *International Organization for Standardization* (ISO). These frameworks do not focus exclusively on information security. Instead, they holistically establish risk management through activities, organizational culture, and competencies [61, 120, 130]. Multiple frameworks consider risk management within information systems. An assessment-based approach for information security risks is embodied in the "ISO 27005:2022" standard [129] while relating to the over-

[1] In this particular context, the term "framework" denotes a series of activities to achieve an organizational goal. If not stated explicitly, the remainder of this thesis uses the term "framework" in its technical context, denoting a set of architectural system components.

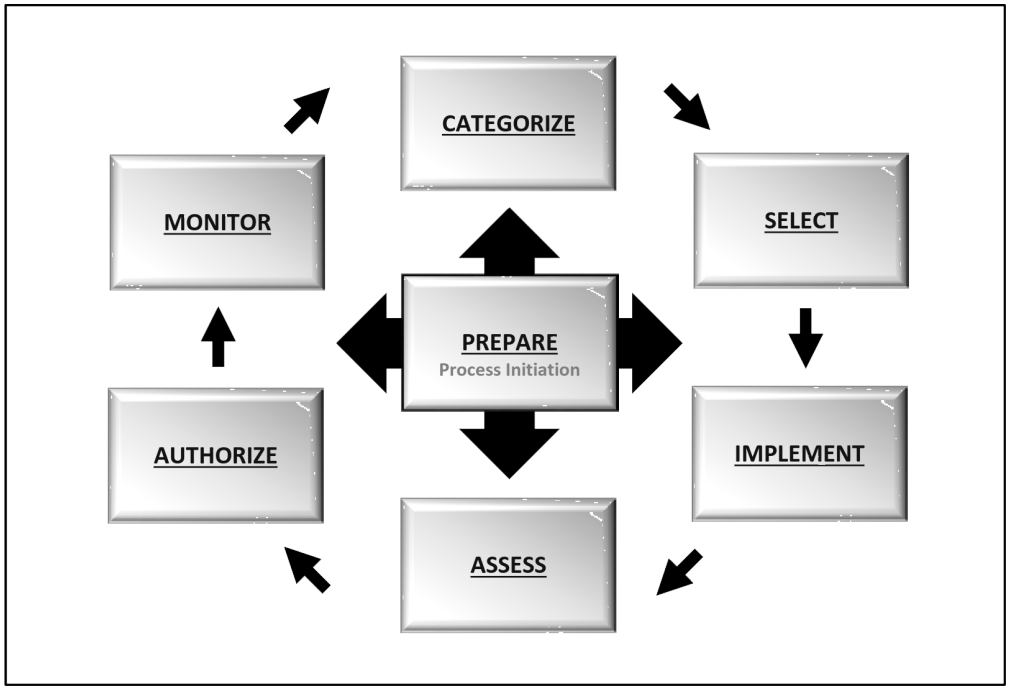

Figure 2.3: Seven Step Risk Management Steps in NIST SP 800-37 [137]

all approach defined by the "ISO 31000:2018" standard [130] as well as "ISO 27001:2022" [131]. Within German-speaking areas, the "BSI Standard 200-3" is followed by organizations. Although not purely a risk management framework, the overarching "IT-GrundSchutz Kompendium" presents activities to assess and manage risks [35, 36]. Similar to the ISO standard, the National Institute of Standards and Technology presents a risk management framework through its special publications. "NIST SP 800-37" presents both the terminology and procedures to manage risk in an organization [137]. Due to its technology-neutral viewpoint and open accessibility, this standard is further introduced to exemplify the information security risk management procedures. NIST summarizes activities for three different areas of an organization (*cf.*, Figure 2.4) under seven steps, as indicated in Figure 2.3:

Figure 2.4: Organizational Risk Management [137]

1. The organization is prepared for the risk management program; roles are assigned, a strategy is developed, and an initial risk assessment is conducted and evaluated. On the system level, stakeholders, assets, functions, and their architecture are defined.

2. Assets, individuals, and organizations are categorized, reflecting the organization's information security risk strategy.

3. Security controls are selected, adapted, planned for, documented, and approved.

4. Security controls are designed, implemented, and documented.

5. The implementation is evaluated by planning, executing, reporting, and acting on a previously performed assessment.

6. The risk exposure is contrasted against the organization's risk tolerance; authorization of the resulting delta by the official.

7. The underlying risk factors are monitored (*e.g.*, environmental changes, control outputs) continuously to detect whether the influencing assumptions hold.

In addition to these industry standards, several other risk management frameworks have been proposed. The CORAS methodology provides a diagram-based risk modeling language and a supporting tool [155, 234]. It assigns roles within the process (*e.g.*, analysis leader, analysis secretary). The target audience consists of customer representatives, subject-matter experts, consultants, and end users, who are guided through eight steps. One of the critical steps in CORAS is the actual identification of threats, which is achieved using threat diagrams that aid in brainstorming and scenario analysis [155].

Similarly, the *Factor Analysis of Information RISK* (FAIR) framework uses a high-level ontology of risk factors – comparable to the ones described by NIST in Figure 2.2. By decomposing a risk scenario into the ontological factors (*e.g.*, contact frequency of a threat, probability of action), a quantitative assessment of the risk scenario is envisioned. Again, threat identification is a critical component of the framework [93]. The same can be observed in other ISRM frameworks, such as OCTAVE Allegro [43], RISK IT [128], and FRAAP [203]. For the scope of this thesis, it can be assumed that the key steps and abstractions are defined by those frameworks. However, these steps are formulated in a descriptive, generic, and flexible manner. For example, [137] mandates as part of the *risk assessment* stage that "*Assessment of security risk includes identification of threat sources and threat events (..) and how the assets are vulnerable to the threats, the likelihood that a threat will exploit an asset's vulnerability, and the impact (or consequence) of loss of the assets*" [137]. Specific approaches (*i.e.*, methodologies or data sources) to identify these aspects are not directly defined by such frameworks but are analyzed in the scope of this thesis.

2.1.4 Threat Modeling and Secure System Design

In the previously introduced risk management frameworks, two observations arise. First, risk assessments can be done during the productive phase of an operation or during the conceptual stage. Furthermore, certain steps are described – without detailing *how* this step can be carried out for particular systems or organizations. This raises the question of which methodologies and tools one should use to actually determine threats to an asset. Ideally, this could be achieved without building the system beforehand since the cost of introducing a change arguably increases exponentially throughout the system's life cycle [17], as illustrated in Figure 2.5.

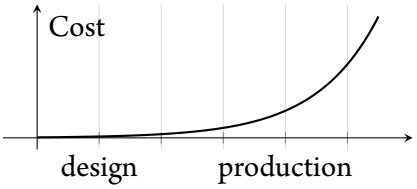

Figure 2.5: Cost of Change [17]

A methodology that has emerged to fulfill this particular gap is *threat modeling*. Threat modeling presents a structured approach to identifying risks, thereby supporting risk identification and vulnerability analysis [287]. Threat modeling is by nature a practical activity [232]. Threat modeling is, to the best of the author's knowledge, not a standardized activity in accordance with [195]. Providing a definition is not trivial – as observed by [246] – since even the authors of a recent survey [292] attempted to offer a unified definition – and ended up with multiple. Avoiding a clear-cut definition and focusing on characterizing the practices subsumed by the term *threat modeling* commonly involves the following questions being addressed [195, 232, 246, 292]:

1. Understanding the scope: what is being built or analyzed?
2. Identifying threats: how could this fail?
3. Designing countermeasures: what can we do against the threats?

This makes threat modeling a preventive measure [232, 246], even though potential countermeasures could operate in a reactive or proactive manner. For example, a web portal could be secured with strong multi-factor authentication (*i.e.*, proactive control) and with continuous monitoring of authentication attempts by the administrators (*i.e.*, reactive control).

The first question will yield substantial complexity since no designer can understand the system down to its most low-level details. Similarly, there is likely no complete knowledge about all the intentional or unintentional factors and impacts of the system's failures. Since the objective of threat modeling is to be practical while still presenting a structured activity, threat modeling makes – as the name implies - heavy use of abstractions. For example, Figure 2.6 depicts a fictitious couponing system. Without knowing (or caring for) the technological details, the depicted process enables the

identification of threats in different environments. For example, existing customers could *(A)* forge coupons, while actors could *(B)* intercept the communication between customers and the company.

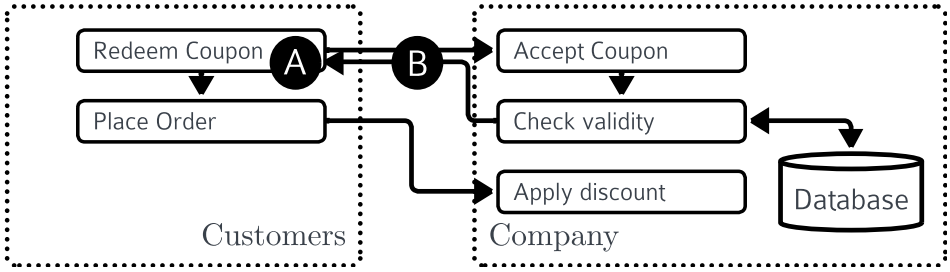

Figure 2.6: Exemplary Threat Model of a Fictitious System

To introduce the practice of threat modeling in closer detail, the remainder of this section iterates over established methodologies used in threat modeling, the representations of assets and threats that are commonly created, and an illustrative set of widely used tools.

2.1.5 Threat Modeling Methodologies

Threat modeling methodologies can be defined as the steps one takes to create a threat model [246]. However, it must be acknowledged that threat modeling may also follow an unstructured approach, where no particular methodology is being applied. For example, simple group-based brainstorming activities are still commonly used to develop a set of assets and threats [232, 248, 274]. One of the advantages of *not* following a particular methodology is its flexibility [274] – allowing threat modeling on virtually any representation or problem domain (*e.g.*, modeling terrorist attacks on critical infrastructure or remote code execution on an embedded sensor system). Brainstorming variants can be used, such as creating structured scenarios or movie plots (*i.e.*, definition of exaggerated threats for inspiration). Still, these vaguely unstructured methodologies present several limitations [232]. For example, the resulting threat model will only contain threats that are known (or imaginable) by the participants. Thus, it is expected that considerable time is spent familiarizing oneself with the literature or using attack taxonomies and libraries.

Structured approaches can be roughly summarized into three categories: asset-oriented, attacker-oriented, and software-oriented [232]. Asset-centric approaches start with the things that the organization (or the presumed attack) values the most and develop threats around them [248]. For example, *STRIDE* is a high-level methodology to discover threats through the threat categories contained in the mnemonic (*i.e.*, spoofing, tampering, repudiation, information disclosure, denial of service, elevation of privilege). *STRIDE-per-asset* iterates over a set of modeled assets and describes related

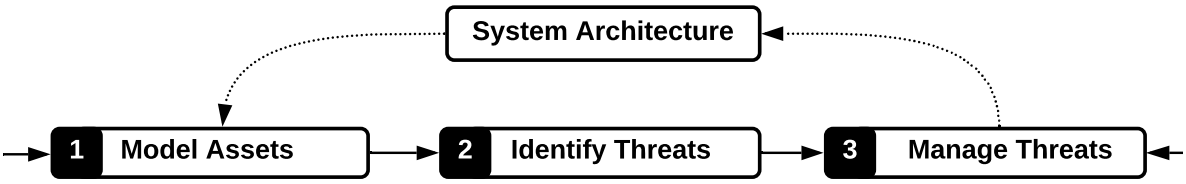

Figure 2.7: Abstract Threat Modeling Methodology [276]

threats according to the categories [45, 248]. Secondly, attacker-focused methodologies reason on the nature, motivation, and capabilities of relevant attackers and deduce possible threats [232]. For example, *Personae non-Gratae* borrows the concept of user type modeling from the User Interaction field to model attackers [164]. Finally, the third category of modeling tools focuses on the software aspect of threats [232]. As the name implies, these methodologies are closely related to the approach of how threats are represented. As one may have noticed from the underlying philosophy of these methodologies, there is a relation between the efficacy of a method and the stakeholders involved in its application. This will later be discussed in Section 3, where tools and methods are reviewed against particular problems.

Assessment Methods

On a high level, threat modeling methodologies (*i.e.*, a set of procedures applied for a particular problem [84]) leverage methods that are summarized by the abstract process of three consecutive steps visualized in Figure 2.7. Modelers are, thereby, implicitly decision-makers at various stages of the threat modeling process. For example, modelers have to decide which elements to include in the model of assets and which aspects to abstract from. Likely, this already influences the threat identification stage since the set of modeled assets influences the threats discovered. Moreover, threats are modeled (or not) based on their relevance and applicability. A similar decision-making is needed when discussing countermeasures. All these decisions ultimately impact the security posture of the information system, thereby exposing the embracing organization to risk. This raises the question of how these decisions should be informed. From a risk-oriented perspective, the answer should be clear: the focus lies on discovering effective control measures. These controls bring the most significant reward when considering their cost and remediation. During threat assessment, this would necessitate the identification of impactful threats. Using the concept of risk involves gauging the likelihood and impact of the risk materializing.

This process of threat assessment is either done implicitly or explicitly. Some threat modeling methodologies do not directly define a threat assessment method (*e.g.*, STRIDE-per-asset, STRIDE-per-interaction). Organizations then either complement this with another technique or conduct an implicit assessment. For example, organizations focus on threats that could be easily fixed – appearing to provide value quickly [232]. This may lead to ineffective threat models since several psychological biases are known in ERM. For example, *optimism* can lead to underestimating specific threats while others are overestimated. This bias can be found in daily life, too: People perceive sharks as more dangerous threats than swimming pools – although there was only ≈ one shark attack per year in the United States, compared to 3,306 cases of accidental drowning, 273 fatalities due to sun exposure, and 44,757 fatal car accidents [191].

Other methods and methodologies try to approach risk exposure more explicitly. For example, *DREAD* focuses exclusively on prioritization [71], defining a set of factors to approximate a threat's probability and impact [232, 248]. The acronym is decomposed – requiring users to assign a qualitative score to the scope of perceived damage, reproducibility, exploitability, affected users, and the threats discoverability [248]. *FAIR* applies risk decomposition to a threat, requiring the user to estimate threat frequency, capability, control strength, degree of vulnerability, and loss event frequency. Estimation follows based on estimation of worst-case and probable losses, making it a *quantitative* approach [232, 285]. However, implementations and real-world experiences within threat modeling are unclear [232] and thus potentially congruent with the voices of some practitioners. They claim that quantitative risk prioritization is unfeasible altogether due to the lack of realistic data [193]. Similarly, a subset of approaches simplify the risk-based model and only model factors such as impact (or severity) [232]. For example, the *OWASP ThreatDragon* tool supports multiple threat modeling methodologies but unifies the assessment using a simple, qualitative scale of *low – medium – high* [232, 274].

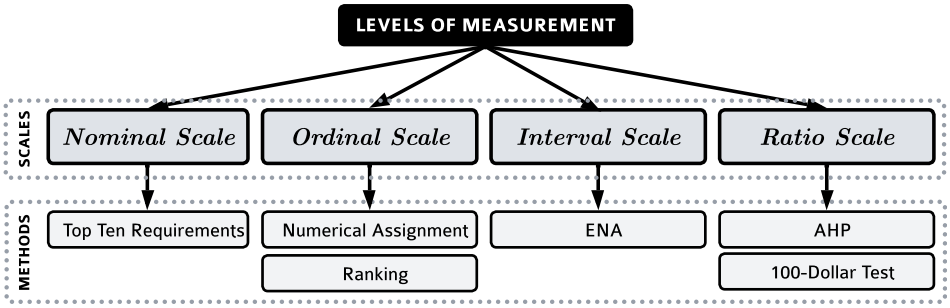

Figure 2.8: Levels of Measurement and Related Prioritization Approaches

Section 3.1.6 reviews related threat assessment approaches concerning their quantification or qualification philosophy. As will be illustrated, the underlying scaling of the measurements produced by those surveyed methods is not always clearly described by their authors or developers. Related disciplines, such as requirements engineering, explicitly describe the underlying measurement scales of proposed prioritization approaches. Figure 2.8 illustrates a set of exemplary prioritization approaches and their underlying level of measurement [5, 20], which are relevant to discuss threat assessment methods, too. Depending on the operations that should be enabled using a resulting value, each scale outperforms the previous one. The four scales are the nominal, ordinal, interval, and ratio measurement scales. The *nominal* scale is merely a categorization. It only enables grouping based on the mathematical operation of equality testing (*i.e.*, $r_1 = r_2$ or $r_1 \neq r_2$). While strict ordering is not feasible, these can be used to infer priority by providing context. Threats could be grouped into either "development time flaw" or "runtime flaw". The *ordinal* scale is the simplest scale to compare levels through sorting (*i.e.*, $r_1 \geq r_2$, $r_1 > r_2$, $r_2 \geq r_1$, or $r_2 > r_1$). In a dichotomous sorting, two groups of threats could be sorted; for example, "minor threats" are less severe than "major threats". Non-dichotomously, more than two groups are compared, *e.g.*, using the commonly adopted "low", "medium", or "high" scale. While approaches producing a score on the ordinal scale enable sorting, they cannot express the distance between those results. One should not express how much distance lies between the hypothetical scores "low" and "medium". The *interval* scale enables the expression of distances (*e.g.*, applying $r_1 + 5x = r_2$, where x is a non-ratio scaled unit). One could estimate that a threat t_1 requires 10 hours of additional work to be mitigated than t_2. While this expresses distance, it does not express a ratio. Aptly named, the ratio scale enables such discussions (*e.g.*, enabling $r_1 = 2 \times r_2$), for example, that one threat requires twice the amount of computational overhead than another threat [5, 20]. Since every prioritization in some way draws on comparing a dimension [5], these underlying scales must be evaluated to compare and contrast methods.

2.1.6 Representations

Threats and their relation to assets can be represented in several ways. A prevalent representation are *Data Flow Diagrams* (DFD), which describe how data is passed between system components. For example, how different actors interact with a process can be modeled through the various data flows to and from the process. Although not a standardized approach, practitioners commonly model abstractions such as processes, data flows, storage units, and entities (internal or external) [232]. The previously introduced example in Figure 2.6 could be categorized as a data flow diagram. A more standardized approach is taken by diagrams using the *Unified Modeling Language* (UML) [31],

which allows precise modeling of the systems' structural, stateful, or procedural aspects [232]. One aspect often adopted by various representations is the notion of a *trust boundary*, which separates the system components into areas with similar trust assumptions. For example, in the previous example, customers and any of their actions are considered untrusted actions from the perspective of the company and hence deserve different considerations than the actions inside its trust boundary. Threat representations can be combined to represent additional details. In the previous DFD, the actors, their trustworthiness, and the attack goal can be represented. However, the different means *how* an attacker may achieve that goal cannot be easily represented. *Attack trees* can be crafted to hierarchically decompose the attack goal into its preconditions and actions, as shown in Figure 2.9: forging coupons could be done by *either* bribing an insider responsible for the issuance of coupons or by simply guessing the codes used to denote valid coupons. The latter would require two sub-steps to be successful – the generation of a compatible coupon code and its validation using the online service. With this decomposition, one can reason on potential countermeasures. For example, mitigating either of the two sub-actions mitigates the guessing attack: using a strong cryptographic scheme or rate-limiting the number of validations a user can perform.

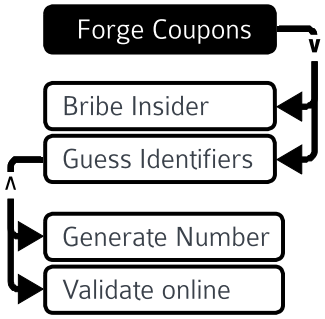

Figure 2.9: Tree-based Decomposition of a Threat

2.1.7 Tools

Threat modeling as an activity has received attention from the research community [123]. Nevertheless, it is a practical endeavor [253], necessitating a look at industry tools, too. Numerous tools have emerged to guide, operate, or simply support the threat modeling process. [230] describes that most threat modeling tools mainly fall into one of two categories relating to the type of artifact used as input. *Diagram-based* tools have emerged to provide an intuitive modeling approach closely related to system modeling. Both commercial tools, including the *Microsoft Threat Modeling Tool* or *ThreatModeler*, and open tools such as *OWASP Threat Dragon* and *OVVL*, enable drawing of the system with stencils. This core functionality is supported by all tools in this group – a subset of them then supports threat identification through pre-defined libraries. Most libraries are oriented around the *STRIDE* method. Conversely, *text-based* tools use more formal inputs, such as YAML files or Python code. For example, Listing 2.1 demonstrates how a database server running CentOS can be formalized – enabling automated threat identification using these details [230]. In summary, both tools require additional effort to either draw the system diagram or to formalize it. However, depending

on the data types supported and the development process followed, some artefacts may already exist (*e.g.*, architectural models, system inventories).

Listing 2.1: Text-based System Description for Threat Modeling [230]

```
db = Datastore("SQL Database")
db.OS = "CentOS"
db.inBoundary = server_db
db.isSQL = True
db.maxClassification = Classification.RESTRICTED
```

At last, a few tools that do not directly guide or automate a particular threat modeling methodology or representation deserve mention. Still, simple, highly flexible media can serve as a dynamic tool for threat modeling. The usage of *whiteboards* to model threats in an ad-hoc manner (*cf.*, Figure 2.10) is common [103, 232]. This can be attributed to their simplicity, flexibility [232], and because these tools are already common tooling in system engineering [248]. However, the low friction of accessing this tool also yields drawbacks, such as the difficulty of persisting models [248] or including a remote modeler (*cf.*, Section 3.1) [274]. Hence, collaborative online tools (*e.g.*, *Miro*, *diagrams.net*) are used, although they do not focus on threat modeling, hence lacking automation and guidance [248, 274].

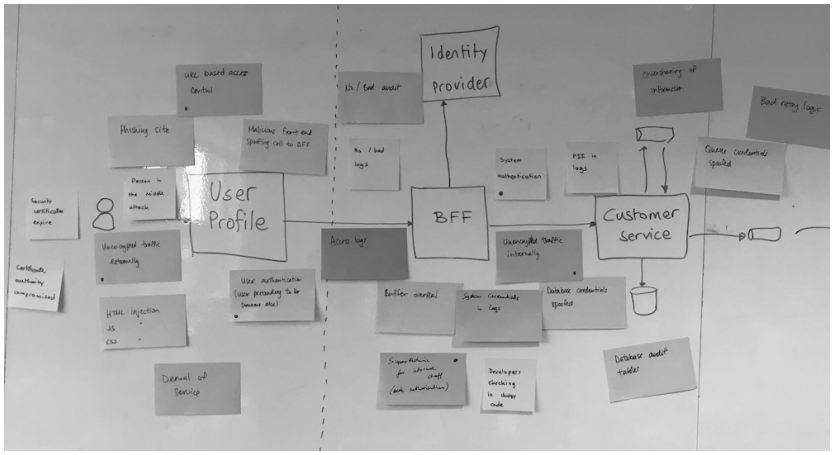

Figure 2.10: Threat Modeling on a Whiteboard [103]

In summary, various tools, methods, and representations are available to conduct threat modeling. The practice that has emerged from this creates value by producing constructive steps to reduce the risk exposure during the system's operation. Relying on abstractions and models can be achieved early in the design process, lowering the cost of implementing control measures. Nevertheless, practitioners and researchers are aware of its limitations and challenges, which will be discussed in Chapter 3.1, yielding the research limitations and opportunities that are the focus of this thesis.

2.2 Preventing Threats, Detecting Attacks, and Responding to Breaches

Threat models are developed to understand the contributing factors of an organization's risk exposure. This understanding then helps to develop measures that reduce the risk. Intuitively, one may conclude that the critical goal is to *prevent* these threats during operation. For example, a threat model of an e-banking system could highlight the susceptibility of a publicly exposed web interface to a remote code execution attack. At development, it could be required that this publicly exposed component must be isolated from other components – reducing the risk stemming from this threat at design time by preventively reducing the attack surface [146, 248]. However, a threat model can also be used to develop *detective* and *responsive* measures. For example, for the e-banking system, potential security requirements could include that an anomaly in the web services execution environment will be analyzed within 24 hours and that an incident response team executes a particular set of steps for each report. Hence, the mitigative actions and requirements derived from a threat model can be discussed according to the *prevent – detect – respond* frame [232].

2.2.1 Prevention

Threat Modeling lends itself to identifying structural controls to prevent (or at least reduce) a threat's success. This is especially true when considering tactics that address structural vulnerabilities at *design* time. However, during *implementation*, it is still trivial to introduce implementation vulnerabilities since every single line of code in an application represents a decision-making process [146, 248]. For example, the previously illustrated *Heartbleed* incident represents a highly localized flaw within a system's implementation, not a design flaw. Subsequently, only high-level *tactics* (*i.e.*, abstract, technology-neutral formulations of security controls) are summarized to present the reader with a high-level overview. The background to key technology needed to implement said tactics is presented in Section 2.3.2.

Similar to the efforts that compile attack libraries, several taxonomies and libraries of information security controls [49, 186, 231] exist. [232] organizes them according to the *STRIDE* mnemonic. To address *spoofing* attacks, tactics related to the authentication of people and resources are key. Nowadays, this effectively translates into the usage of cryptography to authenticate an entity, since other authentication mechanisms (*e.g.*, based on names or IP addresses) have provided weak guarantees. While these cryptographic techniques provide strong mechanisms to verify the claims of an entity, tying this entity to a *person* is a difficult endeavor [232].

Ensuring data integrity against *tampering* attacks can use several techniques. First, tying into the previously mentioned *authentication* strategies; *authorization* ensures that access to data is based on a

permission model [232], commonly involving the definition of users, roles, and permissions [186]. When implementing authorization, it is vital to consider that an entity has no means to further increase its privileges (*i.e.*, *elevating its privileges*). While these techniques do not prevent tampering, cryptographic techniques (*i.e.*, hashing and signatures) play a vital role in detecting such an attack at a later stage, too [186, 232]. When enriched with other data, these techniques are also relevant against *repudiation* attacks [232], since they require gathering strong evidence about an individual's actions [186].

Data confidentiality can be violated in technical and non-technical domains of a system – and during storage and transmission. Many of the previous controls are applicable (*e.g.*, authentication, authorization). In addition, data can be encrypted during transmission and storage [186, 232]. However, if such techniques are used, the observation made by [248] must be emphasized: threat modeling remains an iterative process. For example, if data is encrypted before transmission, cryptographic key material must be used and possibly exchanged. Hence, security controls must be adopted to secure this management of keys [186].

Attacks on the availability of a system using Denial-of-Service (DoS) attacks can unlikely be prevented entirely. After all, every system operates on a finite amount of resources. Nevertheless, a simple brute-forcing attack can be partially prevented by increasing the amount of resources (*i.e.*, network bandwidth, memory, processing power) or by filtering the DoS attack at a location convenient for the defender. Similarly, for specific adversarial purposes, quotas may be sufficient [186]. However, highly targeted attacks address implementation vulnerabilities and require fixing the system's implementation [232].

In addition to these mostly technical controls, it is vital to consider the human factor, too. [231] and [186] acknowledge that not only *technology* but also *people* must be addressed. Non-technical measures such as security literacy training, awareness building, and communication are considered generic measures to improve risk exposure [186].

2.2.2 Detection

Even with these broadly defined preventive tactics, not all threats can be mitigated. Furthermore, relying on a single line of defense would not be sensible. For example, the integrity of a file resting on a web server can be protected through authentication and authorization of its access requests. However, when the file is sent on an untrusted channel, the integrity of the file should be ensured through *detective* measures [186]. This is even the case when the data is transported in an encrypted form since this only protects the confidentiality of the message [232]. In general, the following four

aspects can be monitored [232] with respect to detection: changes in an organization, signatures expressing attack patterns, irregularities from normal operation, and impacts.

In 2006, [150] conducted a study on the effect of introducing *changes* on the security of an information system. As displayed by the two concentric distributions in Figure 2.11, the black line indicates how prevalent specific strategies are in organizations with "medium-performing" information security programs. They discovered that the prevalence of particular control measures was similar to that in "high-performing" organizations (*i.e.*, gray bars). For example, access control was commonly implemented by both low-performing and high-performing organizations. However, the authors argued that adopting *change management* – widely found in high-performing organizations – may represent a key indicator of a resilient organization. More specifically, what those organizations successfully implemented was that they not only monitored the information systems to detect unauthorized changes, but also had actions planned for these cases. Although such organizations likely did not focus on detecting attacks themselves directly, the study revealed that such organizations were more likely to automatically detect breaches and avoid losses. Consecutively, the authors argue that the effect of change management is recognized by the business side, allowing the security personnel to negotiate better budgeting and improve their program [150].

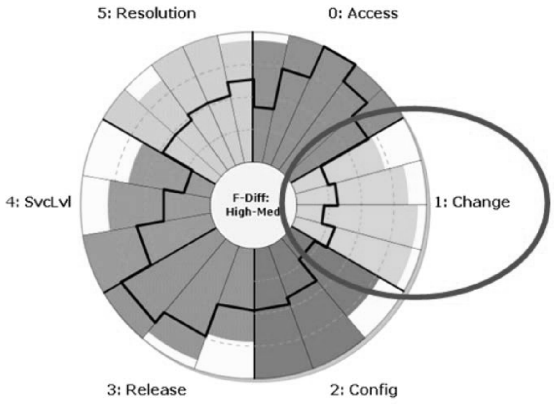

Figure 2.11: Usage of Control Mechanisms Across Organizations [150]

While these approaches focus on changes at various levels of an organization, a great deal of attention has been given to detecting intrusions. This resulted in the development of *Intrusion Detection Systems* (IDS), which aim to detect attacks as early as possible [147]. These systems subsume the concept of detecting either *attack signatures* or *irregularities*. Concerning detecting signatures, the idea is that cyber attacks rely on files, resources, or strategies to execute them (*e.g.*, malware, server with fixed IP address) and that it is possible to detect a known set of malicious patterns [147, 232]. This strategy can be applied at the host or within the network. For example, a file associated with previous malicious activity could be detected on the operating system receiving the file or during transmission by a firewall. Several data sources exist to extract patterns, which can then be formatted and detected using different approaches. Such *signature*-based systems have been demonstrated to detect attacks accurately; however, this only holds for previously detected attacks [147]. Since

this approach relies on static patterns, modifications to the attack patterns strongly decrease the detection accuracy [147, 232]. This had been recognized by threat actors, resulting in polymorphic attacks [232]. For example, ransomware has been proliferated for decades, even leading to the emergence of Ransomware-as-a-Service, which enables individuals to obtain custom ransomware payloads and infrastructure [7]. Recognizing that threat vectors and their implementations change, *anomaly*-based intrusion detection systems focus on the information system to be defended. Then, various data sources and techniques are used to build a model of how the system should appear in a normal (*i.e.*, unattacked) state. In the second phase, this model detects deviations from the normal state, indicating a potential attack. Technically, various approaches ranging from simple statistics to Machine Learning-based systems can be used, yielding varying degrees of performance. Conceptually, these systems could detect novel (*i.e.*, zero-day) and polymorphic attacks, since both are assumed to differ from normal operation [147]. However, two limitations hinder their performance: false positives demand costly inspection [12] and, secondly, change is inevitable for businesses, and hence changes to these systems or their usage could be misinterpreted as an anomaly [232].

The fourth detective category of strategies focuses on an attack's *impact*. For example, data leakage can be inferred from the presence of sensitive internal information on an external system. Similarly, a fraud-related cyber attack could be detected in a completely non-technical manner by the finance department [232]. This exemplifies that cyber attacks, including their execution and impacts, can be detected at various organizational levels. Several recently emerged techniques have been explored to implement these concepts. However, it is critical to stress that these approaches are only effective if the monitoring and detection results lead to a well-defined and integrated response [232].

2.2.3 Response

Historically, a strong focus has been placed on preventive protection. Only two decades ago, there was increased attention to the detection of attacks, leading to a myriad of detection systems. Recently, *response* to cyberattacks has received more attention [225]. As argued by [225], protection centers

Figure 2.12: Incident Response Life Cycle in NIST SP 600-61 [100]

around the usage of *technology*, with *processes* and *people* playing an assistive role. Response, on the other hand, is initiated and executed by people – assisted by processes and technology. Furthermore, it is argued that it is challenging to substantially automate this activity.

The resulting activity of attack response consists of preparing, deciding, and acting [100, 225]. NIST summarizes this process in four key steps, as highlighted in Figure 2.12. The preparatory steps described by [100] stress that even the post-mortem incident response activity must be developed *before* an attack occurs. To be prepared for handling incidents, organizations define responsible personnel and obtain resources and tools to respond to upcoming incidents. It must be determined which internal and external personnel can be involved and how they can be reached – ideally, with different, independent communication media [100]. For example, in a non-security related outage in Meta's data centers, incident response was delayed by the reliance on communication systems that relied on affected systems [85]. Thus, this step requires not only the procurement of complex technical software (*e.g.*, packet sniffers, protocol analyzers, forensic workstations) but also the definition of simple aspects, such as an office facility (*i.e.*, "war room") where the incident response team can operate [100]. Since incident response involves several stakeholders (*e.g.*, management, human resource, legal and communication experts) [100] it is a highly people-intensive acitivity [225]. It is critical to closely align this step with the preventive efforts since an increased number of response actions will be costly [100].

When analyzing incidents, data collected by different systems must be considered. This is not a trivial task for a response team since there are many data sources and types, and specialized technical knowledge is required to analyze the many incident reports. The key job is promptly assessing whether an incident report indicates a security-related event. In addition to this binary decision, additional information must be gathered and documented (*e.g.*, scope of the incident, affected systems and networks, threat actor and vector). These details are the basis for prioritizing the incident – based on impact and recoverability – and communicating it with the respective stakeholders. The next step is to contain the damage and recover the asset. Limiting an attack may involve impactful decisions (*e.g.*, disconnecting or limiting a network), stressing the need for stakeholder collaboration. The containment strategy depends on various factors (*e.g.*, potential damage, need for evidence) and the type of attack. For example, a malware-based attack introduced by an insider attacker could require containment in a sandbox to gather additional information – making it radically different from an application-layer DDoS attack executed by a nation-state actor [100]. Hence, one can also observe an iterative approach aligned with the previous analytical step (*cf.*, Figure 2.12).

Similar to the containment strategy, the approach for eradication and recovery depends strongly on the type of attack – for some, only recovery is applicable. In any case, the objective is to return

to a normal operation mode (*e.g.*, by restricting network access, restoring backups, refining access control). Since attack targets are usually attacked more than once [229], it is critical to discuss the learnings of an incident in a post-processing step. Effectively, stakeholders review how the incident took place, how it was handled, and how well this overall process operated, paving the way for highlighting how it could have been improved. For example, it can be discussed which information could have been needed, whether documentation was adequate, and whether personnel was well-prepared. This discussion can involve both objective and subjective statements. For example, an incident can be characterized by the time it took to resolve and by the perceived performance of the stakeholders. Based on this data, the high-level incident response process (*cf.*, Figure 2.12) can be closed – linking it to preparatory steps. A resulting audit reviews the current plan and identifies deficiencies in the policies, tools, and personnel [100].

2.3 The Cyber Threat Landscape

Thus far, risk was introduced as an abstract concept, and practical approaches to manage risk were introduced by following the *prevent – detect – respond* frame. Subsequently, the threat landscape that organizations have to navigate is introduced. First, conventional and widely described threats that could affect most types of information systems are presented. Since this thesis investigates threats for specific technologies and domains (*e.g.*, Artificial Intelligence, voting systems), these problem areas are introduced from an information security perspective. Finally, novel methods that will later be exploited for research within these areas are described.

2.3.1 Threats and Risks

Attesting to the size of the current information security landscape, several taxonomies try to paint a clear picture of the many existing cyber attacks. [38, 84] describes them according to cybersecurity domains: ones directly affecting a user, attacks targeting applications, software, networks, or devices. [233] structures them through multiple dimensions: the attack vector used, the operational and informational impact caused, the defense methods applicable, and the target. Figure 2.13 highlights the operational impact pillar from [233], which is used to describe abstract threat behaviors.

Malware

Suppose a piece of software has been designed and implemented with the intent of maliciously causing harm to a victim's infrastructure (*e.g.*, networks, mobile devices). In that case, the piece of soft-

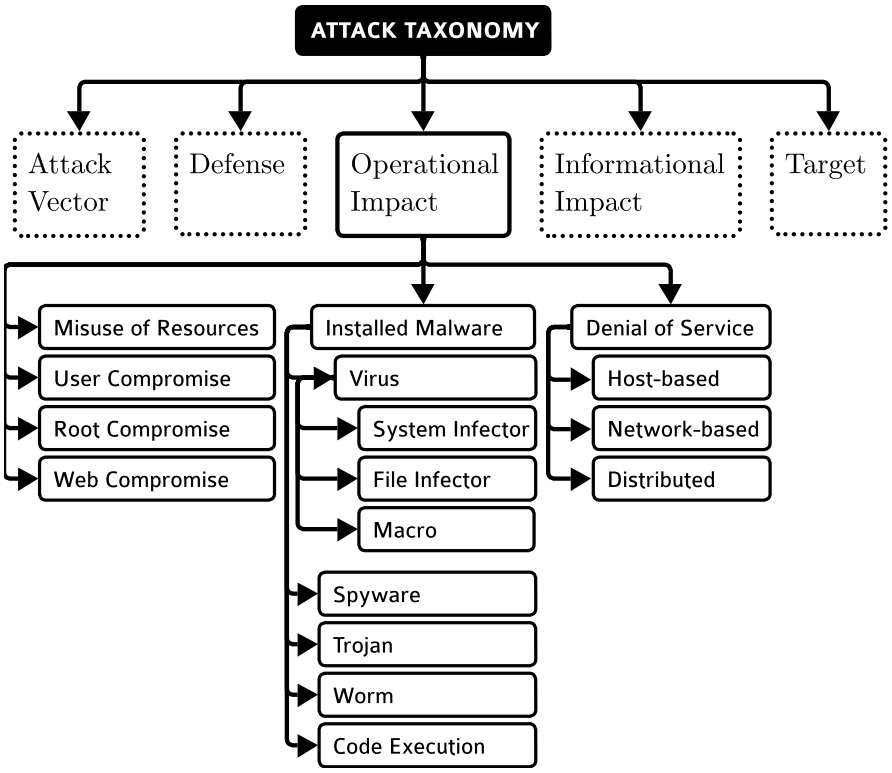

Figure 2.13: Attack Taxonomy: Categorizing Attacks by Operational Impact [233]

ware would be classified as *malware*. Malware used to be implemented rather simply without much thought given to its obfuscation. *Next-generation malware*, on the other hand, has been born out of the necessity to obfuscate artifacts or behavior due to the emergence of malware detection approaches [14]. This set of threats is not necessarily new. For example, [60] used the term "virus" back in 1987 to describe the problem of attaching malicious behavior to an existing computer program [60]. [264] report that many people are aware of the threat. Hence, one may believe that malware is a well-controlled threat. However, it is estimated that $\approx$1 million malware samples are created every day, leading to yearly financial damages attributed to malware of 6 trillion USD [14].

While malware samples vary in behavior, researchers classify them into a set of categories (*e.g.*, viruses, spyware, trojans, *cf.*, Figure 2.13), indicating commonly observed behavior, impact, or strategy [233]. Many malware types can be further classified into subtypes [233]. In practice, variations in malware implementations (*i.e.*, malware strains) are often described in terms of threat intelligence [66]. Due to the polymorphic nature of malware [14], one may find numerous software artifacts relating to a similar strain. For example, *MalwareBazaar* reports 14 samples tagged for the *Mirai* malware strain on the day of writing (September 28, 2024).

Table 2.1: Non-exhaustive Set of Malware-based Attack Vectors

Category	*Description*
Virus [54, 236]	Self-replicating malware that attaches to valid programs or files
Worms [54, 236]	Propagating standalone malware
Trojans [54, 236]	Masqueraded programs providing backdoors
Spyware [54, 236]	Gathers user data
Rootkit [54, 236]	Conceal user-level programs
Botnet [54, 236]	Set of compromised, remote-controlled hosts
Adware [54]	Display unwanted advertisements

Ransomware is one malware type that has received significant attention due to its prevalence and the extent of damage inflicted. Accounting for almost 20% of cybercrime [127], successful attacks cost between 1 to 8 million USD [125]. The evolution of this threat can be observed back to the 1980s when floppy drives were used to distribute malicious software [64]. Nowadays, ransomware can be characterized by multiple subtypes – all sharing the aspect of requesting a ransom to be paid after indicating that damage to a system was done. *crypto-ransomware* threatens data availability by encrypting files with a cryptographic key [6]. As a result, any service depending on the availability of that data is impaired for the duration until the payload is eliminated, and either a backup is restored or the data is decrypted. Some instances (*i.e.*, *scareware*) do not actually encrypt the data, but merely scare the user into believing that the data is encrypted [13]. However, data must not necessarily be made unavailable through encryption – *locker malware* adds unwanted system functions (*e.g.*, lock screens, popups) to complicate data access [51]. To increase the pressure on the user to pay the ransom or to maximize the profits after a victim paid a ransom, *extortion-ransomware* threatens the user to publish previously exfiltrated data [9]. These sub-types share the aspect that these damage functions can use several infiltration methods, ranging from social engineering to phishing [9]. Since attackers have to maintain some level of infrastructure (*e.g.*, to communicate with the victim, receive payments) – a problem that *ransomware-as-a-service* has transformed into a business model by providing this infrastructure, usually involving the provisioning of polymorphic samples [7]. Besides ransomware, several related malware-based threats exist, which may even be combined into different tactics. A brief summary of related malware types is presented in Table 2.1.

Denial-of-Service

Some of the previously introduced *ransomware* malware sub-types already exemplified the threat of disrupting data availability, impacting the service that depends on that data. When a service (or an underlying resource needed to provide that service) is attacked to deny the victim service access, this threat describes a *Denial-of-Service* (DoS) attack. Three variants of DoS can be identified in the literature: host-based, network-based, and distributed DoS attacks [233].

Host-based DoS attacks use a threat vector (*i.e.*, a path to apply the threat to the target) that focuses directly on the host. Thus, flaws within the deployed operating system, system configurations, or user-level software are exploited. *crypto-ransomware* could be attributed to this threat description, too. However, an attack could target the availability of resources other than data [233]. For example, a program that is executed in user space could allocate a large amount of resources. In many cases, these attacks need to exploit specific vulnerabilities of the system's implementation, and may therefore require some implementation knowledge (*e.g.*, algorithmic complexity, encryption methods) [101].

Network-based DoS attacks target or leverage the computer networks employed for the service. A common form of this is flooding, where a large amount of network traffic is created to exhaust the available resources of a network or host [233]. A more advanced attack involves the exploitation of network protocols involved in the communication [101]. For example, Figure 2.14 highlights the three-way handshake needed to establish a connection in TCP. Before any message is sent, the "server" must perform a "passive open" operation to effectively listen for any connection requests. The client performs an "active open" operation, which leads to a TCP segment being sent with the `SYN` control bit indicating a connection request for the connection identified through its `sequence number`. Later, the server will acknowledge the connection request (*i.e.*, setting both control flags `SYN` and `ACK`). Finally, the client acknowledges (*i.e.*, sets the `ACK` control bit) the received confirmation, which completes the connection setup procedure. Since TCP is a connection-oriented protocol, this procedure is stateful. For example, after the server handles the first message with a `SYN` flag, the server remains waiting for the final acknowledgment. If the acknowledgement does not come in during a pre-defined "timeout", the server sends a TCP reset segment (*i.e.*, `RST` control flag) [161]. Therefore, the server must track the connection (*i.e.*, record the transport endpoint address, operate the timer) requiring computational resources (*e.g.*, memory, CPU). A *TCP SYN Flooding* attack exploits this protocol detail by sending numerous

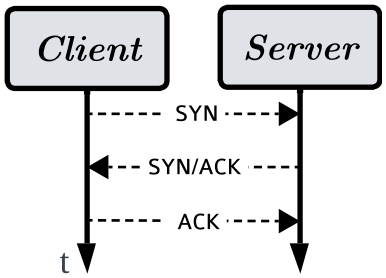

Figure 2.14: TCP 3-way Handshake [161]

TCP segments with the SYN flag activated, exhausting the server's resources through the numerous half-open connections. In addition, an attacker may spoof the source IP address in the packet. The SYN/ACK message will never reach the client – leading to an asymmetric attack scenario. An attacker can further increase the volume of TCP segments reaching the server by using multiple sending hosts – turning the attack into a *Distributed Denial-of-Service* (DDoS) attack [101, 237].

Table 2.2: Illustrative Overview of Denial-of-Service Attack Vectors

	Threat	***Protocol***	***Descrption***
Layer 5-7	DNS amplification [57]	DNS	Query DNS resolver with spoofed source address
	NTP amplification [59]	NTP	Query NTP server with spoofed source address
	HTTP Flooding [56]	HTTP	Computationally-expensive HTTP requests
Layer 4	UDP flooding [237]	UDP	Flood (spoofed) UDP packets
	SYN Flooding [101, 237]	TCP	Exhaust connection tracking in half-open state
	ACK Flooding	TCP	Exhaust resources through connection lookup
Layer 3	Ping Flooding [58, 237]	ICMP	Flood ICMP packets
	Ping of death [58, 237]	ICMP	Send packet that exceeds the max. packet size
	Smurf attack [58]	ICMP/IP	Send ICMP message with forged source address
Layer 1-2	Jamming [190, 207]	Various, *e.g.*, LoRA, 802.11	Overlap radio channels
	Exhaustion [190]	Various	Exhaust batteries through forced retransmissions
	Unfairness attack [190]	Various	Misuse MAC cooperation
	Collision attack	Various	Misuse MAC cooperation

Similar network-based (D)DoS attacks can be observed for almost all layers in the OSI model. Table 2.2 illustrates a non-exhaustive list of attacks exploiting network protocols. On the bottom, attacks exploit lower-layer protocols, which are usually generic enough to attack almost any host (*e.g.*, ICMP flooding) – moving up, the specificity increases, with application-layer protocols targeting specific implementation weaknesses in the service (*e.g.*, identifying an expensive network request, such as a write or search operation) [237]. Unrelated to those threats is the motivation that attackers have. [258] outlines several of them, including accidents where an "attacker" unknowingly causes actions that deny the availability of the service. Financial motives include direct extortion or executing an attack as a service. However, DDoS attacks may also be motivated by other reasons, including protest, revenge, or as a competition [258].

Compromise and Misuse

In addition to Malware and DoS, operational impacts can include compromise and misuse. *Misusing of resources* can refer to a wide range of malicious behavior, where IT resources are used in an unintended or unauthorized way [233]. Since this is a rather broad definition, various vulnerabilities can be exploited to access such resources, including technical and non-technical methods. *Insider attacks* refer to threats executed by actors that already hold elevated privileges or knowledge by design: insiders of an organization. This personnel then acts as a threat actor – intentional or unintentional [174, 277].

Insider attacks are particularly dangerous since they can potentially skip some compromise-related threat vectors needed to obtain initial access. If this is not the case, outsiders have to compromise a system to obtain privileges needed for further stages of an attack (*e.g.*, persistence, fraud). Here, *User Compromise* refers to obtaining a particular user's privileges on a host. While this user may hold certain privileges as specified by an access control policy (*e.g.*, deleting his own files, reading files of other users in the same team), these privileges are usually limited (*e.g.*, a user cannot change the IP address of a system or delete files from a non-member team). An attacker may aim to elevate its privileges beyond the compromised user, ideally targeting a user with administrative privileges (*e.g.*, administrator, or "root"), executing a *root compromise* attack [233].

In addition to these attacks, numerous additional variations and technical details of these threats could be introduced. For example, the previous *user compromise* attack could be further refined by the vector employed, such as a *web compromise* attack, when flaws in web-based technologies are exploited for the attack. Similarly, several threats could be described that do not directly fit the *operational impact* pillar in [233]. Similarly, less impactful tactics, such as *scanning*, could be applied, which is – depending on the asset's sensitivity – not as impactful. For an extensive overview of conventional cyber threats, the cyber kill chain (*cf.*, Figure 2.15) proposed by [124] serves as an illustrative meta-model, depicting the sequential combination of attack strategies, ranging from the initial *reconnaissance* to *damage*.

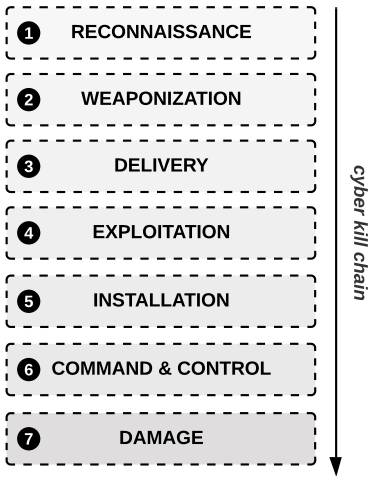

Figure 2.15: Cyber Kill Chain Model [124]

2.3.2 Problem space Domains

Research performed within the scope of this thesis includes the development of artifacts to test a methodology or a model to evaluate the detection of specific instances of cyber attacks. The methodological philosophy behind *design science research* decomposes the relevant aspects in such studies according to two dimensions: theory and practice, and problem and solution space [72]. This relation is visualized in Figure 2.16. Before introducing adopted research and design methods – including the design science research concept – the following three sections present three problem domains wherein the information security research in this thesis was conducted.

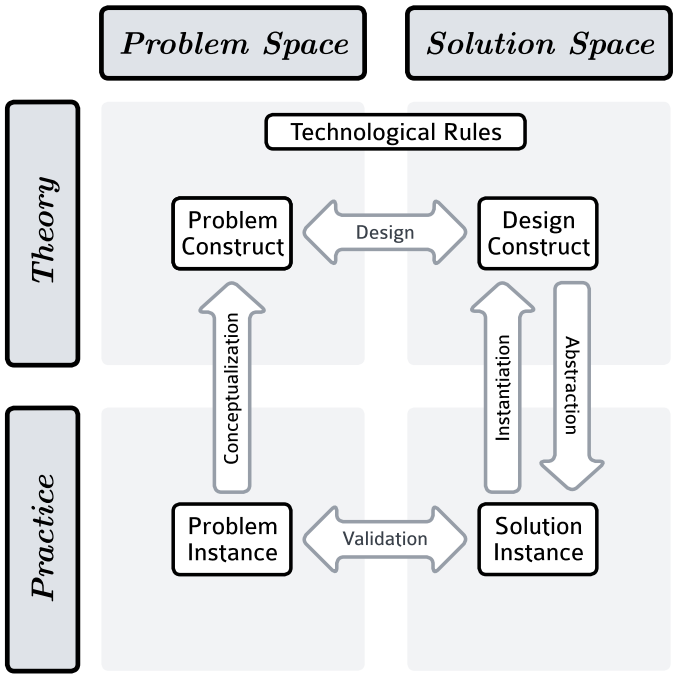

Figure 2.16: Relation between Theory, Practice, Problem Space, and Solution Space [72]

Internet-of-Things

The Internet-of-Things (IoT) can be broadly defined through systems that exhibit comprehensive awareness, process data intelligently, and transmit data reliably [221]. These three aspects can be observed in many devices that have been historically developed – making it difficult to trace the first IoT device. For example, a prototypical system involving a remotely controlled toaster was demonstrated in 1990. Later, *Radio Frequency Identification* (RFID) was integrated into devices for object identification. Nowadays, IoT devices have seen increasing deployment – in some industries, they

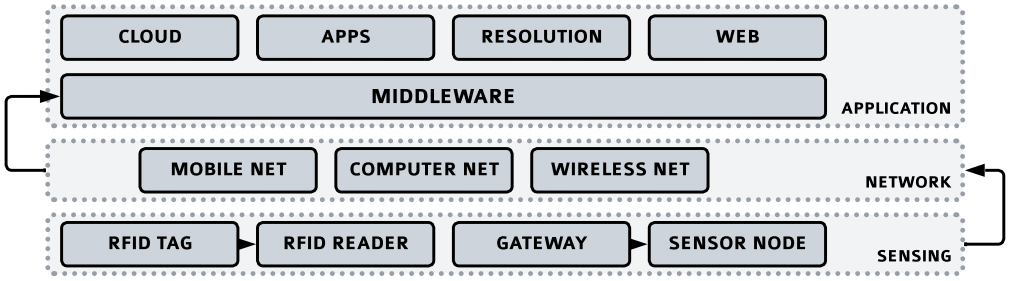

Figure 2.17: Internet-of-Things Three-Layer Architecture [221]

have become almost a best practice for specific use cases (*e.g.*, temperature control using thermostats, manufacturing sensors). This has resulted in a field where numerous devices are brought online every minute, yielding estimates of 31 billion actively connected IoT devices. Predictions estimated that the exponential growth at which "things" are connected to the Internet will yield 75 billion IoT devices. These devices support a broad range of use cases, ranging from smart cities to medical monitoring [221].

From an architectural and implementation perspective, it can be inferred that IoT as a field is not only concerned with the "things" (*i.e.*, a resource-constrained, connected computing device) but with a sizeable infrastructural backbone, as shown in Figure 2.17. The lowest layer represents the perception-related functions (*i.e.*, sensing). This layer contains physical IoT devices that provide sensor or actuator functionality. These sensors and actuators are commonly embedded into an environment [221] as resource-constrained devices [151]. The network layer provides data transmission between IoT devices, from IoT devices to other networking equipment, or directly to a server. Essentially, this layer ensures communication between *devices* and *services*. Several protocols apply to implement the different network types (*i.e.*, wireless, computer, and mobile networking). Finally, the top layer provides actual application functions, necessitating integration with other technologies such as conventional applications, cloud computing, or web services [221].

On the sensing layer, it is common to find devices with limited resources to comply with requirements such as cost-efficiency and low power consumption [156, 221]. For example, [151] describes three classes of constrained IoT devices (*i.e.*, with respective RAM configurations of $<$ 10 KB, $\approx$ 10 KB, and $\approx$ 50 KB) [151]. However, nowadays, IoT devices can also include other hardware, such as *Single-board Computers* (SBC), that have reached far higher specifications of computational resources. For example, [156] argues that the Raspberry Pi family of SBCs still satisfies the requirements for an IoT hardware platform since it provides a cheap yet flexible development basis. These

devices are technically far beyond the resource-constrained device classes defined by [151]. Even the first versions of the Raspberry PI were shipped with several hundred Megabytes of RAM. This allowed them to run operating systems commonly found on other devices (*e.g.*, BSD flavors, Linux distributions) and, thereby, enabled applications to be implemented with various programming languages (*e.g.*, Java, Python, C) [156]. Depending on the use case and deployment model, these devices can still be considered resource-constrained: commonly, such a device will need to operate autonomously and with little administrative intervention from the user [221]. This aspect must also be considered when discussing a key concern of IoT adopters: information security.

Security in IoT is limited mainly due to the computational expense of security schemes such as encryption and digital signatures. Additional economic considerations contributing to this limitation are the short time-to-market. Similarly, security mechanisms are unlikely to contribute to the perceived usability of a device. Usability must receive attention since 60% of IoT devices are directly used by consumers. Finally, a contributing factor to the device's attack surface is, as their name implies, their frequent or permanent accessibility through networks [221]. OWASP identifies several common security weaknesses in IoT devices, including weak passwords, insecure network services, or system interfaces [196]. These vectors could be exploited by threats that are not necessarily novel, including device compromise or malware [221].

One particular subfield of the IoT field that presents an accessible problem space for this thesis (*i.e.*, allowing for problem identification and design validation) is crowd-sensing. In crowd-sensing IoT, sensor data is collected through a crowd-sourcing approach. By collecting data on the utilization of the radio spectrum in a decentralized manner, various services can be offered centrally: transmitter localization, signal classification, or spectrum anomaly detection. To achieve this goal in *ElectroSense*, consumers operate a sensor node using low-cost hardware (Raspberry Pi, Software-Defined Radio Kit), while the project maintainers operate a data collection and processing backend. Since these devices are operated by consumers in residential *Local Area Networks* (LAN) and Raspberry Pi devices operate a full-fledged Linux-based operating system [211], a particular threat model arises. First, threats apply to all three layers of the IoT architecture. For example, the sensing layer could be attacked by tampering with the sensor data [221, 243] or by infecting the device with malware [47]. While the network layer could be attacked by a man-in-the-middle attack [221], the backend could be threatened through a DoS attack. Focusing on the device itself, several security controls are infeasible. For example, the hardware platform does not include a hardware security module [213, 221], and since the networks are operated in consumer networks, network-level defenses such as firewalls [25] cannot be easily enforced.

Cybersecurity in the Public Sector

Organizations have rapidly adopted *Information and Communications Technology* (ICT) systems. Unsurprisingly, offering services online has spurred innovations in the public sector, too. By 2011, the Swedish government estimated that 65% of citizens were filing their tax forms online. As a result, several public services are now not only supported by information systems but also offered as digital services. This includes health services, tax services, social institutions, digital identification, or voting systems (frequently termed e-Tax, e-Health, e-ID, e-Voting) [11].

This increased reliance on ICT systems not only extends the service offering into "cyber space" but also extends the attack surface therein [11]. Given the prevalence of information systems, systems are exposed to cyber risks whether the service is offered digitally or not. For example, in voting systems, advances in information systems have spurred the development of *New Voting Technologies* (NVT) that can aid any phase of the election process (*i.e.*, not only the voting step). Even in a voting system relying on paper ballots, NVT can automate the tallying process [148]. This stresses that not only online services are vulnerable to threats to their information systems. For example, the term *e-Voting* does not strictly imply a *remote* voting system, but simply the usage of electronic voting technology [148]. Factoring in that information services employed in the public sector often lag behind private ones regarding cyber resilience, a concerning picture arises [11].

Many threats that apply to commercialized organizations also appear in threat models for services offered by the public sector. However, threat actors may be driven by other motives (*i.e.*, political ones). In addition, the particular use cases give rise to several specializations of existing threats. For example, in a voting system, the provider could issue additional tokens for non-eligible voters. A few characteristics highlight the particularity of cybersecurity in the public sector. First, the components of information systems operated by the public sector are often operated with a high degree of autonomy. Secondly, the operation of these information systems includes organizations from the private sector, too. Furthermore, human behavior comprises an essential aspect of managing public cyber security [11]. One could hypothesize that not all methods from the solution space apply to public sector systems. For example, 30% of (commercial) organizations considered adopting cybersecurity insurance, allowing them to at least partially transfer their risk exposure [22]. In a public service, such as a voting system, it is unclear how one could conceptually transfer the risk exposure of, for example, a breach in ballot secrecy.

Two of the previously mentioned problems will be exemplified based on a real-world incident in e-government: the crucial aspect of human factors in cybersecurity and the fragmented landscape of processes and technologies underpinning such services. The first case study refers to human factors, as embodied by the threat vectors that can be modeled and described as "insider threats". If one

were to fully model this class of threats using an attack tree, the initial access tactics could be highly simplified due to the role of the attacker. In such attacks, trusted personnel (or the personnel of a trusted organization) act as an attacker. The motives for such attacks can vary substantially, and they may be executed intentionally or unintentionally.

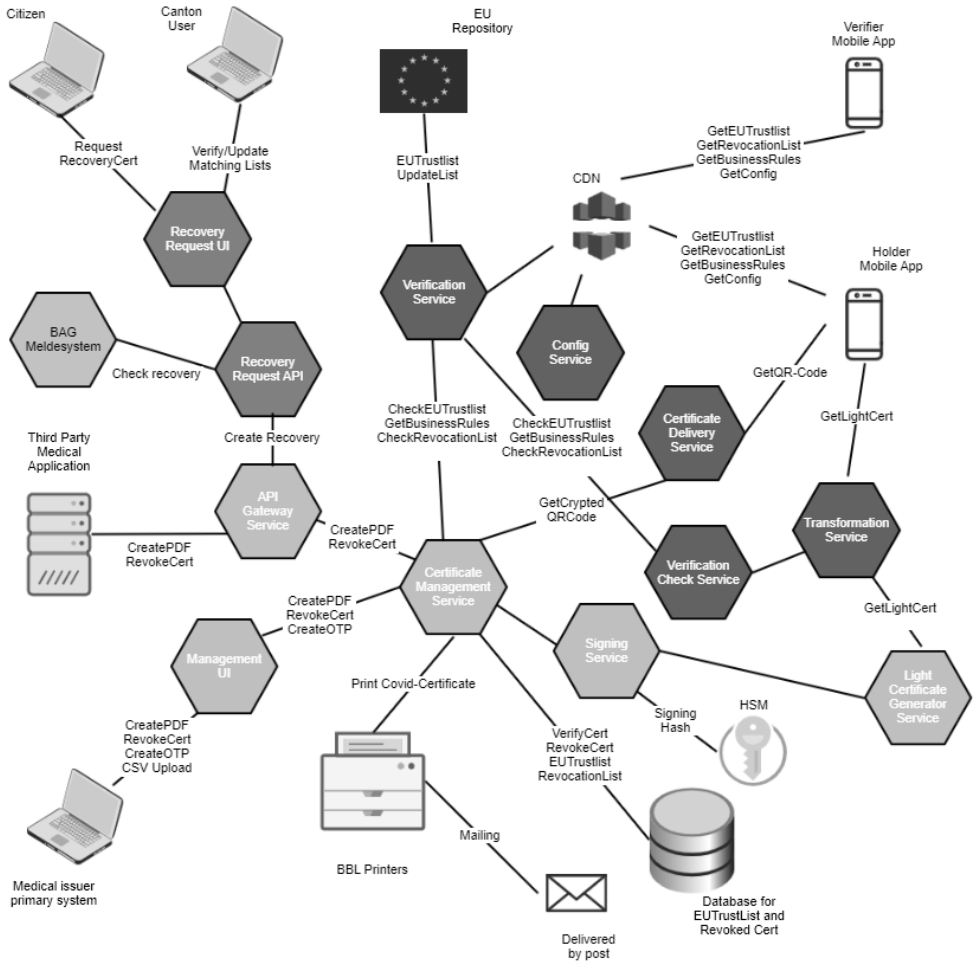

Figure 2.18: Architecture of the Swiss Covid Certificate System [182]

Mitigating such attacks is non-trivial since technical measures cannot fully eradicate the threat of attackers that already hold elevated privileges due to their position [174]. The specific instance of an insider attack used to illustrate this threat is the Swiss COVID-19 certification system. The system's goal was to enable citizens to prove claims relevant to the pandemic's measures to a verifier. Since these claims are highly privacy-sensitive for citizens, privacy and security were considered before gradually rolling out the system. As part of this effort, a large-scale public security test was con-

ducted based on the software's implementation. The public security test led to the discovery of 136 vulnerabilities, many of which were mitigated [182]. Figure 2.18 highlights the system's architecture, including controls such as a hardware security module preventing the exfiltration of cryptographic keys used for signing certificates.

These security efforts, however, did not completely prevent fraudulent behavior using the system: thousands of forged certificates were later discovered [242] – one single actor issued close to 400 forged certificates [80]. To commit fraud, the threat actors did not exploit a technical system vulnerability. Instead, inside personnel from private testing centers forged the certificates. This emphasizes two observations that motivate the development of an insider threat modeling solution. First, a non-technical flaw in the information system was abused. Potentially, the threat could have been mitigated or reduced by increasing procedural controls (*e.g.*, involving multiple employees, statistical checks). It is unclear why such a flaw and its system context were not discovered or discussed. Secondly, one could reason that the exploited flaw was a *design* flaw (*i.e.*, it could have been discovered without analyzing the particular protocol or software implementation). This kind of attack could – at least conceptually – be discovered using threat modeling. In addition, this case study reaffirms the findings on heterogeneous information systems described by [11]: as highlighted in the architecture of the system (*cf.*, Figure 2.18), several parties, systems, and even media (*e.g.*, PDF imports, printed letters) are involved.

Artifical Intelligence

Artificial Intelligence (AI) is a topic that has received widespread attention. In this thesis, it arises as both a problem domain and a solution. However, AI is only a loosely defined term embracing several technological paradigms (*cf.*, Figure 2.19). It generally refers to developing systems where algorithmic solutions solve complicated problems that were traditionally completed by human operators. These systems are expected to generalize concepts from data without explicit instructions [54]. Such problems can be found in several domains, leading to the expectation that AI-based technologies could help solve problems that are otherwise not effectively or efficiently solvable. Potential scenarios involving AI range from industrial manufacturing [133] to data-driven health care [183]. For example, in manufacturing, a regression algorithm could be used to predict the failure of a component. In contrast, in health care, AI could classify medical images into benign or malicious symptoms.

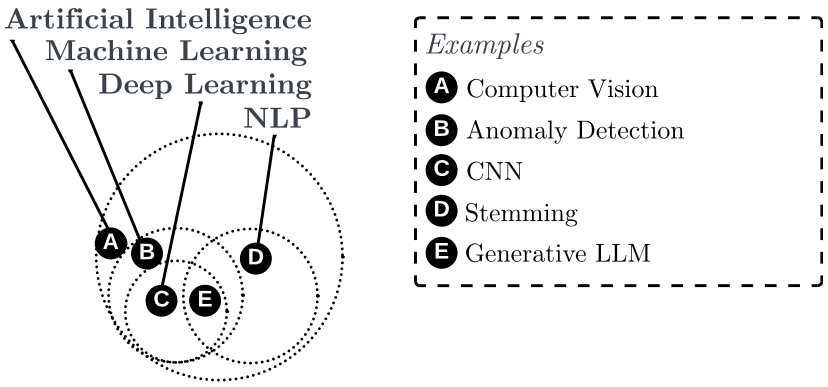

Figure 2.19: Technologies Subsumed under the Umbrella Term "Artificial Intelligence"

Machine Learning (ML) is only one subfield of AI. ML specifies the concept that involves the learning of patterns from input data using mathematical methods. Herein, two major approaches can be described: *supervised* and *unsupervised learning*. In supervised learning, the probabilities of a previously seen dataset are used to learn the probabilities of new observations. Conversely, unsupervised learning abstracts models from unlabeled input data, which can be applied to new observations. Both learning variants apply to the two fundamental problems: classification (*i.e.*, assigning an observation to a category) and regression (*i.e.*, making numerical statements about an observation) [54]. Numerous algorithms exist to implement ML. For supervised learning, algorithms include (but are not limited to) *Naive Bayes Classifiers*, *Support Vector Machines*, or *Decision Trees* [67]. Unsupervised learning can be achieved with algorithms such as *K-means Clustering*, *K-nearest Neighbors*, or *Hierachical Clustering* [54, 67].

An architectural specialization of ML is *Deep Learning* (DL), which refers to the usage of a subclass of ML methods where multiple layers (hence the term "deep") of representations are used for learning [54]. Within DL, *Large Language Models* (LLM) have sparked interest from the research and data science community and the broader population. For example, *ChatGPT*, a platform leveraging an LLM, has set a record in terms of user adoption: Two months after its launch, 100 million users were actively using the platform on a monthly basis [143]. Technically, LLMs are essentially *Deep Neural Language Models* (DNLM) that combine DL with *Natural Language Processing* (NLP) techniques (*cf.*, Figure 2.19). What specializes LLMs in the context of DLNMs is that they are – according to the definition provided by [152] – models with over 100 million parameters and are trained using more than 1 billion tokens. A token refers to an element in an embedding space, where character segments are represented as vectors. In that sense, these models are not necessarily limited to words in natural language. In addition, the trajectories performed by the models are probabilistic [152].

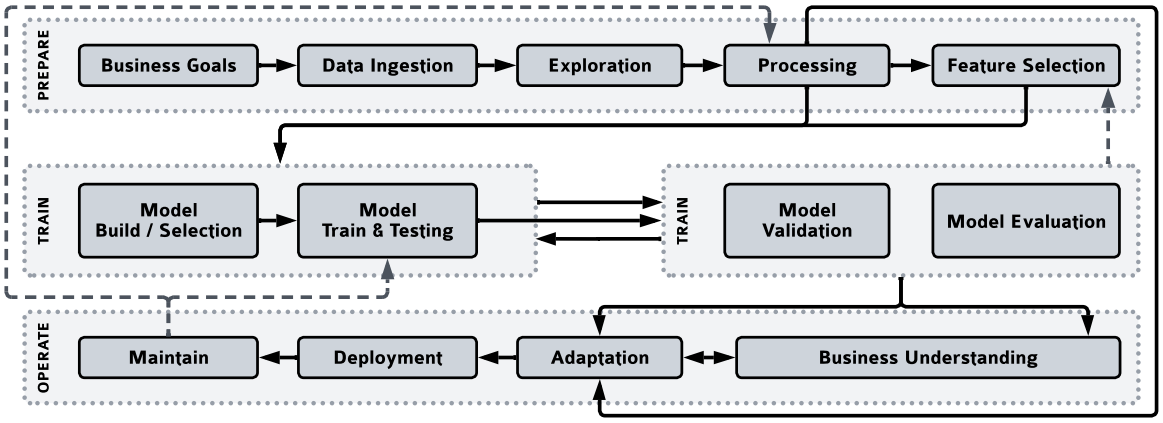

Figure 2.20: Generic AI Life Cycle [75]

To holistically discuss the security of such systems, their architectures have to be discussed in a broader context (*i.e.*, not only focusing on problems and algorithms), especially before introducing security-related aspects of AI-based systems [280]. A procedural view, such as the one proposed by the *European Union Agency for Cybersecurity* (ENISA), paints a holistic – yet technology-neutral – picture of how AI systems are developed and operated [75].

As presented in Figure 2.20, this process should be oriented around business goals. Next, the reference model highlights that these systems heavily rely on data. This data – which can come from various structured or unstructured data sources – must be ingested while considering data privacy. Data is then explored depending on the data type, enabling understanding of the data and its quality (*e.g.*, missing or highly correlated data points). Then, data is pre-processed into a format that can be readily processed while fixing defects in quality of the underlying data (*e.g.*, de-noising, filtering outliers, handling missing values). Next, potentially relevant dimensions (*i.e.*, features) are analyzed, and a reduced dataset is created.

After this step, a cleaned dataset is ready for use: one or more models are built, and their training strategy is selected based on the dataset or problem at hand. Using these algorithms, a model is trained, involving *e.g.*, the learning of model parameters. Usually, this is done on only a subset of vectors in the previously cleaned dataset. Using techniques such as cross-validation and model evaluation, hyperparameters of the selected AI model can be tested, and the resulting models are evaluated. In cases where only little data is available, an alternative flow is applicable: at training time, models are adapted and thereby refined through transfer learning – involving sourcing an existing model to which additional model training is applied.

To evaluate models, several metrics can be computed based on the model's predictions, including precision, recall, and F_1 score [47, 54]. In simple terms, precision (*cf.*, Equation 2.1) denotes how accurately the model predicts positive cases. For example, a binary ML-based spam classifier can be evaluated by observing four outcomes (*cf.*, Figure 2.21): how many e-mails that were labeled as spam (*i.e.*, positives) are actually spam (*i.e.*, true positives). Recall indicates how well the model covers actual positives. In the previous example, it would denote how many actual spam mails (*i.e.*, true positives and false negatives) were correctly classified as such. Selecting an appropriate performance measure is a critical pitfall when applying ML to security [12]. Developers can plot the precision-recall curve or combine both metrics into the F_1 score (*i.e.*, harmonic mean between both measures) [12, 54].

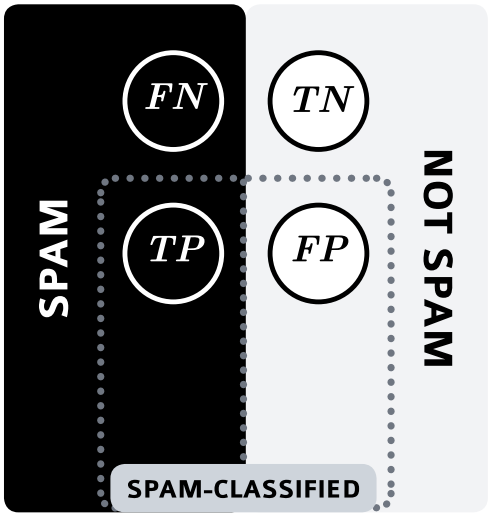

Figure 2.21: Binary Classification

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{2.1}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{2.2}$$

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2.3}$$

After the validation and evaluation of the model, an operational model is available, which must be deployed to an information system. During operation (*i.e.*, inference of the deployed model), the results can be monitored to decide whether retraining is needed. Finally, the iterative process should include an assessment from a business perspective, judging the value addition to the business. Not just subject-matter experts but domain experts must be involved. Furthermore, several of these steps may be performed by different actors, which do not necessarily come from within the organization [75] – a fact that must be considered when modeling these systems [280]. For example, data can be obtained by a data broker, or alternatively, a pre-trained model can be obtained from a model provider. In addition, a cloud provider may be involved in computationally intensive tasks

like model training or deployment. Similarly, numerous assets (*e.g.*, raw data, models, procedures, infrastructure) are involved and need to be protected when building AI-based systems [75].

Table 2.3: Threats Related to ML [152, 194, 198]

	Threat	*Tech*	*Descrption*
Usage	Prompt Injection	LLM	Carefully crafted prompts from internal or external sources lead to unexpected outputs or actions
	Data Disclosure	ML	Sensitive information is extracted (or revealed) from the model through inference
	Model Theft	ML	The model's parameters are recovered and the model is "stolen" through frequent inference
	Evasion	ML	Carefully crafted input leads to incorrect task execution (*e.g.*, avoiding fraud detection)
	Model DoS	ML	The availability of the system is threatened through targeted input (*e.g.*, complex input or frequent interaction)
	Adversarial Usage	ML	Adversaries use ML methods to circumvent security controls
	Phishing	ML	Crafting phishing campaigns using ML models
Runtime	Model Poisoning	ML	Manipulating the model, or its output, post-deployment
	Code Injection	LLM	Output leads to injection attacks in downstream systems
	Input Leakage	ML	Sensitive data in the model's input is exfiltrated
Development	Model Poisoning	ML	Model manipulation in the development environment
	Data Poisoning	ML	Manipulating training data (*cf.*, "*model poisoning*")
	Data Leakage	ML	Accessing sensitive training or validation data
	Model Theft	ML	Exfiltrating ML models
	Backdoor Introduction	ML	Manipulating data, introducing targeted triggers causing the model to take predefined actions (*cf.*, "*data poisoning*")
	Sabotage	ML	Manipulating data to disrupt the model's performance (*cf.*, "*data poisoning*")
	Supply Chain Attack	ML	Model or data manipulation during sourcing

Numerous fields have explored the efficacy of ML/DL to solve problems. Unsurprisingly, information security is one of them. For example, ML and DL concepts have been applied to malware and bot detection, user authentication, behavioral analysis, and access control. Many of these problems can be framed both as anomaly detection or as a classification problem [54]: malware could be detected by learning the patterns of data captured from executing (mostly) benign software or by creating separate datasets of benign and malicious software execution. Similarly, various data sources can be leveraged depending on the threat model [12]. While the security community has historically

focused strongly on static features such as signatures [147, 232], modern approaches combine dynamic sources, such as behavioral fingerprints [47]. Using dynamic features, detection systems could potentially be made more resilient since they are not affected by polymorphism of artifacts, as long as benign or malicious behavior does not change substantially. For example, a signature-based detection service could be bypassed by simply introducing a bit of noise. However, assuming the damage function remains similar, a behavioral detection system could still detect the malicious behavior [47].

While ML/DL approaches appear promising for cybersecurity and other fields, the security of those systems is a challenge in itself, too. Injection attacks – where untrusted user data leads to a change in control – have remained a critical threat for "conventional" systems. Mitigating them required clear separation between input (*e.g.*, user data, input) and control (*i.e.*, software) [200]. This becomes a challenge for data-driven systems, exemplified by prompt injection attacks, a fundamental LLM threat. Since LLMs operate based on the user's prompt, the line between control and data becomes blurred, especially when downstream actors (*i.e.*, LLM agents) execute operations based on the LLM's output. Table 2.3 highlights several threats in the ML domain and organizes them according to the three stages of an ML system described by [198]: development-time threats, threats arising at runtime, and threats that do not require special privileges aside from some form of model inference.

2.4 Research Foundations and Methodologies

After introducing the risk-based concept of threats and discussing several specific risks according to their problem domain, this section moves to the solution space. In the following, *design science research* will be introduced as an overarching research philosophy to provide the conceptual basis contextualizing the research methods, directions, and paradigms in solution space (*e.g.*, Case Study Research, Moving Target Defense).

2.4.1 Design Science Research

The fundamental philosophy in design science research on information systems argues that studies conducted in this space can be categorized into one of two fundamental paradigms [46, 113].

Definition 2.4.1. The behavioral science paradigms seek to explain or predict *human or organizational phenomena* involved in designing, analyzing, designing, implementing, or managing information systems. The main goal of behavioral science is *truth* – obtained from *building and testing theories*.

Definition 2.4.2. The design-science paradigm seeks to create the *artefactual innovations* which serve as a basis for analysis, design, implementation, or management. The main goal of design science is *utility* – obtained from *building and testing artifacts*.

Both fit the definition of what it means to conduct a *research* activity, *i.e.*, to seek valid knowledge for the community. Here, validity refers to the agreement by the research community on values (*i.e.*, research paradigms) that produced this knowledge [46]. Research in information systems (as a field) is focused on advancing knowledge that ultimately contributes to information systems that improve the effectiveness or efficiency of the implementing organization. The *behavioral* paradigm has its background in the natural sciences and does so by explaining organizational or human phenomena. For example, a behavioral approach applied to cyber risk management could inquire why human analysts underestimate certain risks, leading to the discovery of psychological biases that affect decision-making in risk management (*cf.*, Section 2.1.5). These behavioral research activities explain the interactions between people, technology, and organizations. Such findings directly impact – and are affected by – design decisions [113].

The *design science* paradigms have their background in engineering and are ultimately a problem-solving discipline. This paradigm requires creative research activities to yield innovations. These innovations must be created by taking into consideration the findings from the behavioral (and natural) sciences. The development of artifacts in design science research highlights that the two paradigms are, in fact, complementary. For example, a potential design study could incorporate the previous findings about psychological biases in risk management to design, implement, and evaluate a novel risk management methodology that makes users aware of biases during risk assessment. Thus, research in this paradigm seeks to develop and test useful innovations [46, 113]. However, this complement can also be executed in the opposite order, where an artifact is found to be useful for a reason not yet discovered in theory – motivating behavioral research studies.

In summary, at the core of the *Design Science Research* (DSR) is the development and testing of an artifact. [113] prescribes that four types of artifacts can be designed:

A1) *Constructs* enable problem and solution space instances to be defined and communicated.

A2) *Models* enable the understanding of problems and solutions from the real world by using and relating constructs.

A3) *Methods* define a series of steps (*i.e.*, processes) to navigate and apply the solution space.

A4) *Instantiations* implement one or more of the previous artifacts to demonstrate feasibility and enable assessment within an organizational environment.

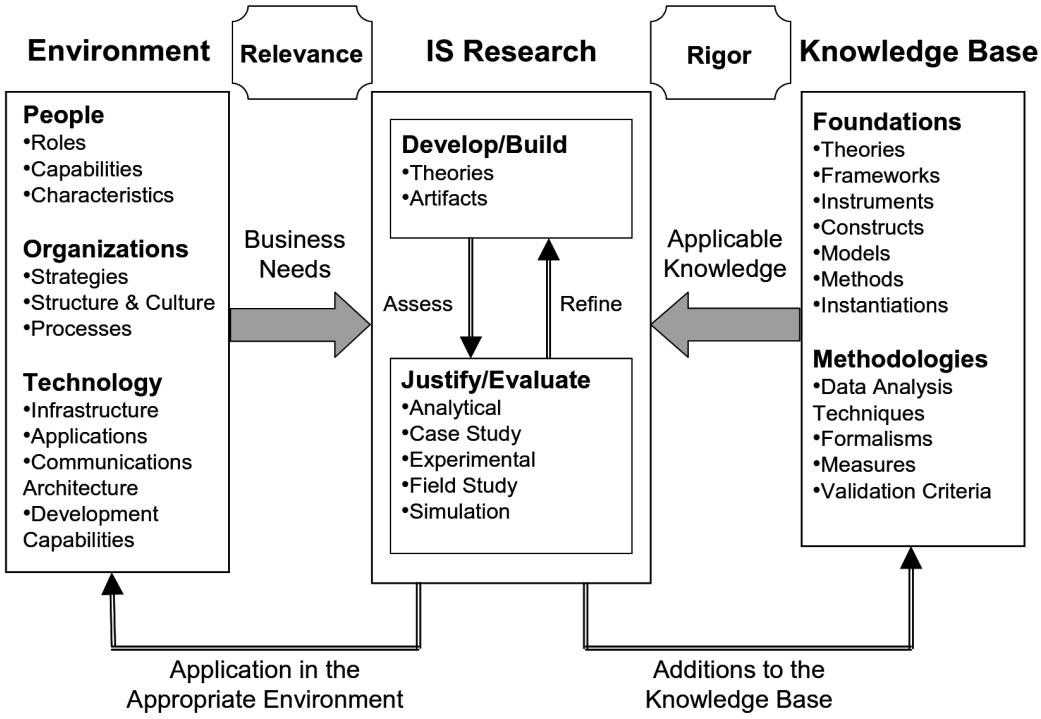

Figure 2.22: Design Science Research [113]

Designing such an artifact is only then considered novel (*i.e.*, contributing to advancements in research) when it either addresses a novel problem, leverages a novel solution design, or improves the effectiveness or efficiency of a design. If none of these factors are given, the activity is classified as system building, where knowledge is routinely applied to known problems [113]. For example, [97] proposed a mathematical model applying economic theories to analyze the optimal investment into cybersecurity. [89] did not change this model but implemented a web-based prototype to apply and visualize the theoretical model – laying the foundation for assessing it using usability evaluations and case studies, which yield new knowledge about the existing model [113].

Figure 2.22 visualizes the conceptual framework of relevant and rigorous design science research. At the core is the main task of a DSR researcher: building an artifact and evaluating it. To do so, the researcher must consider the problems, goals, and motivations of the people, organizations, and technology of an IS. Without this consideration, the artifact may not be *relevant*. On the other hand, the researcher must ensure *rigor* by considering the research community's knowledge base. This is done by analyzing and applying existing foundations (*i.e.*, previous research) and methodologies (*i.e.*, guiding processes to evaluate results). The researcher considers both spaces (*i.e.*, problem, and solution

space) and contributes to both of them by applying the artefact to the problem space, yielding new knowledge for the knowledge base [46, 113].

From a practical viewpoint, conducting relevant and rigorous studies can be conflicting: over-reliance on rigorous methods can weaken the relevance of the business context, and specializing in highly specific environmental questions can question the generalizability of the knowledge base. Nevertheless, both the design and evaluation phases must effectively use the knowledge base. To design an artifact, researchers must be aware of the existing foundations (*e.g.*, theories, frameworks, models, instantiations) to create a novel and effective artifact. To evaluate the artifact, the researcher must know and apply existing methodologies. Since the goal is to demonstrate the utility of the artifact, these methodologies cannot be reduced to quantitative methods. Various methods must be considered, including observational (*e.g.*, case studies, field experiments), analytical (*e.g.*, static or dynamic analysis), experimental (*e.g.*, simulation, experiment), testing (*e.g.*, functional or structural), or descriptive (*e.g.*, informed argument, scenarios). The following sections introduce the foundations and methodologies related to the research performed in this thesis [113].

2.4.2 Research Foundations

Research foundations describe existing knowledge on how to approach the design of an artifact. As such, foundations provide the raw materials to construct new artifacts and apply them to a particular problem. In the following, the ideas, philosophies, and theories behind existing foundations are explained. These foundations are used to build the solutions proposed in this thesis. Process modeling, business impact analysis, and threat assessment are introduced, which relate to research on threat modeling and quantification. For the research lines covering attack detection and mitigation, conceptual foundations for data extraction, Moving Target Defense, and attack deception are provided. A narrower introduction of the *results* previously achieved with these foundations regarding particular *problems* will be presented in Chapter 3 – motivating the research directions of this thesis. The methodologies used to conduct research using these foundations are described in Section 2.4.3.

Process Modeling

Several modeling languages are available to model interactions between stakeholders and systems procedurally. However, the *Business Process Model and Notation* (BPMN) represents the dominant standard supporting three classes of use cases: describing, simulating, and executing processes. Intuitively, users may have a rough understanding of the execution of a process. Descriptions can, however, be made more detailed and explicit through diagrams. The underlying models expressed

through the diagrams can then be used for analysis to identify defects, such as inconsistencies or inefficiencies. Many use cases go as far as to translate – using the standardized modeling language – so it can be semi- or fully automatically executed. However, the aspect most relevant to the thesis remains the analytical capabilities of BPMN models, enabling experts to analyze the underlying process model or the execution data. More explicitly, a business process can be defined according to the following definition [53].

Definition 2.4.3. A *Business Process* (BP) is a set of one or more linked procedures or activities executed in a predefined order that collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles or relationships.

Business Process Modeling (BPM) – not to be confused with the acronymically similar idea of Business Process Management – refers to the manual or automated description or modification of process models. This activity involves describing both the current state of processes and the design of a new or modified process. In a small-scale survey conducted by [53], 40% of respondents modeled processes using the grammar defined by the BPMN 2.0 standard. This version effectively extends BPMN 1.2 through features such as an extensibility mechanism and new constructs. However, many key concepts are already described by the previous standard. Six categorical concepts are refined through several elements modelers employ to represent processes: events, gateways, swimlanes, artifacts, and connecting objects [53]. These are described and exemplified through Table 2.4.

Business Impact Analysis

Business Continuity Management (BCM) takes a holistic and resilience-oriented perspective at risks that organizations are inevitably facing and assumes that it is impossible to mitigate their occurrence completely. Instead, business disruptions are formulated explicitly, enabling the discussion of what

Table 2.4: Overview of BPMN 1.2 Elements [53]

Category	*Description*	*Example*
Events	Define trigger	Store opens, receive letter
Gateways	Deciding Flow	Parallel execution, verify property
Swimlanes	Grouping elements	External employee, internal employee
Artefacts	Static information	Text comment, database entry
Activities	Sub-procedure	Compute discount, supervisor approval
Connections	Connecting elements	Sequence flow, message flow

can be done in case of an actual disruption. Thus, the key goal is to reach maturity concerning processes that will allow a best-possible return to "normal operations", *i.e.*, discussions focus on achieving organizational resilience. Traditionally, BCM embraces the understanding that risk factors are diverse (*e.g.*, earthquakes, cyber attacks) [50].

Implementing a BCM program involves three high-level steps. First, risks to the organization are identified – highlighting that BCM is fundamentally risk-based. Then, the risk is reduced by developing preventive measures (*i.e.*, preventing the occurrence of a risk event or the materialization of a risk's impact). Thirdly, the involved stakeholders are trained, and the plan is tested [50]. Inherently, the analytical first step takes a business-oriented view on assessing risk. One methodology employed to identify risks and discuss their criticality concerning dependent business functions is *Business Impact Analysis* [50, 257]. [257] proposes a four-step methodology to analyze business impacts.

Since BCM focuses on strategic risks, identifying essential products is the first step. Due to the complexity of businesses, a structured approach is needed, such as performing measurements based on potential revenue or reputation loss. Secondly, the business functions enabling these products must be identified by applying a Work Breakdown Structure that enables decomposition regarding departments, costs, or resources. The third step involves identifying the functions that represent a critical path for the product. For example, a company could investigate which functions would have to be recovered first. Finally, the continuity parameters can be established to model the functions' resilience against a defined risk appetite. Two measures are essential for this task: the *Maximum Tolerable Period of Disruption* (MTPD) and the *Minimum Business Continuity Objective* (MBCO). As implied, MTPD defines a target time interval until a function's disruption must be resolved. MBCO, on the other hand, represents a minimum threshold of acceptable operating level [257].

Calculating or estimating losses is pivotal to identifying critical products, their supporting functions, and the impacts of risk events. These losses can be either tangible or intangible. Tangible losses are directly related to specific income streams and are considered easier to calculate or estimate since a materializing risk would directly cause a loss effect. For example, if a web shop is unavailable due to its infection by ransomware, any downtime leads to a direct loss of revenue. However, there can also be indirect, tangible losses that are harder to approximate. Essentially, this category covers tangible losses not incurred by the revenue stream. For example, a regulatory body could impose a financial penalty after the incident is resolved. The hardest to calculate are intangible losses, which do not directly or measurably represent a cost factor. For example, due to the previous incident, the reputation of the company brand could be decreased [172].

Threat Assessment and Statistics

In risk and threat assessment, approaches can be distinguished into qualitative and quantitative ones. As will be surveyed in Section 3.1.6, threat modeling tools and methods still heavily center around the usage of ordinal scaled scores (*cf.,* Section 2.1.5). Conversely, risk assessment – especially when looking beyond the cybersecurity domain – has demonstrated the adoption of quantitative approaches. Still, ordinal-scaled methods and scores are widely found, even in cyber risk management [117, 120]. In a survey described by [117], 61% of organizations leverage risk matrices to represent and convey risks. The same underlying concept of probabilities and impacts is expressed in a risk matrix on an ordinal scale, yielding a two-dimensional matrix. For example, the probability of a threat's occurrence could be qualified into *unlikely*, *likely*, and *probable* while impacts could be assigned to either *low*, *medium*, or *high*. Summarizing the wealth of previous research on the potential fallacies of such matrices, including psychological biases, inability to express uncertainty or distributions, or misleading interpretations [117, 120, 273], would be beyond the scope of this section. Only key aspects of the quantitative approach subject of this PhD thesis' research are presented, while Section 3.1.6 provides an in-depth review from the viewpoint of the threat modeling practice.

Any risk-based approach inherently deals with uncertainty [120] and thus with imperfect information [24]. To develop quantitative approaches, the highest-quality data should be used for decision-making [130]. Practically, this entails using historical data combined with sound mathematical methods. However, even for quantitative approaches, subjective estimates must be considered and explicitly modeled using mathematical models [117]. For example, in daily life, we may not know how many grams of potatoes will be needed for an upcoming meal. Still, once an estimate is developed, one would measure the produce at the store, instead of estimating it by eye.

Instead of producing a plain number (*e.g.*, a threat has a potential impact score of three out of ten, or an "unlikely" occurrence), distributions express the probability of different outcomes. The first distribution to introduce is the *Poisson probability distribution* [10, 117]. This distribution considers a discrete random variable and is commonly used to model occurrences of an event. For example, in an incident response program it could be modeled how many incidents (*i.e.*, they can be counted using non-negative integers) are resolved per week through the response team. Formally, the Poisson distribution can be defined in Equation 2.4 for the discrete random variable x, where $f(x)$ represents the probability of x occurrences in a given interval and μ denoting the expected value of occurrences [10].

$$f(x) = \frac{\mu^x e^{-\mu}}{x!} \tag{2.4}$$

The second related distribution is the *normal distribution*, commonly employed to model continuous random variables. Figuratively, the normal distribution resembles a bell-shaped curve centering around the mean and can thereby be described by the mean μ and the standard deviation σ, as described in Equation 2.5 [10].

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{2.5}$$

Since the normal distribution could also be used to model negative values, the *lognormal distribution* can be preferable for certain problems. Essentially, the lognormal distribution is a simple variation of the normal distribution, where the logarithm of the variable is normally distributed. This can be practical since it presents two properties: it is strictly non-negative and is skewed with a "long tail", as illustrated in Figure 2.23. This makes it applicable for cases where negative values are illogical and where outliers can potentially be far from the mean [117]. The former property is especially useful in cybersecurity – where risks are usually contextualized as operational losses (*i.e.*, there is rarely an upside risk) and thereby strictly non-negative without a logical upper bound. For example, the expected cost of implementing a new defense system will unlikely be negative. Furthermore, although in most cases (*e.g.*, in 90% or 95% of observed or predicted cases) the cost would be close to an expected range, there may be rare cases where the project significantly overruns the budget.

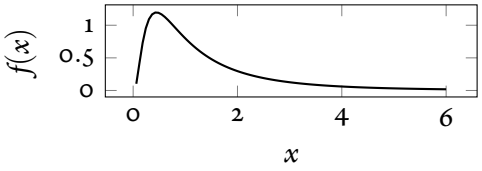

Figure 2.23: Hypothetical log-normal distribution with $\mu = 0$ and $\sigma = 0.5$

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{ln(x)-\mu}{\sigma}\right)^2} \tag{2.6}$$

When constructing such probability distributions based on expert estimates, psychological biases [120] and statistical literacy [117] must be considered. For example, estimates are obtained by interviewing analysts about the values describing a confidence interval of 90% [120]. The bounding values do not represent the worst and optimal cases – there may still be a relevant number (*i.e.*, 5% in this example) of cases outnumbering this upper bound estimate.

Using distributions provides a way to represent outcomes of particular scenarios (*e.g.*, different impacts of a threat, or different arrival rates of incident cases for different processes). An approach to combine scenarios involves the usage of *Monte Carlo simulations*. This requires a relatively simple approach of randomly sampling values according to the defined distributions and convolving them

into new scenarios [117]. Based on the law of large numbers, many simulations must be performed to approximate the simulated scenario. Providing a clear-cut, formal definition of Monte Carlo simulations is challenging. However, the usage usually follows the following three-step pattern [109]:

Step 1) Approximate the scenario by modeling it with probability distributions

Step 2) Compute sample values that follow the distributions

Step 3) Tally samples and statistically describe the resulting scenario

For example, Listing 2.2 demonstrates how to approximate the value of π, by randomly sampling coordinates and computing whether they fall within a circular sector in a square. The proportion of the two outcomes is used to approximate π.

Listing 2.2: Approximating π by Randomly Generating Points in a Square

```
import numpy as np

def monte_carlo_pi(number_of_simulations):
  side_length = 5.0
  radius = 5.0

  # step 1+2) define uniform distribution and randomly generate
  #           points by sampling X and Y coordinates
  x = np.random.uniform(0, side_length, number_of_simulations)
  y = np.random.uniform(0, side_length, number_of_simulations)

  # step 3) compute how many points fall within the circular
  #         sector and approximate Pi
  distance = np.sqrt(x**2 + y**2)
  inside_circle = np.sum(distance <= radius)
  estimated_pi = inside_circle * 4 / number_of_simulations
  return estimated_pi

number_of_simulations = 1000 * 1000
pi_estimate = monte_carlo_pi(number_of_simulations)

print(f"Approximating pi: {pi_estimate}")
```

In other risk management domains, Monte Carlo simulations are an established tool [117, 120]. In cybersecurity, this is not necessarily the case. This highlights that, in addition to psychological biases and statistical literacy, the acceptance of quantitative methods by an analyst remains a relevant

consideration. In [117], 75% of respondents agreed that the cybersecurity domain should eventually use probabilistic approaches. However, out of the group described as "*anti-quantitative group*", only 4% had used the method. This contrasts with the performance achieved when involving human analysts. In a study conducted by NASA, a group of non-experts outperformed a group of subject-matter experts in assessing risks: the non-expert accountants had access to historical data and Monte Carlo simulations. In contrast, the subject-matter experts used a modified version of a 5×5 risk matrix [117].

Data Extraction in Cybersecurity

For detection and response, anomaly detection and pattern classification of cyber attacks is vital. For example, the previously introduced MTD paradigm includes reactive techniques, where a signal is needed to deploy a mitigation strategy. In front of this background, data mining comprises the systematic extraction, understanding, and modeling of data. For data-driven applications in cybersecurity, different data sources and numerous features exist [37].

Data extracted from network communications is a widely used data source for cybersecurity. By capturing network traffic, information about the communication partners can be extracted [219], enabling feature extraction across all levels in the OSI model [37]. From the network-layer, source and destination address can be extracted [37], while the physical layer presents data on *e.g.*, signal frequency or strength. Arguably, the majority of relevant devices in cybersecurity are connected to a network – making network-based data extraction a universal data source. These approaches are classified as "out-of-device source" [219], allowing data to be extracted without relying on the compromised device and thereby on a potentially inappropriate defense model [12]. However, two challenges must be considered when implementing an actual data extraction pipeline. First, the actual capturing approach (*e.g.*, relying on full packet captures or flow-based abstractions) may impose computational and data quality requirements and constraints [37]. Secondly, network-data may not be trustworthy (*e.g.*, impersonated or randomized) and encryption can hinder data extraction without imposing privacy tradeoffs [219].

In cases where network-based approaches are not viable due to the presented data granularity or architectural requirements (*e.g.*, a network-enabled monitoring or processing device), "in-device" sources can be considered, too. Lower-level data can be extracted by gathering data directly on the target device. For example, hardware events or system execution can present precise information about the specific hardware platform and its behavior. Concerning the runtime, resource usage indicates the consumption of workloads relating to resources, including CPU, memory, storage, or communications. Monitoring through system calls and logs moves the observer closer to the user-space

process. Here, the behavior of the software and its interaction with the *Operating System* (OS) can be approximated. Many of these approaches can be used to understand not only artefacts through static features (*e.g.*, signatures, software components) but also to build behavioral profiles. For example, a static approach could identify which system calls appear in a software artifact compilation – a behavioral approach could create a profile indicating in which sequence system calls are executed. Thereby, these approaches can paint detailed pictures of a device's execution behavior [219]. However, it must be considered whether the approach for data extraction is aligned with the threat and defense model [12, 219].

Moving Target Defense

In static cyber defense, where a system is being analyzed and protected at one point in time, an inherent asymmetry emerges between attacker and defender. Since the system does not change, an attacker can take as much time and resources to try to exploit a system. With enough time, an attacker will likely succeed since the system is unlikely "perfectly secure". This asymmetry is acknowledged as a key limitation in the *Moving Target Defense* (MTD) paradigm. Theoretically, MTD does not address the question *if* a system will be breached, but instead, *when*. Practically, MTD proposes to change the underlying static nature of a defense system and, instead, introduce dynamicity into the defense approach. A system is turned into a constantly changing one, potentially reducing the time when a system is vulnerable to a certain attack vector. In that sense, MTD does not aim to make a system perfectly secure. Instead, it increases the complexity for an adversary and reduces the time that an attacker has. A viable attack path thereby "*only works once, if at all* [181]."

To implement an MTD technique, the attack surface of a system (*i.e.*, the part of a system that an adversary can attack), must be identified. Traditional security approaches focus on reducing that surface. In MTD, dynamicity is introduced by developing a method that changes system components so that it introduces variability. For example, a web service may expose a set of TCP ports (*i.e.*, the attack surface). Traditionally, access to these ports would be reduced (*e.g.*, based on ACLs or geographical locations) – in MTD, the port could be changed at regular intervals.

Research on MTD covers three areas: theory, strategies, and evaluations. Whereas theories describe the overall attack-and-defense models of MTD, strategies describe concrete MTD techniques. Evaluations primarily focus on developing metrics to assess strategies [55, 181]. From a practical design perspective, MTD techniques are key to apply this defense paradigm to a particular problem space (*e.g.*, IoT, 5G networks). Fundamentally, the definition and instantiation of an MTD technique requires that the following questions be answered:

What? Defines the component of the system or attack surface that is moved.

How? Describes the set of valid states between which movement is possible and the method to move between them.

When? Refers to the decision process that is followed to trigger a state change.

"*What*" is closely related to the problem definition and the particular scenario employed since only a subset of the existing attack surface can be used to craft a technique [181]. For the "*how*", three variants exist: shuffling between valid states, increasing the amount of distinct states (*i.e.*, increasing diversity, or increasing the number of system instances (*i.e.*, increasing redundancy). To define the transition point (*i.e.*, the "*when*" pillar), a signal is needed based on which to act. This signal can refer to either a point in time or the occurrence of an event. Since the timing strategy may be defined beforehand, such approaches could be considered as *proactive*, and event-based ones as *reactive*. The combination of which is described as *hybrid* approaches [181, 181].

Currently, MTD as a discipline has received attention from the research community to provide security in several areas, including the IoT, Cloud systems, operating systems, and Software-Defined Networks. However, the following limitations and research directions are described for the paradigm as a whole [55, 181]; Section 3.2.2 investigates them on the basis of specific solutions.

1. Many solutions and studies focus on a singular MTD technique and do not answer how defenders can coordinate multiple defense techniques.

2. Solutions are often not implemented beyond the design stage or confined to unrealistic testbeds or simulations, paving the way for testing MTD techniques in realistic scenarios.

3. Reconnaissance attacks represent a major focus in research on MTD; techniques going beyond this phase of an attack are less commonly described.

4. Few solutions integrate and investigate state-of-the-art detection systems for reactive defense.

5. MTD in IoT may exhibit an overreliance on network-based techniques instead of other attack surface areas such as runtime, software, or platform.

Cyber Deception

One of the effects of MTD is the resulting confusion of an adversary. This implies that MTD is overlapping with the *cyber deception* paradigm [55]. In deception, the defender provides false or misleading information to increase confusion for the attacker. Deception and MTD are, therefore, complementary approaches [55, 284] that can overlap when false information is provided as part of the real

attack surface [55]. For example, making up a completely arbitrary TCP port would be classified as deception since the port is not part of the real attack surface. Practically, cyber deception is described by two phases: maximizing knowledge and information about the adversary's intent, capabilities, and motives and developing a deception scheme that presents a combination of misleading and factual information about the asset [55, 284].

Cyber deception follows an opposing understanding of the defender-attacker relationship: the defender knows more than the attacker (*e.g.*, the defender knows how the system works or which information is correct). Any areas that the defender and attacker know can incorporate selectively releasing information, combined with fake information. Areas that the defender knows well but not the attacker should be minimized and are thereby aligned with "conventional" security approaches. Finally, areas known to the defender but not to the attacker are likely the target of reconnaissance attacks (*cf.*, Figure 2.15). Here, the defender should focus on deceiving attackers, denying information, or selectively releasing it [284].

In addition to the attack-defense relationship, the design of a deception technique must obey the domain's fundamental laws wherein they are implemented. Deception executed in cyber space can be complemented by physical or social deception. In any case, time, space, or speed constraints must be obeyed to define an effective technique [284]. A deception technique that mimics a human interacting with a phishing e-mail must conform to the constraints of how a human interaction could be characterized, *e.g.*, mimicking a human opening 100 e-mails per second on a honeypot system is unlikely a realistic deception.

2.4.3 Research Methods

Conducting research on any of the previously iterated research foundations requires the application of sound methodologies to justify and evaluate emerging artifacts. Scientific rigor requires the appropriate selection of both the foundations (*i.e.*, prior research directions) and methodologies, considering how applicable and expressive a methodology is for a given artefact [113]. In this section, eight methodologies (*cf.*, Table 2.5) employed throughout this thesis are introduced.

Literature Review

As previously described, the knowledge base (*i.e.*, research foundations and methodologies) must be known to the researcher. This is a non-trivial task due to the rapid speed at which this growing body of knowledge is produced by the research community. In addition, information may be fragmented across fields, requiring the researcher to synthesize knowledge – motivating the relevance for *liter-*

Table 2.5: Research Methodologies [113, 157, 235]

	Methodology	*Descrption*
OBS	Case study	Deploy and observe the artifact in problem space
	Field experiment	Observe practitioners solve pre-defined problems using the artifact
	Focus groups	Observe interactions and statements of a group of participants
ANA	Static analysis	Structural analysis of the artefact
	Architectural analysis	Investigate feasibility of integrating the artifact
	Dynamic analysis	Analyze dynamic properties (*e.g.*, behavior, capacity) of the artefact
EXP	Controlled experiment	Study artefact in controlled environment
	Simulation	Operate artifact with synthetic data
TES	Black box	Identify defects by testing against interfaces
	White box	Test by studying "internal" implementation
DES	Informed Argument	Discuss artifact's utility using existing knowledge base information
	Scenario	Illustrate and demonstrate artifact's utility in constructed scenario
	Literature Review	Synthesizing knowledge base from literature

OBServational, ANAlytical, EXPerimental, TEsting, DEscriptive

ature reviews as a methodology [235] and written communication as a critical stage of DSR [113]. Three approaches exist to conduct a literature review: systematic, semi-systematic, and integrative. Common to all of them is a high-level approach comprising four stages. First, the study is designed by settling the relevance, purpose, audience, scope, and method of the review. Secondly, the actual review is conducted by following the design and yielding the relevant content of the research body. Thirdly, information is synthesized based on the approach employed. Finally, the synthesized knowledge is communicated back to the research community through writing, highlighting the study's motivation, scope, relevance, methodology, and actual contributions.

The first approach to implement these four stages is a systematic literature review. Systematic literature reviews are narrower regarding their research questions and methodological strictness. This method answers a specific research question by synthesizing evidence, requiring a highly systematic search strategy that targets the evidence presented artifacts while focusing primarily on quantitative results. For example, these studies could present synthesized evidence pointing to an effect [235].

Semi-systematic approaches- the ones most relevant to this PhD thesis- take a broader perspective on literature review, as exemplified through broadly formulated research questions. These approaches are commonly employed to present an understanding of a research area and track its development. The employed search strategy may (or may not) be less systematic and consider qualitative work, too.

For example, a semi-systematic literature review could highlight a trend in methods (*e.g.*, data mining, ML) being employed in a particular field (*e.g.*, network management, health care) [235].

Integrative methodologies may not fit the descriptive categorization of literature reviews since they contribute new research through in-depth analysis. Instead of highlighting a research trend (*i.e.*, established through semi-systematic approaches), new artefacts are derived from the literature, such as taxonomies or theoretical frameworks. These methods often comprise qualitative analysis, include a varied set of sources (*e.g.*, books, articles), and target narrow or broad research questions [235].

Dynamic Analysis

Literature reviews provide an approach to approximate the current knowledge base through literature. However, this confines the analysis of previous research to the manuscript written by the authors. Concerning the research conducted in this thesis, additional information was required to understand the current body of research and its underlying problem domain. For example, in threat modeling (*cf.*, Section 3.1.6), the knowledge base was approached by considering artefacts (*e.g.*, models, applications) from both the academic community and the industry. As will be described, due to the complexity of these artifacts, it is argued that not every theoretical research question can be answered solely through the literature.

To overcome this limitation, methods from software testing are employed. Dynamic testing is crucial in software development, focusing on executing code to uncover flaws and ensure the system meets its predefined requirements. This methodology involves running software with different inputs to observe its behavior and outputs. Unlike static testing, which evaluates code without execution, dynamic testing provides insights into the software's performance under various conditions [111].

Two approaches exist: black box and white box analysis. These differ primarily in their focus and methodology. Black box analysis examines the software's functionality without delving into its internal structure [111]. For example, in this thesis, commercial threat modeling tools rarely provide open-sourced access, and only the functional interfaces (*e.g.*, Application Programming Interfaces or User Interfaces) were accessible to understand their functional contributions. In summary, dynamic analysis testers evaluate the system's responses to various inputs to investigate its behavior against requirements. This method is user-centric and does not require knowledge of the source code, making it ideal for validating user interfaces and overall system functionality. White box dynamic analysis, on the other hand, or structural testing, involves examining the internal structure and workings of the software through execution. This method requires knowledge (and accessibility) of the source code and enables analysis of implementation details. For example, it can be investigated whether a particular requirement translates into executing a specific sub-component (*e.g.*, library, code path) [111].

Surveys and Questionnaires

Questionnaires and surveys are advantageous for two reasons. First, when neither the execution of an artifact nor a description in the literature provides information on a defined research question, surveys and questionnaires can give meaningful answers. Secondly, if stakeholders' perceptions are the subject of interest, these methods pose a valuable tool if they are rigorously designed and executed. Conducting a survey is composed of its design and analysis [18, 226].

The first step towards designing a survey is preparatory. It must be considered whether a survey is appropriate at all. The researcher must decide whether to use a self-completed form or a guided interview. Depending on this decision, additional considerations must be made. For example, when designing a self-completed form, the researcher must carefully decide whether questions are formulated in an open or closed way. In addition, the response format and scaling must be defined. For instance, a closed question may be answered on a Likert or binary scale. Since surveys seek subjects' perceptions, the sampling of subjects must be carefully targeted. If a particular sub-group (*e.g.*, "how do CISOs view quantitative risk methods?") is the target of a study, it must be targeted systematically [18].

Before delivering a survey, the study design can be evaluated by involving a subset of stakeholders. To provide the survey, the delivery method can be tailored to the targeted audience (*e.g.*, postal mail, electronic mail). Similarly, the time of delivery must be decided. Once the survey data is collected, analysis must be carried out using methods appropriate to the data collected. Quantitative data can be analyzed using statistical methods, such as computing the median and reporting it along with other metrics, such as standard deviations, while discussing outliers. If free text is gathered, the resulting data must be coded (*i.e.*, sorted into themes) and reported (*i.e.*, developing free-text descriptions or graphical illustrations) [18].

Experiments and Simulations

Once an artifact is developed based on its motivation through the instances in the problem and solution space, the artifact must be evaluated. Experiments and simulations are research methods commonly used for evaluation in DSR [113]. Fundamentally, experiments involve the development of a hypothesis and the construction of two groups, one of which the hypothesis is tested by controlling a factor. For example, in medicine, one would test the hypothetical efficacy of a drug by administering it to an experimental group, while the control group does not receive the treatment to enable comparison [157]. This has a fundamental assumption: experiments require a certain amount of control over the experimental setting. While this may increase the reproducibility of the study – since another re-

searcher could potentially recreate a similar setting by enacting similar controls – this highlights that the experimental setting is partially artificial since control implies intrusion [121].

The literature describes both *laboratory* and *field* experiments. However, field experiments have been characterized as being more observational than experimental [113]. As argued for by [215], the research community may separate these two methods by their environment, *i.e.*, laboratory experiments test in a highly controlled, artificial environment. In contrast, field experiments test in "the real world". However, as argued by [215], the line between both methods may be blurry, and the two methods could be used in a complementary way since they both carry comparative advantages. Both methods can be used to understand effects in a defined setting. While these methods help answer "what, how, and why" did an effect happen, in highly complex systems, *simulations* are experiments that help answer "what if" questions. As previously introduced, simulations can not only be used as a research method but also as an actual problem-solving method. For example, the previously introduced Monte Carlo simulation method could be used to model uncertainties in an organizational problem, using the simulated outputs for decision-making. Overall, three types of simulations can be conducted. First, in discrete event simulation, an organization is decomposed into modeled entities evolving over time while considering resources and events. Secondly, system dynamics model the behavior of a system in a stateful manner. Thirdly, agent-based simulations maximize an agent's perceived utility by interacting with other entities and resources [69].

Focus Groups

Focus groups share similarities with experimental and observational methods: researchers and moderators can enact a certain level of control by designing the setting and influencing the involvement of participants. However, during the execution of a focus group, only limited control can be enacted, and the role of a moderator may be mostly observatory. Generally, focus groups are formally organized, structured groups of individuals gathered to discuss a topic or series of issues in a particular setting. This method can be used to obtain individuals' subjective perceptions, similar to a survey method. Moreover, they add the dimension of interaction to the setting – which can be observed between participants and between participant and moderator [121, 157].

There is no definitive design approach in order to leverage focus groups. Usually, six to ten participants and a moderator participate. The selection of participants often follows a pre-defined problem, *i.e.*, they can be selected in similar ways as when using the survey method. For the success of this research method, several human factors are critical, such as the competence of the moderator and the development of a suitable environment (*e.g.*, respecting privacy, establishing communication rules). Substantial preparatory efforts are needed to ensure that participants, the setting, and the scope (*i.e.*,

topics and questions) lead to the collection of valuable observations. If successful, the data gathered through a focus group can provide open and unrestricted perceptions of individuals – backed by discussions and interactions. Due to the involvement of small and non-randomly sampled participants, perceptions can be biased, and the researcher needs to use caution when generalizing [157].

Case Studies and Scenarios

Moving closely to the problem space, *case studies* are introduced. The researcher almost completely avoids the experimental philosophy of the previous methods. Instead, no control is enacted on the subject. The case study approach, therefore, relies on the in-depth analysis and observation of a single (or small-scale) case. Organizations, individuals, or time could represent the boundary of this case. Case studies appear orthogonal to the experimental methods – where a hypothesis is tested on a sampled subset of the population – both in breadth and depth: instead of answering a single question with narrowly scoped information from many samples, rich data obtained from a single case is studied, presenting in-depth information, that may even contradict the initial hypothesis [157].

This makes case study research complementary to other research methods [157]. For example, in a quantitative experiment, the performance of a malware classification system is being tested. The results indicate a system successfully detects 999 out of 1000 malware samples. Hence, the experiment is concluded, attesting to the high accuracy of the approach. However, one may wonder what makes the undetected sample particular. Indeed, this question cannot be easily answered, especially without a hypothesis about why the classifier failed. Here, a case study could involve the in-depth analysis of the artefact – highlighting that case studies can not only directly contribute new knowledge (*e.g.*, static or dynamic properties of the sample) but also new hypotheses or research directions.

Providing a clear-cut definition of the procedural steps is challenging. Nevertheless, any case study must define how the case (*e.g.*, individual, group, institution) is bounded. In addition, it must involve detailed collection and analysis of observational, often retrospective, data. This exemplifies a relevant advantage of case studies, which lies in their flexibility (*i.e.*, being able to explore various directions that are guided but not confined by the research question) [157]. However, the researcher must be aware of its limitations. First, case studies include substantial bias introduced by the observer – in DSR, the researcher is often even considered as being "participatory" to the case [121]. Secondly, they expose limited generalizability because a tiny fraction of the population is analyzed [157]. Finally, it is essential to consider what case studies are not: case studies are observational studies of real cases – *illustrative scenarios*, on the other hand, demonstrate the utility of an artifact in a constructed scenario [113]. As such, the latter can involve artificial data, while the previous does not.

Truth informs design and utility informs theory.

— Alan R. Hevner, Salvatore T. March, Jinsoo Park, Sudha Ram (2004)

3 Related Work

Having defined the recurring concepts of both the problem and solution space, this chapter presents a survey of state-of-the-art approaches applying solutions to these domains. Three areas are investigated with respect to limitations and research directions. First, the threat modeling field is investigated by introducing methods and solutions from both the research community and the industry. Herein, limitations are elicited with respect to their support for stakeholder collaboration, their ability to guide threat modeling for new technologies (*i.e.*, the emerging AI systems engineering field), as well as their applicability to include human factors (*i.e.*, instantiated through their consideration of insider threats). The subsequent section surveys solutions dedicated to threat and risk assessment while considering threat modeling techniques that embed threat assessment methods. Furthermore, research on threat detection and mitigation from a behavioral perspective is surveyed. This includes data collection approaches and responsive actions leveraging the MTD and deception paradigm for proactive and reactive defense.

3.1 Threat Modeling

Research and practice on threat modeling relevant to this literature review can be attributed to one of the following types. First, methodologies have emerged from either of the two sources. They describe the guiding concepts and how to apply them procedurally. Secondly, threat modeling tools aim to make the application of threat modeling methods more efficient or effective through aspects such as increased automation, methodological guidance, or the provisioning of knowledge. These two categories cannot always be clearly distinguished, with some tools representing both an application and a new methodology. Thirdly, there are studies conducted on the topic of threat modeling that do not directly yield a new approach or tool. For example, a case study could analyze the efficacy of an application for a particular problem.

3.1.1 Threat Modeling Methodologies

At the heart of every threat modeling tool is a methodology that defines the conceptual and procedural semantics for modeling. [123] provides an extensive survey of such methodologies. Numerous software implementations of these methodologies have since emerged. Most threat modeling methodologies emphasize the stage of threat discovery.

The most widely adopted methodology, *STRIDE*, has been developed and actively used by Microsoft [18]. *STRIDE* aims to help stakeholders create the threat model, *i.e.*, to derive threats. For that, users take on an adversarial view of system properties. For example, while the system's availability may be a desirable property, in *STRIDE*, stakeholders will consider Denial-of-Service attacks. *STRIDE* does not dictate a strict asset modeling approach. At its core, this methodology provides a mnemonic for participants to be aware of different types of threats. The main contribution of said methodology is a categorization of possible threats. The following threats are considered when applying the *STRIDE* model [171]:

1. **S**poofing involves misusing a user's information used in authentication.
2. **T**ampering generalizes over threats that include malicious changes to user data.
3. **R**epuditation refers to the potential to deny an action.
4. **I**nformation Disclosure refers to sensitive information being leaked.
5. **D**enial-of-Service attacks pose a threat to the availability of a service.
6. **E**levation of a user's system privileges

In summary, *STRIDE* supports threat identification by offering a broad classification of threat types [123, 171]. However, it does not contribute explicit procedures or a list of commonly observed threats. Due to this, other methods commonly derive from it or use it as a complementary methodology [123, 232], including the variants *STRIDE-per-asset* or *STRIDE-per-interaction*. In the former, an asset-centric viewpoint is adopted without necessarily (but not preventing the modeler from) modeling complex relationships and interactions. The latter focuses on modeling potential threat vectors according to triples: an asset, a relationship or operation, and another asset. For example, a user (*i.e.*, an asset) uploads data (*i.e.*, interaction) to an admin panel (*i.e.*, another asset).

Other methodologies aim to identify relevant threats through a clearly enumerated set of concrete threats. *CAPEC* offers public access to an inventory of frequently observed attacks, described as patterns from an adversarial perspective[41]. While these approaches may not define explicit steps that the modeler must follow, they implicitly provide such mechanisms. For example, the knowledge base provided through *CAPEC* can be hierarchically navigated, starting at one conceptual node such as an attack domain (*e.g.*, software, hardware, supply chains).

Providing operational constraints and details, the *Adversarial Tactics, Techniques & Common Knowledge* knowledge base (ATT&CK) is often sourced by threat libraries [42]. Other attack libraries are technology-specific. Projects such as OWASP aid organizations to reduce risks by increasing awareness of a small set of arguably highly relevant threats [200], though they are often used complementary with other attack modeling approaches [232]. Some methodologies merely represent modeled threats to facilitate reasoning about threats [232]. These approaches provide only an implicit methodology and do not explicitly prescribe a series of steps to perform. Instead, these methods offer implicit guidance through the structures they provide. For example, by using *Attack Trees*, broadly formulated attacks are hierarchically broken down into tactics [123] (*cf.*, Section 2.1.6).

Methodologies that extend beyond threat discovery are more complex since they delve into assessing threats or their relation to countermeasures. *PASTA* is a notable methodology that begins with defining business objectives, highlighting that the authors of the methodology emphasize the preservation of business relevance. On a more technical level, the scope (*i.e.*, the set of relevant technologies and components) of the model is established. After decomposing these assets, threats, and their potential vulnerabilities are modeled and subsequently integrated into attack scenarios. The final step then relates the threat model to a broader risk management program [260]. Hence, not all threat modeling approaches are entirely detached from the risk management field. For example, *NIST SP 800-154* follows a similar, technology-specific approach to threat modeling while positioning it as a "*form of risk assessment*" [180].

3.1.2 Threat Modeling Tools

Section 2.1.4 introduced two types of representation-based approaches according to which threat modeling tools can be categorized: diagram-based and text-based modeling tools. This review starts by introducing low-complexity, generic tools that are, nevertheless, commonly applied for threat modeling. Overall, the inclusion of tools is based on their reference in academic literature. Industry tools were included based on a keyword search. Open-source projects that are declared as deprecated were excluded. After introducing the selected tools, they are reviewed against specific problems.

Whiteboards are one of the first choices considered when modeling threats. They are advantageous since they exhibit a great degree of flexibility. They could likely accommodate any visual methodology, providing a rapid start to modeling without acquiring or learning specific tools. However, a substantial drawback is that this flexibility also implies a lack of guidance, potentially giving rise to common pitfalls described for threat modeling, such as the symptom of a "hero threat modeler" [253]. Furthermore, it is hard to digitize, store, recollect, and adapt the resulting models [232].

Other tools not explicitly designed for threat modeling but experiencing strong proliferation are *Office Suites*. These software implementations provide comparable flexibility so that different methodologies can be applied. Such tools can store, distribute, and share the resulting threat models in digital form. Additionally, numerous multimedia formats can be embedded. However, *Office Suites* are usually commercial applications with limited or complex extensibility. Although some templates exist to aid *e.g.*, threat assessment or reporting, these tools may not implement a particular threat modeling methodology [232, 256].

diagrams.net – formerly known as *draw.io* – is a general-purpose diagramming application that has been used successfully to model threats. Since the main goal of the platform is to enable flexible, web-based diagramming, visual threat modeling approaches are most closely conceptually aligned. Although the project's codebase is considered closed-for-modification [134], this modeling application enables the creation of custom element libraries and templates [134]. It is, therefore, possible to use *diagrams.net* as a threat modeling tool by using generic shapes or third-party element libraries [112]. If users are comfortable storing their models on a cloud provider's service, *diagrams.net* can be used in a collaborative setting. However, the lack of interfaces and its highly complex implementation makes for a strongly manual approach. For example, implementing automated reporting features based on a diagram or integrating a threat enumeration library would be a complex procedure [134].

Microsoft Threat Modeling Tool (MTMT) is a popular choice to modeling threats in a software-supported way [170, 232]. In fact, it was one of the earliest tools available for this purpose [248]. MTMT focuses on a diagram-centric view that facilitates explicit communication of threats and coun-

Table 3.1: Overview of Surveyed Threat Modeling Tools [274]

Tool	*OSS*	*Free*	*Engine*	*Input*	*Methodology*
Whiteboard			Manual	Physical	Flexible
Office Suite			Text, Tables	Unstructured Text, Spreadsheets, CSV	Flexible
diagrams.net	✓	✓	diagrams.net	XML, SVG, PNG	Flexible, STRIDE,
IriusRisk			diagrams.net	XML	STRIDE, Flexible
MTMT		✓	Custom	Proprietary (TM7, TB7, CSV)	Microsoft SDL, Flexible
OWASP TD	✓	✓	Custom	JSON	STRIDE, LINDDUN CIA, Flexible
ThreatSpec	✓	✓	GraphViz	Code (Comments)	Custom
pytm	✓	✓	Custom (Code)	Code (Python)	Custom, STRIDE
ThreatModeler			Unknown		Attack Trees, CAPEC
Tutamantic			None	XML Annotations	OWASP, CAPEC, CWE, STRIDE
Threagile	✓	✓	GraphViz	Code (YAML)	STRIDE, Flexible
TRIKE	✓	✓	Tabular	Microsoft Excel	TRIKE
SeaMonster	✓	✓	Eclipse Modeling		Attack Trees
securiCAD			Unknown	Infrastructure,	Modeling & Simulations (STRIDE, MITRE)
MAL	✓	✓	None		Flexible, CAPEC
Raindance	✓	✓	Unknown	Markdown	Attack Maps
SDElements			Tabular	Microsoft Excel, CSV	Custom expert system: OWASP, CWE, etc.
CAIRIS	✓	✓	GraphViz	XML	STRIDE, KAOS, Personas

termeasures between the relevant stakeholders. Such stakeholders must not be security experts. Although MTMT is integrated with Microsoft's *Security Development Lifecycle* (SDL), it can accommodate other methodologies such as STRIDE. The resulting diagrams and derived reports can be shared between team members using a shared storage service. Although the tool is distributed free of charge and extension mechanisms exist, the source code behind MTMT is closed [170].

Another diagram-centric implementation of a threat modeling tool is the commercial application *IriusRisk*. Although it provides free access to a simplified version of the tool, the main components are not published to the open-source community. However, IriusRisk uses *draw.io* as an internal drawing engine, enabling compatibility with existing models [126]. Similarly, *OWASP Threat Dragon* (OTD) provides a simplified modeling environment. Only five generic visual elements are available to model

assets, threats, and mitigations. In that sense, OTD differs from other tools like *IriusRisk*, which provides visual elements for specific technologies. For example, within *IriusRisk*, users are able to use a particular stencil to model a Docker container. In OTD, users would have to label a process element manually. However, this simplicity is a deliberate design choice to ensure a simple User experience. The open-source tool is accessible as a web- or desktop-based application [197].

Differently, *Threatspec* is not built around a drawing engine that allows threats and assets to be modeled freely. Instead, developers annotate their code, generate a threat model, and report using a script. With that, the authors aim to bridge the gap between designing security and software engineering. Since *Threatspec* uses code as a medium, version-control systems can be used for sharing data [255]. *pytm* is similar in that it uses code to model assets. A set of predefined asset abstractions can generate a threat model in the form of a sequence- or data-flow diagram. Since the application focuses on automating threat modeling, it is similar to *Threatspec* and, therefore, unlikely to be used interactively [247].

By relying heavily on automation and threat intelligence, *Threat Modeler* tries to overcome the need for manual collaboration with security experts. Said application provides vendor-specific elements to model processes, assets, and perimeters [254]. Using third-party attack libraries, *Threat Modeler* can automatically generate attack trees to identify specific threats [232]. Integrations with issue-tracking software enable traceability of threats during software development. Extending ThreatModeler requires consideration of its closed-source and non-free nature.

Similarly, *Tutamantic* strives to achieve automation by relying on established threat libraries as a source. An existing model of the target architecture must be annotated with security-related metadata. For example, an architectural model may have been previously created and is then annotated using a descriptive language, which can be done using a *draw.io* plugin [114]. Based on that, *Tutamantic* automatically provides a list of threats which can be exported as a machine-readable report. It appears that this tool does not dictate a specific methodology; extending the tool with a new one may potentially require collaboration with the developers of this commercial tool [259].

A tool that models assets using a descriptive language is *Threagile*. Invoking the toolkit implies that the architecture is evaluated against a set of rules to identify risks. By representing the architectural description using YAML files, the modeling procedure can be integrated using an *Integrated Development Environment* (IDE). Finally, *Threagile* produces reports using different representations such as *Data Flow Diagrams* (DFD), text-based management summaries, as well as machine-readable representations [223]. This toolkit is implemented as an open-source command-line application to be tightly consolidated into the software engineering and operation process (DevOps) [224].

TRIKE refers to both a modeling approach and an implementation thereof. To adopt the methodology, a spreadsheet or desktop application can be used [33]. Since the spreadsheet is the leading distribution of the methodology [232], it is most common to see threats being modeled in tabular representation. Using the conceptual framework implemented by *STRIDE* requires that three models be created. First, a requirements model must be created depicting relevant actors, assets, actions, and the rules surrounding these actions. Once those are elicited, a second model is created to contrast requirements against their implementation. Here, one finds elements known from other methodologies, such as DFDs and use flows. Once these models are derived, the actual threat modeling is carried out. Again, this process is multi-phased with commonly known elements, including vulnerability analysis, attack graph, and threat generation [33].

By providing threat modeling using three viewpoints, *SeaMonster* aims to bridge the knowledge gap between security professionals and software engineers. This desktop application lets participants model threats using attack trees and use-case diagrams [204]. At the time of writing, *SeaMonster* is not actively developed but is still available as an open-source application [232].

A toolkit that automatically creates a threat model based on specific architectural models of multiple cloud platform vendors such as *AWS* and *Azure* is *securiCAD*. The *enterprise* distribution of *securiCAD* provides a web-based implementation where multiple users can collaboratively perform threat simulations. Additionally, experts can manually model assets using the *Professional* distribution. Although users are not required to model threats manually following a specific methodology, simulation reports are still mapped to known methodologies. Specifically, *securiCAD* leverages taxonomies from *STRIDE* and *MITRE* [86]. Closely related to *securiCAD* is the meta-attack language platform *MAL*. With the *MAL* syntax, modelers can create a language specification to describe the assets and access techniques to these assets. By capturing benign and malicious access techniques, threats are modeled. Once the language specification is compiled using the *MAL* compiler, one can programmatically derive the likelihood of threats. This compilation enables integration with other tools, such as the previously described *securiCAD* tool, to provide statistics and graphical representations [87].

Raindance propose a novel methodology called *attack maps*. The main idea is to derive attacks for external, internal, and privileged attack scenarios. *Attack maps* are considered a supplementary methodology that can be consolidated into the development process by incorporating markdown files into the projects' version control system. Although *Raindance* provides templates to link attack files to the project, it remains unclear how users derive threats and how the evaluation during continuous delivery can be achieved. It is unclear whether the tool is actively maintained [68].

Manual modeling of assets can be avoided by applying *SD Elements* to model threats. Here, assets are derived using an automated survey or by analyzing existing sources. Then, threats are automati-

cally derived using an expert system, and appropriate mitigations can be tracked using existing issue-tracking software like GitHub, Jira, or Azure DevOps [227]. Various families of methodologies, such as OWASP, NIST, and CWE, influence the output of the expert system [228].

CAIRIS was developed as a security requirements management tool. It allows threats to be modeled using different methodologies such as STRIDE and KAOS. To do so, the tool supports the generation of attack trees and DFDs [39]. By porting the application to a web-based software-as-a-service distribution, the authors made modeling possible outside on-site settings [76]. In addition, *CAIRIS* provides interfaces to integrate it into other collaborative platforms such as Trello [39].

Evidently, several tools that differ in complexity, approach, design accessibility, input data, and methodology are available, as highlighted in Table 3.1. Although some have a specific technological focus, the applicability of these tools is arguably rather broad. With this in mind, the following sections each review a relevant subset of these solutions and dedicated academic contributions against specific problems.

3.1.3 Collaboration

As highlighted by Figure 2.16, problems are conceptualized from a specific problem instance. This first review of the threat modeling solution space considers the following problem instance, in which the thesis author participated. Due to the political relevance of postal and electronic voting systems, a threat model was collaboratively developed, considering the design of the voting process adopted within one canton and its municipalities in Switzerland. Thus, the author was involved as a practitioner in the problem space. To do so, suitable tooling was reviewed to conduct the threat modeling project, yielding two problems due to the constraints of the project.

First, the project had to be carried out according to remote working environments imposed by regulations motivated by the COVID-19 pandemic. Popular elicitation techniques found in risk management, such as interviews or workshops [120], were not feasible. As an effect, the usage of tools was impacted, too. For example, the applicability of the previously mentioned whiteboard, acting as a generic and highly flexible tool, had to be re-evaluated. Secondly, the threat model had to consider the delicate situation where the implementation of artifacts was not available. However, white box descriptions of systems were accessible. The voting process involves many stakeholders and processes, while fragmented across systems and authorities. Hence, the security of the voting process from a design perspective had to consider non-technical aspects of the overall information system, which had to be elicited from non-technical stakeholders, too. In summary, the two key problems for creating the threat model are *(i)* conducting a threat model while including remote and asynchronous participation and *(ii)* including diverse artifacts from technical and non-technical sources and stake-

holders. The solution space was questioned regarding the applicability of threat modeling tools and methods, considering remote, asynchronous, and cross-functional collaborative settings.

The previously identified tools were analyzed based on the literature and dynamic black-box analysis to discuss the applicability of existing threat modeling tools. The latter is necessary since industry tools cannot be ignored due to their existing user base, contributing to a relevant area of the solution space. However, since they are commercial, closed-source applications, only functional testing is feasible to understand if they support, for example, remote threat modeling in a synchronous manner – especially if no documentation or scientific literature is discussing this functionality. For this survey, existing threat modeling tools were categorized based on their support for synchronicity requirements (*i.e.*, whether users can model simultaneously) and remote participation options. This survey examined the collaborative features of 18 tools, categorizing them into one of five proposed categories, which follow a similar approach, as summarized in Table 3.2

Under *General Purpose-Tools*, the author subsumes widely used applications that do not adhere to a specific methodological approach but are flexible enough to support threat modeling. *Whiteboards* are popular for quick modeling but lack support for asynchronous and remote collaboration [274]. For example, persisting a threat model for an asynchronous collaborator is not trivial. Although modelers sometimes use webcams and online meetings to include remote participants, this method is arguably suboptimal. Similarly, office suites allow for asynchronous sharing of threat models. *diagrams.net* is an online diagramming tool for which third-party threat modeling stencils are available. In that sense, this is the only platform that partially provides guidance, albeit only from a structural perspective (*e.g.*, DFD stencils, asset tables). For this set of tools, limited support for remote collaboration is present, and these platforms do not offer guidance for untrained or asynchronously participating users. However, the flexibility of these tools enables the modeling across almost any asset representation (*e.g.*, users can draw a system as process, architecture, or source code by using a whiteboard) [275].

Microsoft Threat Modeling Tool, *OWASP Threat Dragon*, *TRIKE*, *SeaMonster*, and *CAIRIS* are considered *manual modeling applications* featuring a diagram drawing component. These tools follow a specific threat modeling methodology, making them suitable for settings where this methodology applies. Many of these tools are web-based, allowing asynchronous and remote use to some extent. However, creating models synchronously is not feasible according to the tests performed and descriptions in related surveys [23]. In addition, their tight coupling to a methodology can be perceived as a limiting factor, coupling their efficacy in collaborative settings to the methodology [232, 275]. For example, if a tool follows a technology-oriented approach, stakeholders must be aware of this. In addition, this may exclude certain artifacts or representations.

Recently, two new threat modeling trends have emerged. *Automated Threat Modeling* approaches leverage existing artifacts (*e.g.*, source code or architecture diagrams) to automate threat and countermeasure discovery. Tools such as *IriusRisk*, *Tutamantic*, *securiCAD*, and *MAL* do not support synchronous modeling. For example, the securiCAD toolkit can be used in a collaborative setting. Remote users are able to view the model without any restrictions. However, modifying a model requires the user to acquire a lock. Hence, only one user can perform mutations. For asynchronous collaboration, *securiCAD* provides a convenient way to merge diverged models [86]. Similarly, in *IriusRisk*, collaboration is achieved in the threat mitigation step through its integration with other applications that allow vulnerabilities to be found and managed in an automated manner. However, during the modeling stage, it is not possible to directly collaborate with remote users [126].

Overall, the dependence of automated tools on a specific methodology or input and the usage of a non-collaborative editor can make certain collaborative activities, like workshops, impractical. However, these tools' automated threat discovery indicates the presence of a knowledge base, allowing inexperienced users to identify threats without being security experts. Still, the dependence on an existing input limits threat models to methodologies where the artifact is applicable [275]. Similar features are found in the modeling approach used by tools like *ThreatSpec*, *Threagile*, and *Raindance*, which aim to *integrate* and connect threat models to the software development lifecycle using sources such as markdown files or source code [275]. Finally, *pytm*, *ThreatModeler*, and *SDElements* combine aspects of the described approaches into a *hybrid* methodology. These tools provide similar support for collaborative settings [232, 275].

In summary, the following three limitations are derived from analyzing the literature and artifacts concerning their collaborative capabilities. First, there is a lack of studies *(i)* explicitly investigating the role of collaboration in threat modeling. To the best of the author's knowledge, [168] is the only paper explicitly examining the role of collaboration in threat modeling. However, this work focuses on the asset identification stage. Nevertheless, the authors emphasize the role of collaboration in threat modeling, which is congruent with the voice of practitioners that state "*We have come to value: (..) People and collaboration over processes, methodologies, and tools [253].*" Secondly, the results of this review question *(ii)* both fully automated and highly manual approaches. The former likely requires as input existing artifacts that already impose limitations (*e.g.*, type of system being modeled, stage of the project), while the latter can accommodate any threat modeling methodology at the loss of guidance. Finally, the investigation of collaborative features from a technical perspective highlights *(iii)* a lack of tools that support synchronous remote threat modeling and a lack of methods leveraging asynchronous working styles. For example, the previously introduced biases from ERM could be

Table 3.2: Comparison of Toolkit Support for Collaboration [275]

Threat Modeling Approach	Async & On-site	Async & Remote	Sync & On-site	Sync & Remote
General-Purpose Tools (3)				
Whiteboards	✗	✗	✓	✗
Office Suites	✗	✓	✓	✗
diagrams.net	✗	✓	✓	✓
Manual Threat Modeling (5)				
MTMT, TRIKE, OWASP Threat Dragon, SeaMonster, CAIRIS	✗	✓	✓	✗
Automated Threat Modeling (4)				
IriusRisk, securiCAD	✗	✓	✓	✗
Tutamantic	✗	✓	✗	✗
MAL	✗	✗	✗	✗
Integrated Threat Modeling (3)				
ThreatSpec, Threagile, raindance	✗	✓	✗	✗
Hybrid Modeling Approaches (4)				
pytm, SDElements	✗	✓	✗	✗
ThreatModeler	?	✓	?	?
CoReTM	✓	✓	✓	✓

✓ = satisfies property ✗ = limited or no support ? = support unclear

tackled by leveraging asynchronous collaboration: to counter "group-think", participants could be asked to use separate areas of a tool to discuss the relevance of a threat, which could later be merged by the tool.

3.1.4 AI-based Systems

In the previous review of threat modeling tools and methods, generic approaches (*i.e.*, potentially applicable to any technology or system) were at the focal point due to the research questions posed. The problem synthesized from the previous problem instance was considered a cross-cutting concern. In this second problem instance, a more technology- and system-oriented problem is consid-

ered. Here, the author was involved in a security effort regarding the design of an AI platform for the health care sector. *AISym4MED* – the umbrella of the project – aims to establish a novel platform for engineers, practitioners, and researchers working with healthcare data. The platform is designed to provide a dependable dataset system augmentation with controlled data synthesis, promoting experimentation and modeling within the healthcare sector. Additionally, it strives to enhance data privacy and security by incorporating advanced privacy-enhancing mechanisms, including anonymization or federated learning [293]. The primary functionalities of the envisioned architecture are summarized through three main modules providing the ML pipeline (*cf.*, Figure 2.20): Model Training, Model Auditing, and Data Synthesis. These are supported by seven underlying modules: Data Anonymization Toolkit, Data Publishing, Cross-Border Database, Data Explorer, Dataset Builder, Dataset Evaluator, and Federated Learning [19].

Throughout the project, no method or tool dedicated to particular architectural assets of AI-based systems was employed to guide and automate the creation of threat models. Hence, the modelers manually developed abstractions of the envisioned system, modeled it using the drawing tool *diagrams.net* (*cf.*, Section 3.1.2), reasoned on their relevance, prioritized threats, and discovered countermeasures. Due to the apparent novelty of integrating AI technologies into a system architecture, modelers resorted to manually reviewing security-relevant information about these systems. Based on these experiences, additional limitations for the threat modeling domain are described. Specifically, this review analyzes whether threat modeling solutions guide and automate the secure design of AI-based systems.

To assess this, a similar methodology was followed – comprising literature analysis (*i.e.*, academic manuscripts and software documentation) and dynamic black box analysis. The findings on threat modeling of AI-based systems are described through three areas: *(i)* descriptive research on AI threats, *(ii)* the previously introduced and widely available threat modeling tools, and *(iii)* academic literature particularly focusing on the intersection of threat modeling and AI [280].

The first area appears to receive wide attention [280]. For example, a recent survey categorizes cyber attacks on AI systems along the previously introduced ML pipeline. Data poisoning attacks compromise the resulting model by injecting adversarial samples during data collection – which can be done at the source or at rest [162]. The authors describe various attack objectives and multiple poisoning strategies (*e.g.*, randomized or targeted) [177]. Moving to the feature selection and model training stages, model poisoning strategies replace the model with a tampered one [220]. Threats focusing on model inversion, inference, and failure are described during the production stages of the pipeline. Model inversion recovers training information [220], while extraction threats aim to recreate and approximate the model without direct access to it [220].

Table 3.3: Studies Applying Threat Modeling to AI Systems [280]

Work	*Year*	*Contribution*	*Evaluation*	*Domain*
[288]	2020	Method, Survey	Illustration	Requirements Engineering
[159]	2021	Degradation Method	Demonstration	Adversarial ML
[106]	2022	Threat Model	Demonstration	Cellular Networks
[160]	2022	Methodology	Illustration	AI Threat Modeling
[280]	2024	Methodology, Tool, Threat Quantification	Case Study, Field Test	AI Threat Modeling

Considering the previously introduced threat modeling tools and methods, none of the ones commonly deployed in the industry, such as the *Microsoft Threat Modeling Tool*, *CAIRIS*, *Threatspec*, *SDElements*, or *Tutamen*, are specifically designed for AI systems [39, 228, 255, 259, 280]. Again, the interplay between guidance and flexibility is noticeable: particular tools are flexible and allow users to develop custom threat libraries. Still, no taxonomies or methodologies specific to the AI paradigm are integrated. In addition, it must be considered that many are non-free or closed-source (*cf.,* Table 3.1), limiting the degree of extensibility. In addition, it is not clear how approachable security tooling is for non-security experts, especially if it must be customized first. In a small-scale interview conducted when testing the *ThreatFinder.AI* tool (*cf.,* Section 4.4), none of the participants with a data science background were aware of common threat modeling tools. Thus, it could be restated that there exists a trade-off between flexibility and guidance. For instance, *diagrams.net*[134] had been known to these data scientists – albeit not for its ability to model security aspects.

Although the industry developing those threat modeling tools may focus specifically on AI systems, other industry-related activities highlight efforts to enumerate attacks (*i.e.*, develop threat libraries) [280]. *MITRE* has extended its well-known catalogue of attacks through *MITRE ATLAS* which presents attack tactics specific to AI [251]. Microsoft has provided guidelines on how to mitigate AI-related threats [169]. Similarly, *OWASP* released a similar guide detailing threats and controls across three stages (*cf.,* Table 2.3) [198]. Similarly, the *European Union Agency for Network and Information System* (ENISA) provided an asset-based meta-model and a comprehensive report on AI-related threats [75]. While these efforts are likely fruitful for the industry as a whole, they do not fit the classification of a threat modeling tool or method.

The third area closely aligns with the research question – targeting literature integrating the AI paradigm within threat modeling. Only a handful of publications [160] (see Table 3.3) investigate how to connect potential threats to the elements found across the phases of the ML lifecycle [280].

[288] propose a methodology to apply conventional threat modeling based on DFDs and STRIDE. While the methodology successfully maps a threat ontology to an illustrative case, this assignment is manual, requiring a security expert's substantial involvement. Furthermore, this study is limited since it presents only a single illustrative case study based on synthetic data and no investigation into usability [288]. [159] used a gold standard dataset to test model degradation during inference. Hence, the paper yields a metric quantifying degradation loss, which could potentially be used to approximate its impact. However, the description of the method implies that this work focuses on trained and available models (*i.e.*, does not apply to at design time). For secure system design, however, the design stage is highly relevant, too, questioning the applicability of this method for threat modeling. Other authors focus on specific domains. A threat model focusing on Open Radio Access Network (O-RAN) architectures was developed by [106]. In that sense, the authors propose a particular threat model for one domain – instead of a generic approach. Finally, the work most closely aligned is [160]. The authors propose the integration of threat modeling methodologies for AI security analysis. They introduce *STRIDE-AI*, adapting the STRIDE mnemonic for ML systems. This approach assigns ML-specific translations to security properties, supporting AI threat identification. However, this mapping process is done manually, limiting automation and scalability. The evidence of its efficacy is comparable to the previous studies: it is based on a single use case description without incorporating actual users [280].

Thus, this review highlights that a set of threat libraries has been established in this context. When surveying the literature, these compilations comprise many of the attacks described by the literature; however, novel trends within AI would need extension or consideration of novel libraries. For example, [198] describe generic attacks, which, on a high level of abstraction, relate to new trends such as LLMs and FL. However, when analyzing the integration of this knowledge into practical threat modeling tools and methods, a lack of automated and guided approaches is apparent – even though related research demonstrated that threat modeling, as a discipline, can be applied to particular AI systems. In addition, it can be observed that the existing AI threat modeling studies mainly focus on technical representations of model degradation or threat identification. Assessing and quantifying AI threats appears to receive little attention [269, 280].

3.1.5 Insider Threats

For the third review in the field of threat modeling, additional questions from the problem instance defined in Section 3.1.3 are leveraged. Two observations form the basis of this review: the usage of process diagrams by the domain expert and the prevalence of insider attacks in this problem instance (*i.e.*, postal voting). Focusing on the latter, insider attacks are highly relevant in the voting process. For

example, in 2021, an election official accessed physical voting envelopes before they were subject to tallying [241]. Although this does not directly correspond to a pure attack on a computerized system, it does affect the information system's security as a whole. As such, it is argued that information must be analyzed holistically, focusing on media breaks and the flow of information throughout the system. For this, the process models developed for documentation could have served as an abstraction to model such threats. For example, electoral data is first stored digitally through the electoral register, then transformed into a physical representation by printing voting envelopes, and later in the process, aggregated digitally. For the following survey, it is hypothesized that process models could be used to model threats on a process encompassing different systems. More specifically, the usage of BPMN is investigated as a de facto standard [53]. Insider threats are brought into focus as an exemplary attack. Combining these two aspects, this review questions the tools' support when modeling insider attacks and the usage of process models to do so.

Again, related studies are summarized using three areas, and an overview is presented in Table 3.4. Firstly, several papers explore insider threats from a psychological standpoint. These papers aim to explain the underlying organizational factors and their impacts. As such, they explain contributing factors and do not necessarily focus on problem-solving research – although a small minority of studies contribute in the form of prescriptive explanations or academic programs. For instance, [174] uses game theory as a theoretical model, leading to a simulation approach that explains the organizational dynamics between adversary and defense actor. It is concluded that these studies are crucial for building knowledge bases, but they do not directly address the research questions since they do not contribute to threat modeling tools or methods.

The second area includes studies that are closer to the subject, since they focus on threat modeling or risk-based methods and specifically consider insider threats. For example, [32, 110, 189] employ various methods to achieve threat modeling or risk analysis. This includes the development of attack graphs or strategic plans to integrate insider threats into the risk management of an information system. However, none of these solutions and studies offer an automated threat modeling approach. Furthermore, many do not focus on process models as an input abstraction.

Thirdly, a few solutions address insider threat modeling or utilize BPMN for cyber attack management. For example, [294] contributed a methodology and a prototype. Their approach focuses on compliance verification against existing security requirements. It can be argued that BPMN is a valuable abstraction for model-based security management. However, the scope of this study did not specifically focus on insider threats and required that security requirements engineering had already been performed. Another related study is [99]. The authors used BPMN's extensibility feature –

Table 3.4: Studies Investigating Insider Threat Modeling or BPMN [277]

Sol.	*Year*	*View*	*Aspect*	*Methodology*	*Prototype*	*Insider*
[142]	2021	PSY	ORG	adversarial risk analysis	no	yes
[174]	2016	PSY	ORG	agent-based model, game theory, system dynamics	no	yes
[15]	2017	PSY	IS	SecureInT, Look-ahead analysis	no	yes
[296]	2023	PSY	ORG	Fuzzy Set Theory, Bayesian networks	Academic Program	yes
[144]	2017	SOC	ORG	3-step explanation	Isabelle/HOL	yes
[110]	2018	RA	IS	Several	-	yes
[189]	2014	RA	IS	NIST	ADVISE	yes
[32]	2008	RA	Process	TRIP	-	yes
[52]	2005	TM	IS	Challenge Graph	-	yes
[26]	2014	TM	Process	Fault Tree Analysis	Automated FTA	yes
[294]	2020	TM	Process	SQUARE	-	no
[217]	2017	TM	Process	SecBPMN	Query Engine	no
[99]	2023	TM	Process	ENISA, OWASP	BPMN annotator	no
[107]	2021	TM	Process	NIST-based	coreLang	no
[277]	2024	TM	Process	Knowledge Map	BPMN modeler	yes

ORG=Organizational, PSY=Psychological, SOC=Sociological,
RA=Risk Assessment, TM=Threat Modeling, IS=Information System

introducing annotations to model threats by mining knowledge from security frameworks. Their approach provides further evidence of BPMN's utility through a case study. Yet, since they introduced new annotations, it is unclear whether existing process models can be used without substantial effort. In addition, the authors did not revolve around insider threats.

Other approaches focus on insider attacks. [52] model them using graphs. Therefore, this study does not leverage process models. The most relevant study for this review is [26]. In this study, the authors analyze processes to identify vulnerabilities. Unfortunately, only two threats are included: data exfiltration and sabotage. Moreover, each threat is analyzed using different methods, complicating the approach and lacking a comprehensive view of all possible insider attacks.

Additional limitations are synthesized from these findings: None of the discovered approaches effectively analyze business process models for insider threat identification. Nevertheless, the research shows that process models, and specifically BPMN, are a valuable abstraction for security discussions. Notably, [99] demonstrate that annotations can enable BPMN for threat analysis. Thus, insider threat modeling using BPMN emerges as both a limitation and an opportunity.

3.1.6 Threat Quantification

This review analyzes the numerical scales (*cf.,* Section 2.1.5) employed and the steps followed when assessing threats. As will be illustrated, quantitative approaches – especially ones oriented towards business value and ones using an interval scaling – are rare. Hence, related research is depicted from two perspectives: introducing threat assessment methods and an illustrative set of quantitative approaches to risk management.

Table 3.5: Threat Assessment Tools and Methods [273]

	Sol.	***Threat Assessment***	***Importance Metric***
Methods	*STRIDE*	None	None
	LINDDUN	None	None
	DREAD	DREAD	Score *(I)*
	CVSS	CVSS	Score *(I)*
	PASTA	PASTA	Score *(O)*
	PbD	DREAD-based	Score *(R)*
	Attack Centric	Custom	Score *(R)*
	Security Cards	None	None
Tools	*OWASP TD*	Severity Estimation	Categories *(O)*
	CAIRIS	Risk-based	Categories *(O)*
	TRIKE	Exposure Estimation	Score *(I)*
	MTMT	Priority Estimation	Label *(O)*
	ThreatModeler	Severity Estimation	Label *(O)*
	SeaMonster	None	None
	IriusRisk	No Response	Categories *(O)*
	SDElements	——— No Vendor Response ———	
	securiCAD	——— No Vendor Response ———	
	Tutamantic	Actuarial models, Markov-chain	Categories (O) Numerical Score
	Threagile	Risk-based	Labels *(N)*
	ThreatSpec	None	Manual Labels
	QuantTM	BIA	Business Loss *(R)*
	ThreatFinder.AI	BIA for AI	Loss Simulations *(R)*

O = Ordinal, N = Nominal, I = Interval, R = Ratio

Table 3.5 highlights related tools and methods. Based on the literature, dynamic analysis, and direct vendor inquiry, it is summarized as to whether and how threat assessment is achieved. In addition, the importance metric is elicited and assigned to a scale.

A subset of tools and methods completely exclude an explicit assessment step and focus on threat identification without defining a compatible assessment approach. For example, the popular *STRIDE* method does not define how threats summarized by its mnemonic could be assessed [273]. Similarly, only a few methods focus exclusively on threat assessment. However, *DREAD* is still a commonly employed method for assessment. Modelers explicitly but qualitatively define values for several predefined factors that should be subjectively assessed. For example, the reproducibility of an attack must be assessed on a nominal scale with four values $S = \{0, 5, 7.5, 10\}$ which correspond to the assessments of $D = \{$*Difficult or Impossible*, *Complex*, *Easy*, *Very Easy*$\}$. All values are summed up and qualitatively interpreted using a nominally scaled value (*i.e.*, $\{$*low*, *medium*, *high*, *critical*$\}$) [71]. From a risk-based perspective, one could argue that this approach lacks several functions: Interpreting the final outcome is subjective and unrelated to a specific business value. Furthermore, this score may not be suitable to express uncertainty regarding the modelers perception of a threat (*i.e.*, it is not feasible to express that a threat could yield "no damage" or "information disclosure") [273].

Similar observations are made for threat modeling tools, frequently incorporating non-business-oriented methods or producing qualitatively scaled assessments. For example, *Tutamantic* leverages statistical analysis of actuarial models and Markov-chain analysis. While the approach does produce an interim, quantitative value, risk is deduced from the sum and complexity of inbound data flows per system component. Thus, the approach quantifies technical details instead of the threat's business relevance. In addition, while interim results are available as a quantitative score, the final result is assigned to a set of categories. In addition, while interim results are available as a quantitative score, the final result is assigned to a set of categories, making the output qualitative. Other tools, such as *OWASP Threat Dragon*, do not guide the threat assessment stage and only store a subjectively assigned qualitative score. Other tools altogether avoid threat assessment [273].

To summarize the threat modeling field concerning threat assessment, there is no widely accepted metric or methodology for assessing the importance of threats. Approaches either omit threat prioritization or use qualitative scoring systems. Additionally, there is an apparent lack of business-oriented solutions that could potentially convey the business impact of threats and foster collaboration between technical and business stakeholders. As such, no approaches in this field exploit methods from BCM, such as BIA. Most threat modeling methods do not translate the technical results, making data collection and prioritization interpretation challenging for non-technical collaborators. These findings can be contrasted with studies of quantitative risk management. Value-at-Risk (VaR) is common in finance but less adopted in cybersecurity. [73, 90] introduce a cyber value-at-risk model, demonstrating its applicability through a case study in insurance. However, the model depends on parameters, such as defense probabilities, which may not align with threat modeling in the design

phase, where controls are not yet modeled, and only structural vulnerabilities are assessed. Similar limitations arise in [202], which relies on vulnerability degrees that may not be sensible to model during system design. In contrast to the quantitative cyber risk management field, [16] critiques risk quantification, arguing for simpler, qualitative tools like questionnaires to estimate threat likelihood while emphasizing the inclusion of diverse stakeholders and the bias towards common threats. Other works, such as [201] and [91], explore rule-based analysis and SME security assessments, incorporating expert opinions and historical data, though their broader adoption remains unproven. In summary, quantitative methods are more widely explored in risk management, but their adoption and applicability to the threat modeling domain remain unclear.

3.2 Threat Detection and Mitigation

Actions taken to reduce risks – whether those are preventive, detective, or responsive – cannot be considered in a siloed manner. For example, a detection strategy developed for a specific threat must consider the usefulness of the produced signal for a defense model (*e.g.*, what does it tell the defender about the attack? At which stage of the cyber kill chain is the signal produced?). Likewise, a responsive approach is likely coupled to a detective measure. For example, the effectiveness and efficiency of reactive MTD approaches depend on the detection system's accuracy: many false positives will waste unnecessary resources. In contrast, false negatives decrease the resilience of the system.

This chapter reviews approaches for attack mitigation (*i.e.*, supporting preventive, detective, or responsive actions). As a framing problem instance, the previously introduced *ElectroSense* scenario represents the point of view from which the limitations of related approaches are summarized. Herein, the review is framed by the relevance of malware-based threats, the limitation in terms of resources (*i.e.*, technical and operational) on the defensive side, and the overarching question of leveraging data-driven approaches for both proactive and responsive mitigation. The following sections directly summarize limitations within that frame.

3.2.1 Malware Data Collection

Any data-driven approach requires the collection or ingestion of data. For the threat and defense model considered in this scenario, data about a malware artifact must be collected. For this, analysts use both dynamic and static analysis techniques. In static analysis, findings are drawn from examining the artifact's source code, script, or binary. The resulting data describes static features, for example, the number of libraries linked in a binary (*cf.*, Section 2.2.2). Recent years have seen the rise of ML-based techniques to understand malware *behavior*. These approaches could potentially outperform

static approaches since those are bypassed through obfuscation or polymorphism. Expectedly, gathering realistic behavioral data requires the execution of malware artifacts to enable observation of its behavior. Dynamic analysis, which involves running an artifact to observe its real-time behavior, presents an approach for behavioral data extraction [18].

Dynamic analysis, by itself, is not a new method. It is applied and established across different subfields of computer science (*e.g.*, software testing, performance analysis). However, these approaches arguably execute *benign* software. The execution of *malicious* software presents challenges. For example, the infrastructure where an artifact is executed must be secured. In addition, data must be extracted that yields meaningful behavioral patterns without necessarily knowing the internals of the artifact. Thus, dynamic malware analysis is non-trivial. Since behavioral data may inherently be damaging, sandboxes can represent an execution and data collection environment for malware artifacts that align with threat and defense models.

Several solutions applicable to dynamic malware analysis are outlined in Table 3.6. The review of those solutions examines the supported execution platform and isolation mechanism. These factors are relevant since the underlying platform could affect the patterns of that data. For example, the behavior of a sample – expressed through its relative computational performance consumption within a device – may differ across devices with varying resources. From a technical perspective, the review considers whether platforms are actively maintained, allow for real-time dynamic analysis, and support continuous information collection. This ties into the subsequent stages of the ML pipeline (*cf.*, Figure 2.20), where analysts have to discover features that could represent valuable features. Finally, the review considers whether platforms enable repeatable and reproducible data extraction. Reproducibility – which is already a widely acknowledged goal in the research community – may be vital for reactive defense since behavioral models would have to be retrained using data collected across platforms (*e.g.*, executing a malware sample for different device configurations) or due to temporal changes (*e.g.*, new or modified samples are discovered).

Tools like *TTAnalyze*, *CWSandbox*, and *Ether* illustrate that many sandbox environments are focused on the Windows OS, limiting their ability to execute and monitor artifacts targeting other systems. This limitation is also observed in proprietary, cloud-based services such as *ANY.RUN*. This category of solutions typically uses *Virtual Machines* (VMs) to isolate dynamic execution data from the host infrastructure, indicating VMs' proliferated role in these settings [81].

Several tools have been developed to analyze malware artifacts targeting Linux-based platforms, including *DRAKVUF*, *Cuckoo*, and *LiSa*. These tools use virtualization technologies such as VMs and emulators. However, they do not specifically focus on behavioral data collection. Additionally,

Table 3.6: Dynamic Malware Analysis Solutions [81]

Sol.	*Year*	*Platform*	*Isolation*	*Maintained*	*Real-time*	*Reproducibility*
TTAnalyze	2006	W	Emulation	Unknown	No	No
CWSandbox	2007	W	VM	No	No	No
Ether	2008	W	VM	No	No	No
DRAKVUF	2021	W, L	VM	Yes	Yes	Only Collection
Cuckoo	2019	W, L, M	VM	No	No	Python-based
ANY.RUN	2023	W	VM	Yes	Yes	Only Basic Presets
LiSa	2019	L	QEMU	No	No	via RESTful API
Tamer	2022	L/ARM	QEMU	Yes	No	Autom. Interaction
V-Sandbox	2020	L	QEMU	No	No	ELF configurator, C&C simulator

W=Windows, M=macOS, L=Linux

solutions like *V-Sandbox* and *Tamer* extend these concepts to the IoT domain, showing support for specialized platforms [81].

Overall, multiple tools exist for various platforms and scenarios; however, many prototypes lack ongoing maintenance. Meanwhile, commercial solutions are often closed-source, limiting their extensibility. Real-time interactivity and data visualization are typically limited, which is a drawback since ML-based methods benefit from extensive exploration of artifacts and features for feature selection and pre-processing. Moreover, reproducibility is frequently overlooked and not explicitly described or supported using dedicated functionality. Several solutions offer interactive use through a user interface, making it difficult to ensure consistent execution conditions during data generation. For instance, repeating an experiment on a different Linux distribution or hardware configuration might lack accuracy. Tools like *Cuckoo* and *LiSa* offer features that enable reproducible malware interaction through their interfaces, though they do not directly guarantee reproducible experiments [81].

To summarize, dynamic malware analysis faces several constraints when contrasted to the scenario adopted herein. First, several platforms are abandoned and unmaintained. Furthermore, real-time analysis capabilities, which aid in exploring data and artifacts, remain limited. Thirdly, collaborative functions (*e.g.*, transferring artifacts among practitioners or platforms) are essential for designing security analysis platforms that address the limitations of current security management practices [274]. For instance, refining a security policy or control depends on data from the incident response teams that first discovered an artifact. The long-term dynamic data collection could benefit from reproducing the same procedure that the incident response analysts applied. Finally, the challenge of data reproducibility is critical. In today's dynamic threat landscape, ML models are retrained fre-

quently using new information, necessitating that experiments can be transferred across platform configurations (*e.g.*, hardware resources, services, OS distributions) with reproducible outputs. As previously illustrated, even for one malware family (*i.e.*, the Mirai strain), more than a dozen samples are published in [3]. The dependency on highly provisioned VMs and manual input reveals the need for lightweight, automated analysis platforms. Thus, there is an opportunity to develop and test lightweight approaches that foster reproducibility while mimicking execution environments particular to the targeted scenario. Assuming appropriate guardrails are employed, container isolation could be used to collect various data sources based on dynamic execution. As will be reviewed for reactive defense, such data could involve network traffic captures, resource consumption fingerprints, or the interaction between user space and kernel space through system calls.

3.2.2 Reactive Malware Defense and Offense

Reactive MTD and deception approaches should answer four questions. [181] prescribes that they must define *what* to move, *how* to apply such a transformation, and *when* to carry it out. However, a survey by [55] also highlights that most studies define a single MTD technique, which is unlikely to cover a threat model completely. From a holistic defense perspective, the defense within the previously mentioned scenario would require multiple parallel defense techniques. As such, it is postulated that a defense system should also be able to decide *which* technique to deploy. In the following, related studies are reviewed concerning *(i)* which reactive techniques they propose against several malware threats, *(ii)* how they orchestrate MTD techniques, and *(iii)* if it is possible to develop MTD techniques that can not just react but avoid the damage function of malware (more specifically, ransomware) after infection.

Threat-oriented Techniques

The survey of [55] highlights the integration between intrusion detection of particular threats and their MTD-based mitigation as a limitation of the field. The MTD field is reviewed concerning how related approaches could mitigate the multi-purpose malware threat model composed of malicious behaviors, considering botnets, backdoors, ransomware, and rootkits [267].

A recent review on IoT MTD techniques categorizes moving parameters, assigning them to categories such as *Network*, *Data*, *Software*, *Runtime Environment*, and *Platform*. The authors also categorize 32 current IoT MTD studies according to the WHAT principle, revealing that 54% focus on network, 20% on runtime, 13% on software, 10% on data, and only 3% on the platform. Starting with network-based MTDs, the most common, [295] introduces *μMT6D*, which reduces recon-

Table 3.7: Constrained MTD Techniques Targeting Malware [267]

Sol.	*Year*	*Attack*	*What*	*How*	*When*	*Eval*
[295]	2018	C&C	N	IP Shuffle	Pr	Sim.
[187]	2019	C&C	N	IPv6 Shuffle	Pr	Sim.
[105]	2015	Code Reuse	R	Code Random.	Pr	HW
[282]	2020	Side-channel	D	Encryption Key	Pr	-
[208]	2016	Side-channel	D	App. Encr. Key	Pr	Sim.

Network (N), Runtime (R), Data (D), Proactive (Pr), Reactive (Re).

naissance attack opportunities by frequently rotating IP addresses. Another network-focused MTD approach, [158], uses MAC address manipulation to maintain communication in the presence of source-selective jamming. [187] presents AShA, a rapid, secure, and collision-free IPv6 address renewal method. Both of these previous methods were tested in simulated settings. By contrast, the techniques developed in this thesis address both network-specific attacks and those targeting data and runtime environments. In the runtime category, [105] introduces a combination of software and hardware techniques that randomize executable code layouts to counter code-reuse attacks and isolate binary code to limit exposure to attackers. Thus, it does not protect against attacks that exploit library linking employed by certain rootkits. Addressing the platform or operating system of constrained devices, [1] introduce *µArmor*, which leverages executable space protection, address space layout randomization, and stack canaries to counter memory corruption attacks [267].

In the data category, [282] offers an MTD mechanism designed to be resilient against electromagnetic side-channel attacks, using regular rekeying to minimize the computational load. A similar data-driven approach, [208], proposes the *Dynamic Key-Length-Based Security Framework* (DLSeF), where frequently updated shared keys ensure end-to-end security. However, neither of these solutions addresses attacks that compromise the system through local data encryption. Table 3.7 compares the main aspects covered by these studies. It is evident that recent malware, capable of encrypting and exfiltrating sensitive data, altering libraries, and concealing their samples and behavior, remains largely unaddressed by current MTD solutions. Additionally, most MTDs are evaluated in simulated environments, and developing comprehensive frameworks for reactive MTD deployment against multi-purpose malware is an opportunity for the field.

Technique Selection

Assuming that several defense techniques are available, this review of reactive defense methods inquires if and how related studies consider orchestrating multiple defense methods. Overall, the field

of MTD, especially when considering resource-constrained environments, is still evolving, as shown by the limited validation on actual devices [132, 181]. Current research primarily emphasizes the actual moving parameter (*what*), which increases adversarial complexity [181]. There is a notable lack of optimized MTD orchestration models, highlighted by the few methods incorporating multiple defenses [55]. Recognizing the decision problem of *which* MTD strategy to deploy is crucial for real-world applications, especially for IoT, which faces a diverse set of threats.

Table 3.8: Comparing MTD Policy Selection Methods [270]

Sol.	***Year***	***Malware***	***Method***	***Metrics***	***IoT***	***Eval.***
[154]	2017	Various	Markov game	None	No	Sim
[244]	2019	Various	Markov Robust	None	No	Sim
[299]	2019	V, M	*Joint Defense*	None	No	Sim
[239]	2020	Various	Signal game	None	No	Sim
[298]	2020	Various	Dynamic Markov	None	No	Sim
[245]	2021	ATP	Differential Game	None	No	Sim
[283]	2021	Crossfire	Simple Algorithm	Packages	No	Sim
[210]	2021	DoS	Simple Algorithm	CPU	No	Sim

M = Multiple, Sim = Simulation

In [44], multiple moving parameters across architectural layers are recommended to enhance deployment effectiveness. Some proposals, like [154], use *Markov Decision Processes* (MDP) and game theory to determine optimal MTD strategies with methods such as MG-MTD, which selects strategies based on defense tactics including hopping elements, strategy, and timing (*i.e.*, *when*). Their case study evaluated this approach on a network-oriented setup without considering constrained devices. Similarly, *MRG-MTD* described in [244] combines MDP with robust game theory to optimize defense, simulating a topology that excludes resource-limited devices. Using genetic algorithms, [299]'s *Joint Defense* strategy optimizes defense responses by generating various mutation configurations that are effective under multiple simulated attack scenarios. A signal game approach by [239] enables defense strategy optimization based on equilibrium solutions, where attackers respond to defender-generated signals. [298] introduces a differential game model incorporating MDP to balance timing and uncertainty. The work is assessed on a business network with defense elements without IoT components. *FlipIt* uses game theory in policy selection for optimal MTD strategy deployment, emulating an SDN platform that is not aligned with defense capacities considered in this scenario [245]. Meanwhile, [210] implemented a policy selection strategy that activates MTD methods like shuffling

and redundancy upon hitting 40% CPU utilization. The evaluation of the approach was executed in a cloud simulation environment.

In summary, current literature reveals gaps in deploying multiple MTD techniques (*i.e.*, *which* technique to deploy) within IoT contexts, as real-world testing remains scarce (*cf.*, Table 3.8). In addition to the lack of real-world testing, investigating specific metrics and methods within the orchestration system is scarce, especially when considering malware threats.

Attack Avoidance

The usage of data-driven approaches is prominent for cyber defense. Unfortunately, there are also many pitfalls when employing ML-based systems. [12] postulate that an inadequate threat model is one of the critical pitfalls for such systems. This review analyzes the detection system by considering a particular defense and threat model to assess whether the data collection and attack detection approach is sensible. The threat model considered here is the following: a system is breached by user space ransomware. It is assumed that a defense system could rely on data collected in the kernel but not in user space since any user space execution is inherently untrustworthy. In addition, it is analyzed whether the detection signal that is made available could be used to avoid damage. In other words, it is discussed whether damage (*i.e.*, damage is already partially or fully done) or an early indicator is detected. Furthermore, the rise of self-adapting ransomware [266] is considered. Hence, whether the defense method allows an attacker to learn and refine its attack strategy is discussed.

Cryptolock alerts users of the Windows operating system of potential encryption threats. The system calculates a reputation score for each process, signaling suspicion levels over time. It also employs *sdhash* to compare file versions pre- and post-write operations. As scoring is computed post-write, data recovery can only start after detection [272].

In [153], entropy-based ransomware detection is performed using file suffixes. The system computes and compares entropy values with backup data, with optional ML models such as KNN, linear models, and deep learning being employed. The authors find ML-based entropy detection promising, although real-world defense scenarios and file recovery approaches remain untested [153, 272].

Since the execution of ransomware may involve network interactions (*e.g.*, key exchanges, instructions), *RansomSpector* monitors file system and network activity. Using a hypervisor to capture system calls, the system checks for matches against malicious patterns. If the patterns of file and network behaviors align, a process is flagged as malicious. RansomSpector's evaluations focus on detection accuracy but not defensive effectiveness. Furthermore, its alert-based strategy may yield data loss, requiring extensive management of alerts [272].

Redemption introduces detection and prevention capabilities through mirrored file operations and entropy checks. Tested on Windows-based systems, this solution's evaluation primarily covers detection and resource usage, leaving real-world defense metrics unclear. Moreover, [266] warns that feedback to ransomware may enable attacker adaptation, as the solution lacks deceptive aspects [272].

ShieldFS provides self-healing ransomware protection for Windows-based platforms. It uses an I/O sniffer to gather entropy, process IDs, and timestamps, discarding malicious copies and terminating processes as needed. Upon testing using Virtual machines, the authors validated the detection and prevention capabilities but discovered limitations like DoS vulnerability and adaptation risks [266, 272].

In [267], Linux-based ransomware detection relies on kernel-derived performance metrics, offering MTD techniques after successful classification. These methods can terminate a running ransomware but risk data loss and attacker signaling. Proactive MTD deployments were considered to be inefficient due to resource consumption.

With ransomware's adaptive nature, ongoing evolution in detection is essential. *FeSAD* introduces a binary classification for the Windows API calls and supports potential defense mechanisms like isolation and termination, but lacks preventive and deceptive features [272].

[268] extends MTD defenses from [267] to include recursive directory traps and suffix obfuscation in Linux-based operating systems. In testing, it effectively blocked certain types of ransomware based on their traversal strategy. However, impacts on benign processes remained unassessed, and partial data loss was accepted depending on the adversary's strategy [268, 272].

Seeking enhanced explainability in detection models, *XRan* employs a Convolutional Neural Network to provide local and global insights using dynamic and static sources. While it improves detection, real-world defense scenarios and cross-platform compatibility are not discussed [272].

Table 3.9 categorizes these systems based on mitigation, prevention, detection strategy, underlying platform, data collection methods, and evaluation metrics. Even though work on OS-level defense is advancing, five challenges persist. First, there are only limited studies focusing on ransomware targeting Linux-based systems. Secondly, a minimal understanding of ransomware mitigation exists outside of halting their processes. Thirdly, defense scenarios are rarely considered when evaluating detection systems; instead, studies focus on purely analytical factors (*e.g.*, model accuracy or recall), making it unclear whether such a detection system is viable for defense. Current detection and defense systems may expose feedback to attackers, which can be used to learn optimal adversarial patterns dynamically. Finally, the previously identified limitation is reiterated: evaluation often excludes interaction with real-world malware samples [272].

Table 3.9: Categorization of Surveyed Related Work [272]

Work	*Year*	*Mitigation*	*Prevent*	*Platform*	*Evaluation*	*Data Collection*
[188]	2016	Alerts, Termination	✗	Windows	D|M	User Data
[62]	2016	Shadow Drive Buffering	✓	Windows	D|M|P|U	IRPLogger
[8]	2017	Buffers, Termination	✓	Windows	D|P|U	I/O System
[153]	2019	Backup Recovery	✗	Storage	D	✗
[79]	2020	User Notification	✗	Windows	D	System Calls
[267]	2023	Trapping, Hiding	✗	Linux	D|P	Performance Metrics
[268]	2023	MTD: Trapping, Delays	✗	Linux	M|P	Performance Metrics
[83]	2024	Isolation, Termination	✗	Windows	D	Windows API Calls
[102]	2024	Explainable Alerts	✗	Windows	D|P	Windows API Calls

D=Detection, M=Mitigation, P=Performance, U=Usability Impact

3.2.3 Proactive Defense

Proactive approaches to defense present an alternative or complementary approach to reactive defense. For example, in cases where a particular detection system cannot be deployed on a platform, proactive approaches could be employed. Similarly, undetected attacks (*i.e.*, false negatives during detection) in a reactive defense scenario could be mitigated through a proactive second-line defense. The following review focuses on the MTD paradigm to reduce attack success probability. More specifically, reactive OS-level approaches are investigated due to the previously described threat and defense model since these apply to domains without access to external security components (*e.g.*, centralized monitoring or defense orchestration).

As previously stated, a common MTD limitation exists: many approaches rely on third-party actors and lack maturity in terms of implementation and evaluation [181]. This review further highlights this by investigating key design decisions of related approaches, described through *What*, *How*, and *When* they adjust the attack surface. As shown in Table 3.10, numerous strategies have been devised to counter specific attack types. In addition, the implementation is a relevant factor since it enables realistic evaluations and experiments [268].

Both [82] and [104] use shuffling strategies to address attack types like Denial-of-Service, botnets, and browser-based threats. However, they do not apply or validate these within an operating system in live scenarios. Application-layer MTD methods using event-driven diversification are proposed by [216] and [2], focusing on threats to ML-based systems. Additionally, [34] and [70] concentrate on the early stages in the cyber kill chain model – addressing reconnaissance attacks, leveraging specific network devices, such as Access Points and CAN Buses. However, only [252] employ operating

Table 3.10: OS-level Moving Target Defense [268]

Sol.	*Year*	*Attack*	*What?*	*How?*	*When?*	*Impl.*
[104]	2019	Fingerprinting	Browser	Shuff.	Events	Idea
[216]	2019	Input	Algorithm	Divers.	Events	Live
[2]	2021	Input	Model	Divers.	Events	Live
[281]	2020	Memory	Deployment	Redund.	Events	Live
[252]	2014	Reconn.	OS	Divers.	Interval	Live
[34]	2020	Reconn.	CAN bus	Shuff.	Proact.	Virt.
[70]	2020	Reconn.	Access Point	Redund.	Events	Idea
[82]	2023	Botnets	Ports	Shuff.	Events	Virt.
[267]	2023	Ransomware	Files	Shuff.	Events	Live
[214]	2024	Reconn.	SDN, SDN	Divers.	Proact.	Virt.

system-based solutions. Here, operating systems or their distributions are shuffled for attack evasion. A recent study [214] incorporates both OS-level deceptions and decoy nodes to increase complexity, presenting IoT-centered simulation results without introducing an actual implementation. Although it addresses different threat vectors, such as reconnaissance, the study illustrates that an OS-level defense can support proactive orchestration. [267] offers an MTD framework to mitigate ransomware by shuffling among real files created at the application level within user space, exploring both reactive and proactive approaches. The authors conclude that proactive approaches are less efficient than reactive ones [268], presenting an opportunity for optimization.

Beyond the MTD paradigm, the previously described attack surface of accessing the file system has also been investigated from other perspectives. As illustrated by prior research introduced in the previous section, ransomware, for example, can be effectively detected through file system monitoring, with some solutions pairing detection with file recovery. These findings reinforce the potential of file system-based approaches for MTD, as existing methods have primarily focused on detection and recovery rather than proactive malware prevention [268].

In summary, existing MTD research does not explore file system-based implementations, especially for proactive defense. At a broader level, OS-level solutions like [252] and [192] operate within the same abstraction and trust model as this work. However, no MTD strategies are implemented at the OS file system layer, although [267] shows that using real files can effectively counter malware. Since file system-based defenses outside the MTD paradigm have proven viable, an opportunity arises to explore file system-level defenses as a proactive and dynamic MTD solution without relying on other actors. Nevertheless, the proactive defense would require consideration of its resource cost.

3.2.4 Limitations and Opportunities

The analysis of the state of the art presented in this chapter, established through various methods, including literature review, dynamic black box analysis, or vendor inquiry, highlights several limitations and opportunities for the use cases in the described problem instances. These findings motivate the research that involves creating and evaluating the artifacts, which will be introduced in the subsequent chapters. Research on systematically identifying attacks is at the core of this thesis since it adopts the threat-oriented philosophy that is established in threat modeling. In addition, this thesis focuses on quantitative assessment in threat modeling, *i.e.*, on a particular downstream threat modeling method. For those research lines, the following limitations and opportunities are addressed:

1. Practitioners and researchers agree on the relevance of collaboration for threat modeling. However, no studies or solutions focused on the role of modelers being separated by time, location, or function. This presents an opportunity to develop and evaluate artifacts to understand how threat modeling tools and methods can contribute in terms of guidance and automation.

2. The community agrees on the relevance of technological trends in threat modeling. Concerning the development and integration of AI-based systems, studies have described various attack vectors and manually built threat models for particular systems. However, no studies have investigated how tools and methods can contribute to guiding tools that enable systematic threat modeling, impact and risk assessment, and control identification – yielding an opportunity for design research.

3. Although human factors are acknowledged as a relevant dimension in information security, studies have not explored how tools and methods can exploit existing process models to conduct systematic threat modeling of insider attacks. Thus, there is an opportunity to design and evaluate such artifacts in practical settings, yielding not only knowledge about insider threat modeling but also the usefulness of process modeling for information security.

4. Quantitative, business-oriented approaches for threat assessment and prioritization appear to be almost entirely absent when threat modeling is practiced. Existing approaches are either purely qualitative, quantitative based on technical aspects, or methods borrowed from risk assessment without practical experiences described in the literature.

In addition to modeling and quantifying threats, the remaining research lines follow the same perspective (*e.g.*, mitigating sets of threats instead of focusing on particular weaknesses), presenting research on threat detection, avoidance, prevention, and reaction. These address the following findings from the literature:

5. The feature selection and model-building stages in ML have been approached with behavioral and dynamically established data by the cyber security research community. Surprisingly, few studies have investigated how to collect that data. Research on designing such data collection platforms could explore how to do so efficiently and effectively, considering data reproducibility, automation, resource consumption, and data transferability.

6. Reactive MTD has been postulated as a key research opportunity in recent literature reviews. However, realistic defense systems must not just define *when* to deploy an MTD technique, but also *which* locally available technique to deploy against a threat within a particular threat model. This research opportunity questions which metrics can be used to make this decision and how effective this system is when tested in a realistic scenario.

7. Related to the previous limitation, autonomously operated devices are faced with a myriad of threats. Research on developing holistic MTD frameworks would require developing and evaluating several MTD techniques against a particular threat model.

8. The efficiency and effectiveness of proactive MTD are questioned in related studies. With respect to the scenario considered in this thesis, further research could investigate how to exploit OS-level techniques and optimize them for a lightweight defense that could act as a complementary line of defense.

9. Reactive MTD frameworks and malware detection systems commonly focus on detecting damage. While this might enable damage-limiting actions (*e.g.*, by disconnecting a device, terminating a process), research should investigate how to detect damage before it is applied and how to avoid it. A threat-oriented perspective would further require that detection be done with an appropriate threat model. In addition, the defense needs to consider that adversaries can learn from the defense signal, paving the way for research on deceptive mitigation.

It is wrong to suppose that if you can't measure it, you can't manage it – a costly myth.

— William Edwards Deming (1993)

4 Modeling and Prioritizing Threats

Developing threat models requires a systematic approach involving the understanding of the assets to be protected and the potential threat events endangering them. Furthermore, this procedure must consider the effectiveness (*i.e.*, the reduction, awareness, or acceptance of an asset's risk exposure) and efficiency (*i.e.*, the resources invested into this approach). As will be argued, this requires that threat modeling be approached with appropriate tooling and methodological guidance to accommodate the specific scenario considered. This chapter introduces several solutions that address the limitations and opportunities one through four (*cf.*, Section 3.2.4). Figure 4.1 contextualizes each solution according to the DSR frame: highlighting the environmental challenges addressed and the contributions to the knowledge base made, based on the evaluations performed.

4.1 Collaborative Threat Modeling

The role of collaboration in threat modeling is – although not explicitly investigated by related studies and supported by existing tools and methods – considered critical. *CoReTM* explores how four different collaborative settings can be supported for visual threat modeling. Furthermore, the work investigates the suitability of including cross-functional aspects with existing methodologies, such as

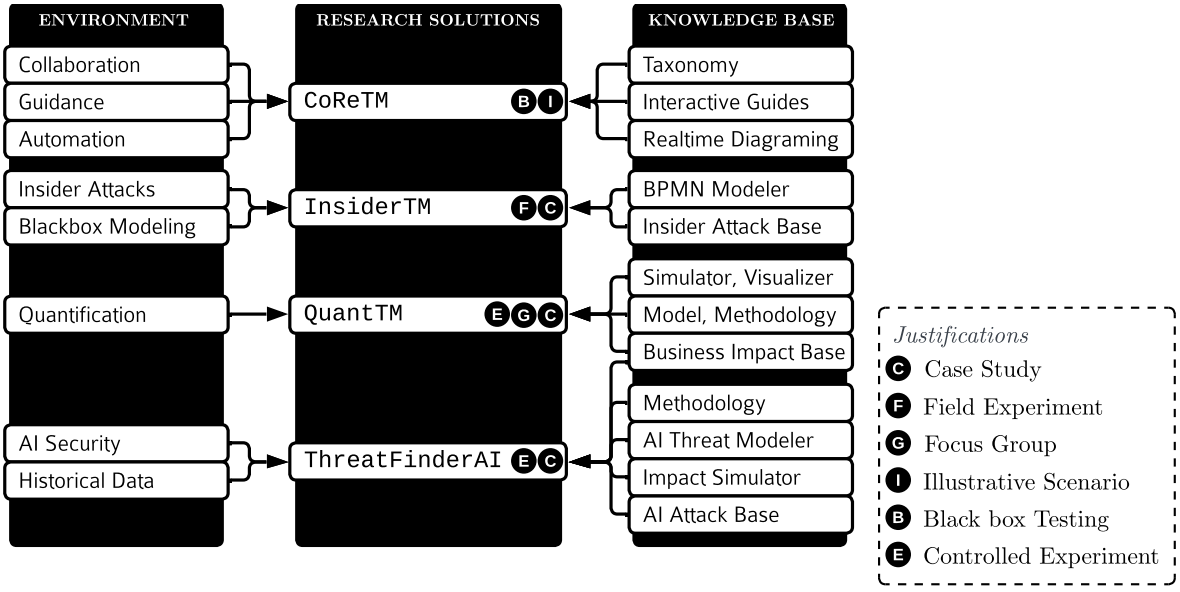

Figure 4.1: Threat Modeling Research: Environmental Challenges, Solutions, and Contributions

when representing contextual information from the business to which technical details could be related [274]. The design of *CoReTM* is oriented to the already popular whiteboards – aiming to present a toolkit that yields similar flexibility while simplifying aspects such as participation, persistence, and guidance. The result of this design study is a comprehensive methodology-agnostic threat modeling approach designed to facilitate threat modeling in collaborative environments, including fully distributed work settings. The approach is designed to support modeling in on-site, remote, or hybrid contexts, accommodating both asynchronous and synchronous participation. The visual threat modeling tool behind *CoReTM* includes *(i)* an annotation-based collaborative editor, *(ii)* automated threat reporting, and *(iii)* DevOps integration to support the downstream activities. Additionally, *CoReTM* fosters collaboration among a diverse group of stakeholders with varying backgrounds and expertise by featuring a modeling methodology selector, visual annotation libraries, and method descriptors [274].

4.1.1 Design and Architecture

CoReTM is a tool-supported approach designed to facilitate collaborative threat modeling among diverse, local, and remote stakeholders participating synchronously (*i.e.*, in real-time) or asynchronously. It is centered around a flexible modeling editor, including a threat modeling methodology selection procedure, a user action-driven guide, integrated knowledge bases, and publishing with downstream tracking systems. These aspects are detached from specific methodologies by explicitly incor-

porating an overarching meta-modeling process. Hence, the modelers can use various methodologies, including high-level ones like STRIDE and Attack Trees, technology-oriented methodologies (*e.g.*, OWASP, CAPEC), and domain-driven methodologies (*e.g.*, OCTAVE and PASTA). Representations of assets and threats are achieved by supporting annotating a variety of input formats [274].

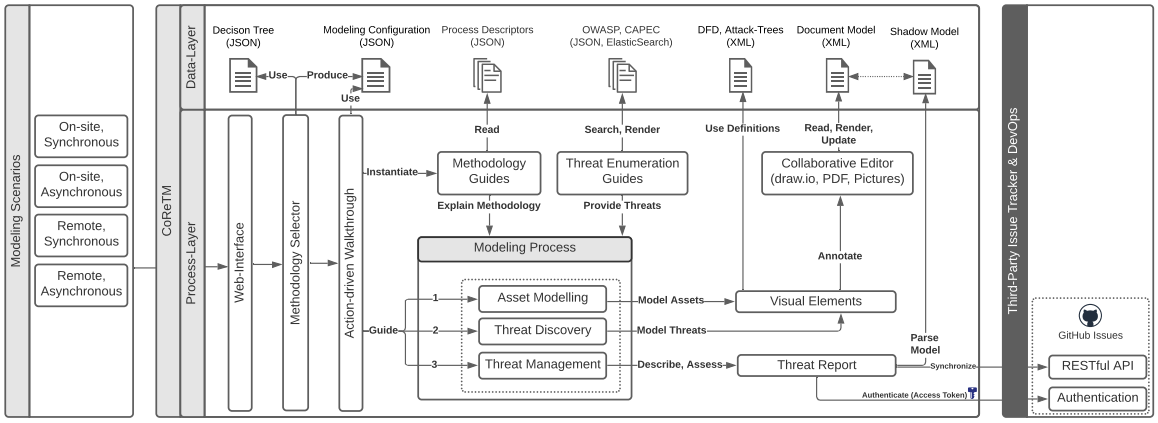

Figure 4.2: Overview of *CoReTM* Components and Interfaces [274]

The framework architecture of *CoReTM*, depicted in Figure 4.2, supports different scenarios for modeling threats, including on-site, remote, synchronous, and asynchronous meeting styles. All users leverage the *CoReTM* prototype through a unified web-based interface. Each key component is factorized into its dedicated *microfrontend* [205]. Initially, users are guided through the modeling process, where they can configure the scenario. The automated *walkthrough frontend* guides users through the setup procedure, defining a meeting style and selecting appropriate methodologies using a tree-based knowledge base. For instance, an administrator might create two separate workshop spaces: one for project managers to model high-level threats of a process model using a simple misuse case diagram and another to model threats of a particular system interface, such as a web-based application using a specific technology-oriented standard. Once the navigation of this component terminates, the workspace is prepared [274].

Participants access the web-based interface as part of a workshop or meeting, where the *guiding* component leads them through the previously identified threat modeling methodology, explaining and following its procedures. Depending on the methodology, additional *knowledge bases* are available and can be queried. In any case, the modeling process begins with visually identifying assets. Since efficiency is a crucial concern of threat modeling, *CoReTM* aims to be as flexible as possible and reuse any existing asset representation a customer may already hold. Thus, the philosophy behind [274] follows the one postulated by [253], prioritizing collaboration over narrowly operating

tools. Nevertheless, one must acknowledge the diversity and heterogeneity of artifacts, which are addressed through a *modeling component* that can render and annotate unstructured data such as PDF documents and pictures. In addition, users can create new models using a diagram editor. Both modes of usage employ annotations to define semantics, allowing for automated report generation and integration with downstream systems even when sourcing multiple input streams [274].

Upon modeling assets, the walkthrough component guides the identification of threats. Discovered threats are attached to assets by the users using annotations, enabling automated processing and reporting of threats. Three components support threat identification: *(i)* graphical elements ensure automated processing of the annotations, *(ii)* process descriptions, and *(iii)* knowledge bases interactively guide the underlying modeling methodology [274]. All identified threats are automatically compiled into a report, where users can describe and prioritize threats. This allows teams to leverage heterogeneous data sources, methodologies, and users to create a single threat modeling report. In the end, modelers can link identified attacks to GitHub issues using its API. This enables the threat model to be automatically changed based on a development pipeline [274].

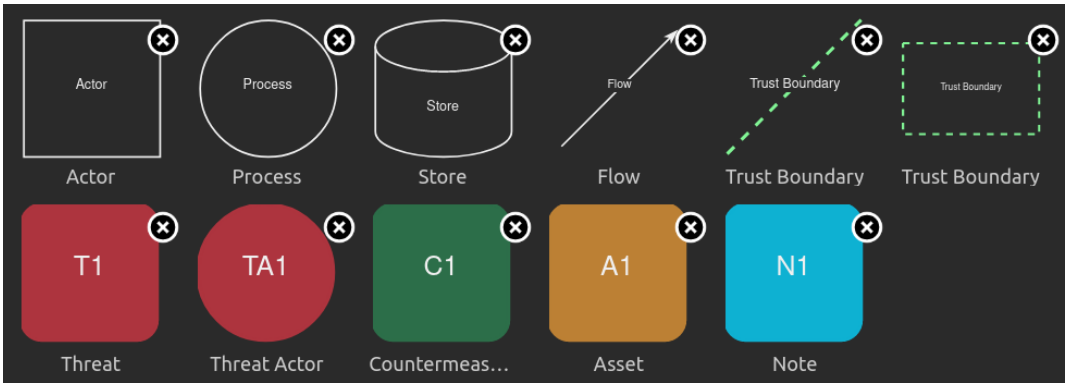

Figure 4.3: Visual Stencil Library Provided by *CoReTM* [274]

Visual Modeling

To accommodate different collaborative modes, the design of the visual modeling component is critical. A collaborative, real-time editor supports participation in different settings, providing annotation-based threat modeling using various methodologies, knowledge bases, and input sources. Real-time editing is facilitated by a versatile editor that supports both diagram-based and unstructured artifact modeling. The XML-based editor allows the creation and ingestion of diagrams, ensuring compatibility with all supported methodologies and specific domains. *CoReTM* includes a set of abstract visual elements that carry additional metadata for machine-readability. Custom libraries are provided for *Data-Flow-Diagram* (DFD) and attack-tree modeling, enabling threat annotations on both new and

imported diagrams. Two mechanisms support threat annotation: a specific color palette for color-coding elements in existing diagrams instead of replacing elements in the diagram. Additionally, a set of visual elements allows the creation of new diagrams. Figure 4.3 shows the provided elements for modeling using a DFD, and Listing 4.1 displays the internal representation of a *coretm-threat* element, highlighting the annotation meta-data. Through this internal metadata, the semantics can later be retrieved for automated processing [274].

```
<mxGraphModel>
  <root>
    <object coretm-type="threat" id="2">
      <mxCell vertex="1" parent="1">
        <mxGeometry width="80" height="80">
      </mxCell>
    </object>
  </root>
</mxGraphModel>
```

Listing 4.1: Internal Representation of a Threat Elements [274]

By using this editor and its visual libraries, users can apply different methodologies to model threats. To support all meeting styles, the model's underlying representation must be carefully synchronized between editors. For asynchronous modeling, a shared workspace with centralized model access is sufficient. Additionally, asynchronous modeling benefits from the functionality where users can place notes within the workspace for asynchronous control flow. For example, two editors could be set up with instructions for different participants to focus on a particular area of the architecture. Synchronous modeling poses greater (technical) challenges due to potential synchronization conflicts. Reliable transport is ensured through the web-based interface, and a dependable storage component mirrors changed models to other participants. Unlike other solutions, models are not exchanged at regular intervals, and differential synchronization algorithms are not used, as they rely on fuzzy text-based methods unsuitable for structured models [92]. Each workspace is divided into multiple documents to minimize collisions, with changes synchronized using a three-way merge approach [274].

Integrating Methodologies and Tooling

Section 3.1 introduced several established methodologies and tools to conduct threat modeling. Some offered advanced features such as automated identification using source code or integration into a DevOps workflow. However, reliance on such data arguably creates entry barriers. For instance, these

tools might not support creating a threat model for an architecture if the software is being developed from scratch. Using code as a medium for modeling may exclude stakeholders like program managers, architects, testers, or requirements engineers. Moreover, attack enumeration-based methodologies (*e.g.*, *OWASP* or *CAPEC*) are technology-specific and limited to specific scenarios. *CoReTM* approaches this issue with two features. First, it provides an abstract modeling process that can integrate existing methodologies for different macroscopic steps [274]. Based on a survey of existing methodologies, *CoReTM* proposes a suitable threat modeling methodology according to applicability and requirements derived from an interactive questionnaire, as depicted in Figure 4.4. The chosen methodology is then applied using the annotations in the previously mentioned diagram editor – which abstracts the details of the methodology. This way, threats identified using any methodology are automatically compiled into a report. Secondly, *CoReTM* incorporates multiple supporting and guiding knowledge bases. Procedural knowledge bases interactively guide users through asset and threat identification according to the respective methodology. If applicable, threat libraries are provided that can be queried against keywords extracted from the asset models [274].

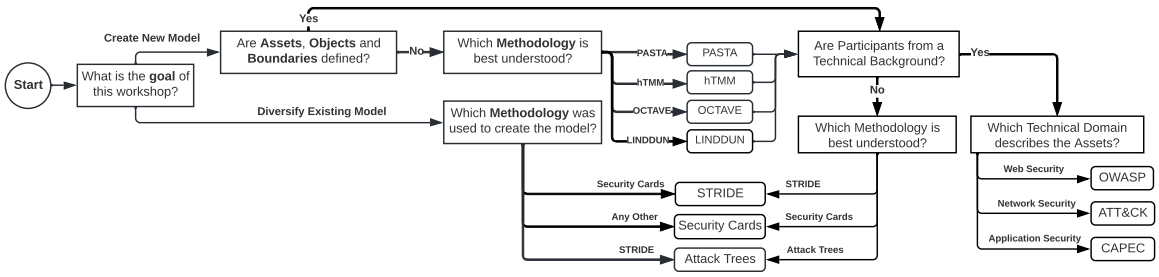

Figure 4.4: Threat Modeling Methodology Selection Questionnaire [274]

In addition to integrating with existing methods, *CoReTM* leverages the internal annotations to preserve the semantics within a threat model for downstream tooling. *CoReTM* preserves machine-readability by automatically creating reports that can be linked to other procedures. Linking the generated threat model to other processes ensures it provides value throughout the process. First, the model is parsed, extracting the annotations inherited from the stencil library. Using the embedded XML attributes and color codes (*i.e.*, the annotated metadata), the semantics of the elements are extracted, *e.g.*, answering whether a diagram element is a threat or an actor. Threats are then displayed in a tabular format with flexible options for additional user input. In asynchronous settings, users can discuss the report using text notes. Later, threats within a report can be converted into *GitHub issues* with a single click, triggering a request to the GitHub RESTful API. By employing these methods, the threat model can be utilized in other collaborative settings, such as in agile software development.

This approach also ensures the model remains up-to-date. For instance, when countermeasures are implemented and issues are resolved in the version control system, the threat model in *CoReTM* reflects this updated state, highlighting a potential reduction in risk exposure. This level of linkability is not achievable when threat models are created using third-party diagram editors, whiteboards, or spreadsheets – potentially leading to threat models that end up as unactionable reports [274].

4.1.2 Prototypical Implementation

The feasibility of the *CoReTM* approach was demonstrated through a prototypical implementation. Due to the widespread use of the *diagrams.net* editor among software engineers [212, 280], it was selected for integration. The complex nature of *diagrams.net* and its integrated storage solutions (*e.g.*, Google Drive and Microsoft OneDrive) are not open for further development [134]. Thus, a custom storage integration had to be devised. To enable collaborative editing, *diagrams. net* is embedded into a web-based application using *iframe* elements. Although directly extending the editor's codebase is not encouraged [134], the tool does provide a clear *JSON*-based messaging protocol [135, 274]. A subset of the resulting microfrontends is presented in Figure 4.5, representing a hypothetical scenario during the threat identification stage.

The platform was implemented using common web frameworks and a digital orchestration platform that offers shared storage. The web-based platform can be deployed off-premises or installed within an organization's infrastructure. A custom library of visual elements was developed for the *diagrams.net* editor since, by itself, it does not provide any processing of created diagrams. Another microfrontend processes the annotated model using a publish-and-subscribe data exchange, displaying identified threats in a table. Users can add information or synchronize threats with GitHub issues in this tabular representation. These elements address the general requirements of the approach and are not tied to any specific technology, standard, or process. Nevertheless, several methodology-specific frontends were developed to directly and interactively guide the specific methodology if it is selected by the user. The guided brainstorming component for threat discovery was implemented for the *STRIDE* methodology. Furthermore, the OWASP ASVS certification library is integrated as a microfrontend. Additional attack libraries, including custom enterprise-specific ones, can be integrated by developing new components, highlighting two advantages of this architecture. Firstly, by employing a microfrontend architecture, new components can be added without increasing the complexity of existing ones since they are entirely decoupled. Secondly, microfrontends operate as standalone applications, allowing multiple instances of the same component to co-exist in the same workspace. For example, two teams can use two instances of an editor in the same workspace, each

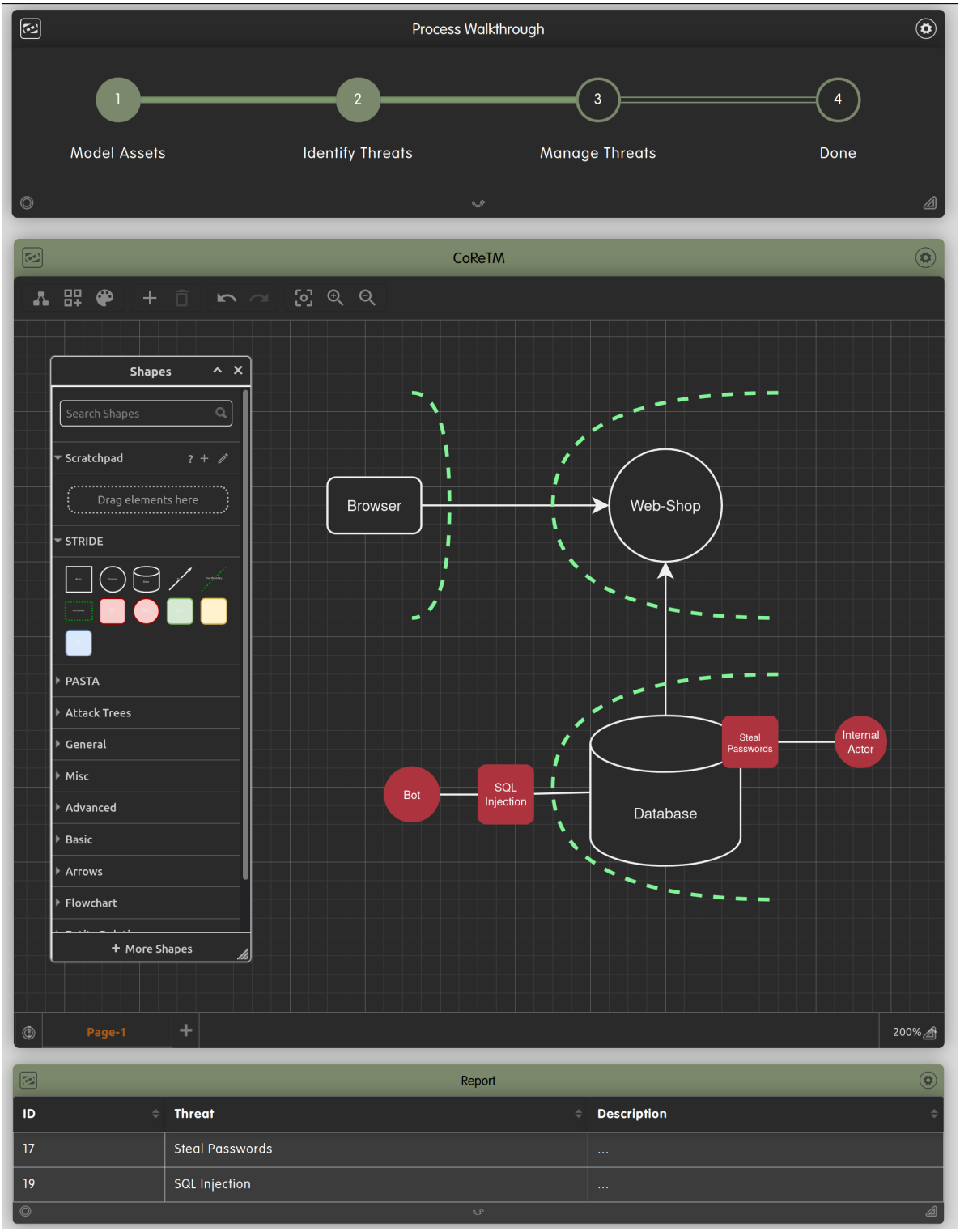

Figure 4.5: Key Prototype Components At Modeling Stage [274]

performing modeling with different methodologies, potentially leading to the discovery of a more significant number of threats. Similarly, two modelers could follow the same methodology asynchronously, combining and contrasting their results at a later point in time [274].

4.1.3 Evaluation

To illustrate the collaborative functionality provided by the *CoReTM* approach that covers previously stated limitations, the tool is deployed against a two-part illustrative scenario (*cf.*, Section 2.4.3). It is assumed that a government agency seeks to implement a digital COVID-19 certification system that offers digital proof of recovery, vaccination, or negative testing outcome [145]. As a reference, in the Swiss certification system, these claims are verified using an app that presents and parses a QR code. The provided claims contained therein are verified using digital signatures. Validators can then use this verified information – personal identification or health data is accessed through other channels [78, 274]. For example, a validated negative test result would require personal identification using an ID card to assess whether the person was actually tested.

This use case is selected as it demonstrates the need to analyze the entire digital certification system and not only its subcomponents (*i.e.*, focusing only on a purely technical perspective). Moreover, a digital certification approach involves various experts, including software engineers, healthcare workers, and government officials. To identify, trace, and mitigate threats, this illustrative scenario comprises a threat modeling workshop to gather domain knowledge, including the definition of high-level threats. Such high-level threats serve as an expression of antithetical requirements. For example, while a use-case specification could document the essential functions that should be offered, a misuse-case diagram could specify high-level risks. Developing this high-level model is followed by a technical threat assessment involving software engineers and security consultants, focusing on the sensitive areas and requirements identified in the workshop [274].

Modeling Misuse Cases

Adequate security measures require that the business value of the asset be understood. Therefore, the first use case is illustrated by conducting a threat analysis focusing on critical assets to ensure economic effectiveness. To develop an initial "misuse case" threat model, domain experts from a COVID-19 testing lab, a healthcare representative, and government personnel, alongside a software and security expert, could be involved. Due to the pandemic, technical experts work on-site while other stakeholders participate online. It is crucial that the findings from the workshop are reported to the head of the public health department [274].

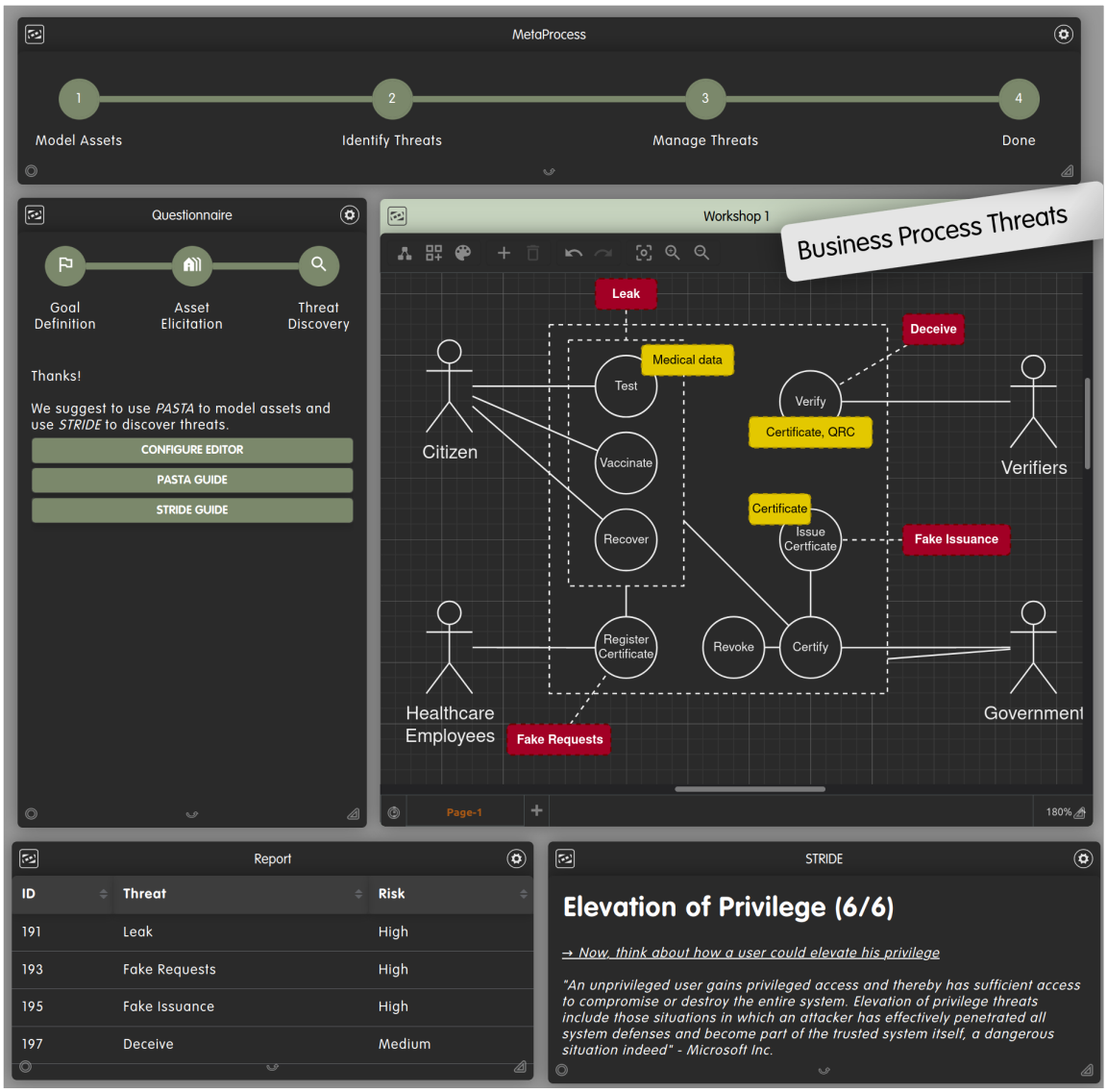

Figure 4.6: Workspace After Conducting the Misuse Case Modeling [274]

A cybersecurity expert would set up the workspace in the *CoReTM* via the web interface, opening a new instance of the *Questionnaire* to establish appropriate settings for the workshop. Since an existing business process model is available, the questionnaire categorizes the audience, noting that most stakeholders are not software engineers and are unfamiliar with threat modeling. Consequently, the tool suggests *STRIDE* as a high-level methodology. Based on this, the tool prepares the *editor* component to provide the stencils compatible with said methodology. The *walkthrough* component reacts to this selection, too – guiding the workshop, while the *STRIDE* component interactively provides methodological guidance. Figure 4.6 illustrates the various components used [274].

Once the workspace is prepared, the workshop leader invites participants to the platform. During the scheduled meeting, participants access the shared workspace in real-time, similar to a shared whiteboard. The architect uploads the existing process model, which the tool converts and stores in the internal modeling database. The solution then identifies assets based on simplified, colorized stencils that leverage the pre-defined color palette. For example, participants attach an asset label to the "testing" use case – noting that medical data is created during this use case. Participants identify high-level threats by applying the *STRIDE* methodology that roughly guides the brainstorming, discussion, and annotation activities. Since participants are mainly from a non-technical background, they convey abstract threats that correspond to sensitive outcomes, such as leaking medical data or fraud. The *walkthrough* then indicates the transition to the threat management phase. Here, threats are enumerated and qualitatively rated based on the sensitivity of the assets. At this stage, only domain knowledge is captured; metrics like vulnerability and attack cost are not yet considered [274].

Using *CoReTM*, the workshop yields the identification of two critical assets and three threats, even though participants lacked technical knowledge and were not able to meet onsite. For example, a key threat stems from authorized personnel launching insider attacks, highlighting the need for procedural (*i.e.*, not just purely technical) control strategies. Furthermore, since certificates contain personal data, there's a risk of information disclosure. Threats related to privilege escalation or data tampering in certificate issuance need further modeling and analysis with respect to the structural vulnerabilities of the supporting information systems. These findings can be shared interactively with the head of the department by providing access to the fully interactive workspace. These results set the stage for a second threat modeling iteration, where the critical threat areas are analyzed from a technical viewpoint while preserving the business context and considering non-technical considerations of threats and controls [274].

System Threat Modeling

In the second workshop, a remote security expert guides technically knowledgeable participants, including software engineers, architects, and testers. The guide participates asynchronously due to timezone differences. The *questionnaire* indicates that existing diagrams should be reused and instructs modeling threats via the knowledge base developed by *OWASP*. Thus, each component in the workspace is provisioned by the questionnaire component. The *OWASP* component guides the workshop using procedures and accessible knowledge. One of the interactive features is to elicit the "sensitivity level" that the knowledge base uses internally. Here, the highest level is automatically suggested, especially for components storing or processing medical data – a decision that the knowledge base automatically uses for configuration [274].

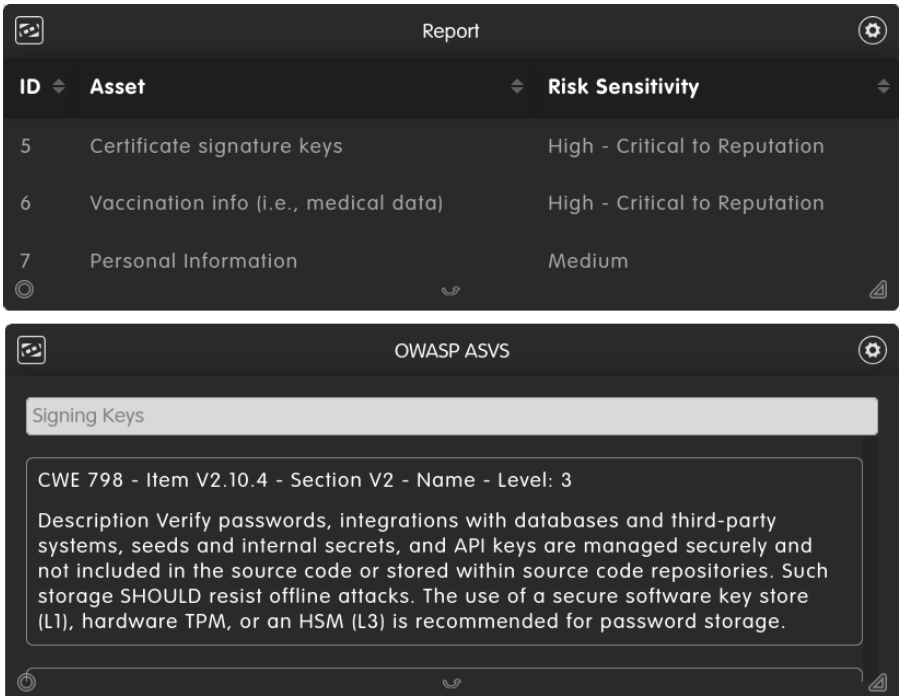

Figure 4.7: Deducing Threats and Controls from OWASP ASVS [274]

Threats are discovered by searching the catalog based on keywords or browsing the categories. For example, searching for "cryptographic keys" – a term *CoReTM* extracted from an asset element – yields recommendations for secure key storage. Similarly, searching for "medical data" yields threats related to data protection at rest. These threats are annotated on the diagram and thereby compiled into the reporting frontend. Thus, critical threats relating to the management of cryptographic material and access to sensitive data are covered. For example, based on an automatically extracted asset

name, the library outlines offline attacks against the keys (*cf.*, Figure 4.7), warranting the adoption of a Hardware Security Module – a control measure that could require justification concerning its procurement. The findings established by this model can be integrated into the development process and used asynchronously, for example, by including them in planning a development sprint. This scenario illustrates and argues that technical personnel could benefit from collaboratively identifying software threats by sourcing knowledge from existing security standards and frameworks. Furthermore, due to the consideration of the previously created misuse case diagram, the business-sensitivity of assets can be considered [274].

4.1.4 Key Observations and Discussion

Research on the *CoReTM* approach highlighted several findings. First, threat modeling research appears well-aligned with DSR as an overarching research construct. For example, *proving* the correctness of a threat model produced by *CoReTM* is arguably infeasible – how would a "perfect" threat model look? What is the ground *truth* sensitivity of each of its threats? Instead, it appears more aligned for the researcher to approximate the *utility* of the artifact. Still, communicating the academic contributions of threat modeling artifacts is challenging. For example, *CoReTM* had been employed in the real-world scenario described in Section 3.1.3 – due to the sensitivity of the project, the results of this participatory case study could not have been communicated. Thus, the solution's functionality was illustrated using a related scenario grounded in a real-world case but executed synthetically.

As highlighted through this scenario and based on the real-world usage of *CoReTM*, it is argued that the tool is applicable to the four work settings described. It is illustrated that threat modeling can include non-technical threats through misuse cases that capture the sensitivity of high-level threats from the domain's perspective. These can be related to and refined from a technical perspective. Nevertheless, *CoReTM* still heavily uses the inclusion of expert stakeholders – either as domain or subject-matter experts. Therefore, this solution likely inherits limitations of the field as a whole, such as limited acceptance from software engineers and cost pressure. Here, automation is commonly sought to relieve those factors. *CoReTM* conceptually argues that not only can the widely focused threat identification stage be automated, but the overall application of existing threat modeling methodologies, such as STRIDE. *CoReTM* does not provide fully completed threat models as output but rather aims to automate as many cross-cutting concerns (*e.g.*, modeling assets, navigating methods, annotating threats, creating reports, transforming reports into actionable tasks) of a method. Due to this design philosophy, *CoReTM* v2.0 was developed in collaboration with an industrial partner, leading to the pilot described in Chapter 6, which serves as a now fully functional and publicly available prototype for one particular threat modeling methodology.

4.2 Modeling Insider Threats

As exemplified by the study on *CoReTM* and the participatory observations described in Section 3.1.5, this solution explores how to improve the limited research on insider threat modeling. Thus, this solution explicitly focuses on making human factors more explicit in threat modeling. More specifically, it questions to what degree the modeling using *trust boundaries* is sensible, where insider attacks represent a key counter-argument to do so. Concerning potential solutions to this issue, this solution hypothesizes whether process models could be exploited to derive relevant information about potential areas of weakness against insider attacks.

The design and evaluation of [277] contributes a solution artifact that takes BPMN models as input and guides the modeler through a proposed methodology where the solution automatically yields related threats. The work includes a threat library that relates potential attacks to standardized BPMN elements based on an attack taxonomy compiled from the literature. The artifact is evaluated in two parts – confronting it with both commercial and non-commercial scenarios.

4.2.1 Approach and Architecture

To develop a solution that provides insights into both the practical application of insider threat modeling and the usefulness of BPMN models for this context, the architecture shown in Figure 4.8 is proposed. The development of the solution and its description in the following paragraphs is macroscopically guided by abstract stages of threat modeling: Objective Identification, Assessment, Decomposition, Threat, and Vulnerability Identification. The methodological and structural components supporting it are introduced [277].

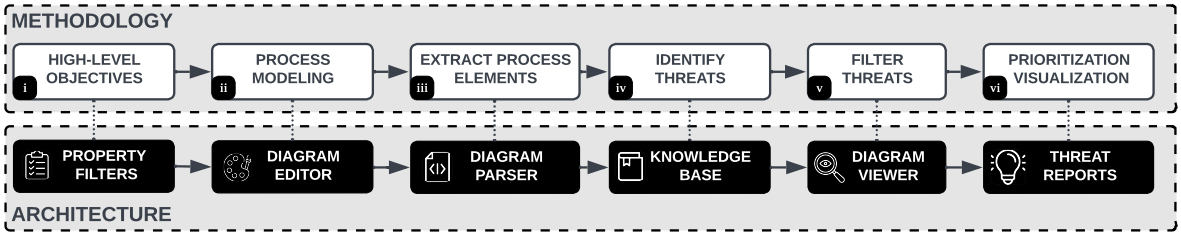

Figure 4.8: Approach and Key Components [277]

In the *objective identification* step, key security objectives are formulated by identifying a critical process and defining the guiding security properties using *property filters* (*e.g.*, Availability, Integrity) representing the CIAA principles. This step can be informed by goals such as modeling a specific

system or component or using business metrics. For example, [120] proposes to use *Earnings Before Interest and Taxes* (EBIT) to sketch the importance of a mission [277].

The *assessment* and *decomposition* steps involve modeling the process with the BPMN language. Existing models can be used for security analysis without introducing additional annotations. For example, in the aforementioned security analysis of a voting system, BPMN-based descriptions were readily available – not the system's internals. Similar to *CoReTM*, this solution leverages a visual modeling approach. Hence, an integrated *diagram editor* can be used to model the interactions, which a *diagram parser* automatically extracts, thereby decomposing the overall models [277].

The previously described step represents mainly an engineering aspect (*i.e.*, generically applying BPM, which is already well-described in the solution space). The approach to *threat identification* is a more novel research aspect of this solution. Here, a bespoke insider attack *knowledge base* is utilized, which is developed by reviewing and compiling knowledge from the literature, identifying commonly described insider threats. After reviewing multiple sources, 99 insider threats are elicited from five sources [277]. The threats in the knowledge base are structured using the security principles *Confidentiality*, *Integrity*, *Availability*, and *Authenticity*. A decomposition in terms of the number of threats found per publication is presented in Figure 4.9 Each threat is mapped to BPMN elements representing potential attack entry points. For instance, the threat of *viewing confidential data* can occur at rest (*e.g.*, when *data store* is accessed during a process) or during a *receive task* in the BPMN language. Threats are reviewed for each specific BPMN element in an expert-based manner, with two researchers discussing the attributions. In addition, the development emphasizes areas in BPMN involving human interaction (*i.e.*, activities, data objects, and message flows that could be affected by an inside actor) [277].

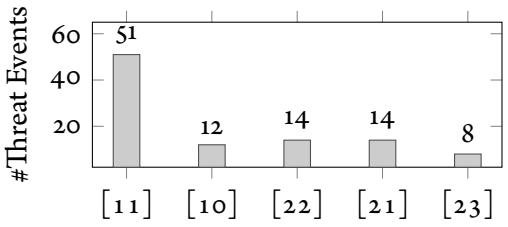

Figure 4.9: Knowledge Base Sources [277]

Based on this output, the knowledge base can be queried against extracted elements to support *threat identification*. Here, the users (*i.e.*, expert modelers) review and reduce the threats using domain knowledge. Still, this involves manual effort – where users have to identify threats during modeling manually: they must ensure that only realistic threats are included out of the automatically proposed ones. As an example, *credential theft* might not apply or be less sensitive if multifactor authentication (*i.e.*, a technical control, not directly visible from a procedural view) is in place [277].

The final stage, *vulnerability identification*, visualizes the resulting threat model by leveraging the input model. The aim is to identify areas that require additional security consideration and the inclu-

sion of additional controls. Since the input of the approach is a BPMN diagram, these areas can be highlighted in place. One could use internal company data to assess impacts or color-code steps to show threat frequency within the *diagram viewer*. Here, the solution emphasizes procedural elements with a high share of potential attacks (*i.e.*, indicating at which place the highest number of identified threats are located). However, this visualization must be carefully interpreted since this number may not inherently reflect higher risk exposure [277].

4.2.2 Web-based Implementation

A prototypical toolkit was implemented to provide a web-based access method for the methodology and knowledge base. This design was chosen due to the availability of mature BPMN diagram editors for web-based applications (*i.e.*, HTML and JavaScript). Furthermore, this enabled local and offline data persistence. The tool follows the methodology from Figure 4.8 [277].

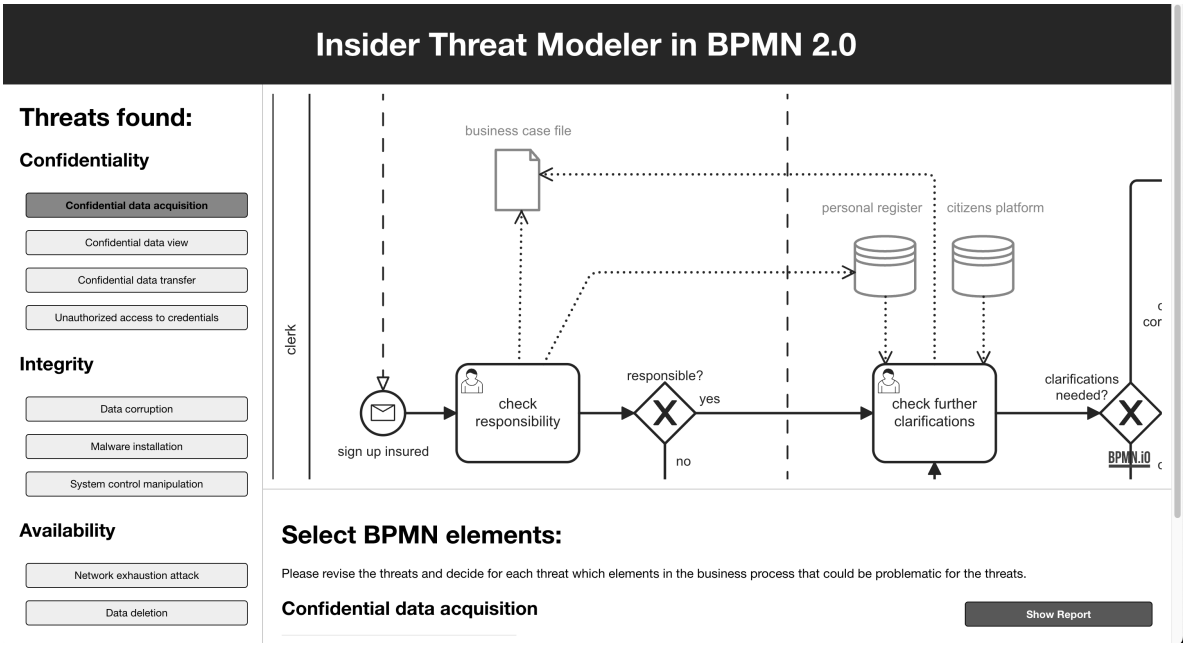

Figure 4.10: Reviewing Threats in Context [115]

On the first page, users define the security principles as indicated in the methodology. In the first iteration, users define the most sensitive properties. A diagram editor developed by the maintainers of the *bpmn.io* JavaScript library [40] was integrated to upload BPMN diagrams. Process elements can be extracted automatically using the JavaScript library and this editor. Then, the previously described theoretical knowledge base is ported to JavaScript, which presents a simple rule-based implementa-

tion that can be queried from the browser. Similarly, the editor enables the visualization of identified threats contextualized by the related element [277].

Users review the threats that the prototype suggests and decide in a binary manner whether they appear applicable. Figure 4.10 exemplifies this step from a prototypical perspective, demonstrating how threats are iteratively introduced in the visual context. Relevant attack scenarios are added to the threat model that is stored locally (*i.e.*, using the *localStorage* API). The report page visualizes selected threats, displaying them per affected element and summarizing their occurrence in the sidebar. Once all threats are reviewed, the resulting *threat report* can be exported to various formats, including PDF, SVG, or XML, for iterative and collaborative work [277]. Thus, the output can serve as an input to a more technical analysis, as previously illustrated for the *CoReTM* solution [274].

4.2.3 Evaluations

As argued when evaluating *CoReTM*, testing the correctness of modeling approaches is inherently a challenge (*e.g.*, how would one *prove* that the threats suggested by this solution exhaustively enumerate all insider attacks for any scenario?). One of the reasons contributing to this is described by [120]: cyber and information security are still considered emerging risks from the perspective of ERM. For such risks, ground truth information might not be available (in this context: a complete list of threats with comprehensive information on attack probability or impact) [139]. Therefore, the evaluations follow this philosophy. They question whether the solution's approach can be practically applied to real-world processes and, if so, if there are any indicators that the approach actually contributes valuable insights. Besides testing whether the tool can leverage input in terms of realistic BPMN models, the output is assessed, too. Here, the evaluations focus on the relevance of the provided output, *i.e.*, whether the solution can offer relevant threats that could be used to control the underlying risk. In the following sections, a case study tests the tool's applicability in a commercial setting. Later, an expert-based user experiment considers the tool's application in a governmental (*i.e.*, non-commercial) context [277].

Case Study

The research methodology behind this case study exhibits participatory aspects, described by [118]: Hence, both academic and non-academic stakeholders are involved – representing subject and domain experts. Critically, the complexities and context of a real-world setting are preserved through the artifact used as input (*i.e.*, a process model). This is crucial for providing an unbiased view of the BPMN models' usefulness as input [277].

Two criteria were discussed to discover potential participants (*i.e.*, companies): formalized processes via BPMN and commercial operation must be given. Unfortunately, this excluded several of the initially contacted companies since they did not meet the criteria. For example, one early-stage software engineering company completely lacked formally defined processes. Four companies fitting the requirements were invited as participants in the case study. Ultimately, one company was willing to participate and was thus selected for in-depth analysis. The selected company provides IT solutions for social insurance organizations, aiding in digitizing their day-to-day business. The business process provided by the participating organization details the tasks a clerk in an insurance organization would have to complete when an insuree (*i.e.*, a citizen) requests the issuance of an insurance number and card. The process includes ten tasks, two data stores, two artifacts, and an intermediate catch event to receive a message. It spans three swim lanes with the clerk as the central actor [277]. For a complete representation of the tasks' complexity, refer to [116].

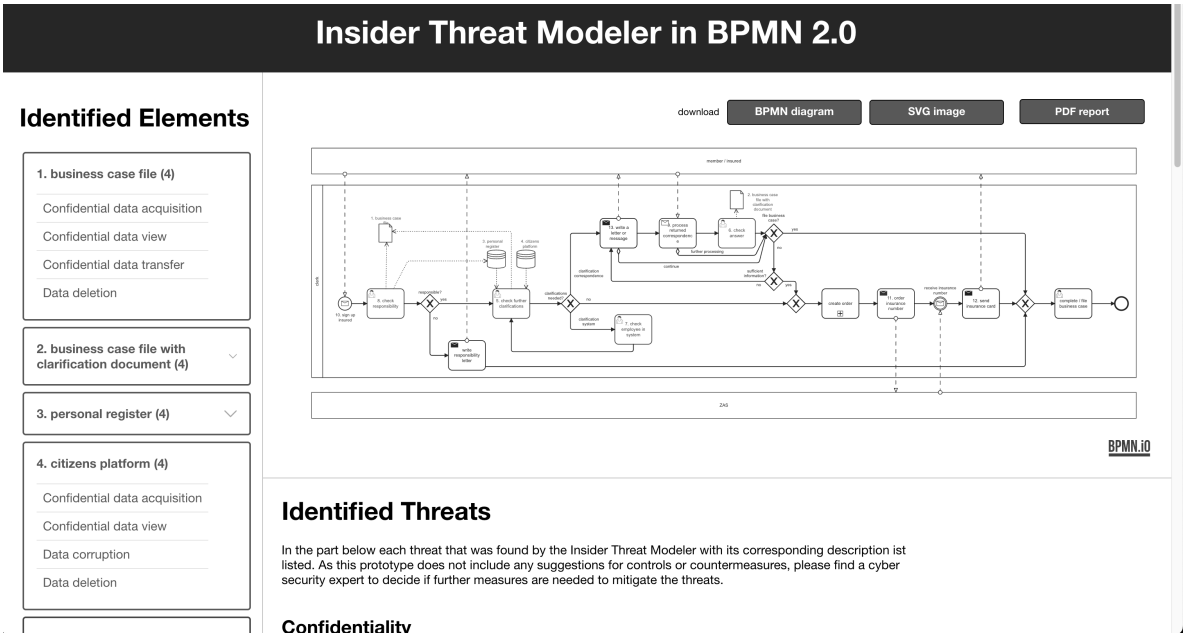

Figure 4.11: Reported Insider Threat Model [277]

After receiving the process description from the organization, it was converted into the BPMN 2.0 format, and sensitive information was removed to maintain confidentiality during analysis. The process was then loaded into the web-based application, and each step, as previously introduced, was executed. Additional input from the enterprise was collected, defining the key security requirements based on which filtering was conducted. The reasoning behind each decision was documented in written form. The resulting threat model and the logic behind its creation were presented to the com-

Table 4.1: Insider Threat Report [277]

Asset	*Type*	*Insider Threats*
Business Case File	Data Object	DA, DV, DT, DD
Case File, Clarification Document	Data Object	DA, DV, DT, DD
Personal Register	Data Store	DA, DV, DC, DD
Citizens Platform	Data Store	DA, DV, DC, DD
Check Further Clarifications	User Task	DC, SC, DD
Check Answer	User Task	DC, SC, DD
Check Employee in System	User Task	DC, SC, DD
Check Responsibility	User Task	DC, SC, DD
Process Return Correspondence	Message Receive	DV, MI
Sign Up Insuree	Message Receive	DV, MI
Order Insurance Number	Message Send	DT, DC
Send Insurance Card	Message Send	DT
Notification Letter	Message Send	DC

DA=Data Acquisition, DV=View, DT=Transfer, DC=Corruption, DD=Deletion, SC= System Control Manipulation, MI=Malware Installation

pany to understand the perception of the tool's output. To guide and capture this step, a questionnaire was designed to gather feedback. The questionnaire was distributed in a synchronous discussion session involving business owners and company security experts [277].

Figure 4.11 shows the resulting report page of the case study, illustrating the threat overview, with insider attacks being grouped and displayed per asset. Additional information on each threat is provided below the process diagram. The complete set of insider threats comprised seven key threats (*e.g.*, Data Acquisition, Data View) and thirteen assets (*e.g.*, process steps or artifacts). Table 4.1 summarizes the full threat model. The most critical areas of concern were the *Case File*, the *Personal Register*, and the *Citizen Platform*, where an inside actor could execute attacks. The visualization emphasizes these key assets, and the number of applicable threats is displayed in parentheses. Two qualify the threat model, the methodology, and the results were presented to the representatives. The upcoming feedback session was held with the enterprise's *Chief Information Security Officer* (CISO), their deputy, and the process architect. The results are summarized in four questions to which their answers are synthesized [277].

- **True positives** denote here whether the threats in the model are perceived as relevant. All discovered threats were deemed relevant, according to the three experts. They agreed on their significance since the data assets' confidentiality (both in digital and physical form) is highly

sensitive in this context. However, in this scenario, the company had already considered these threats in its own threat model, as evident from the existence of security controls for these threats. It is argued that this does not imply that the threat model is irrelevant, but instead, that the company already holds a mature security posture. In that sense, the congruence of the two threat models indicates that the solution could help present applicable threats if an organization holds a similar process model [277].

- **False negatives** refer to unidentified threats. The security expert noted that injection, privilege escalation, and social engineering attacks could be additional relevant threats [277].

- **Are there countermeasures for the identified elements?** The CISO enumerated organizational controls such as employee education, fine-grained access control, and confidentiality agreements. An audit trail ensures the traceability of modifications, and critical decision-making steps incorporate the four-eyes principle. In addition, technical controls supplement these organizational ones. For example, the risk stemming from *Data Deletion* threats is reduced through resilient backup and archiving procedures, *Data Corruption* through secure logging, and *Malware Injection* through filtering and sanitization of messages [277].

- **Are there procedural assets where threats other than those identified could be relevant?** The CISO indicated that the database could be vulnerable to DoS attacks. For example, an insider could launch complex database queries, diminishing the database's availability [277].

From the interaction with domain representatives, it can be concluded that the approach can provide effective output, suggesting relevant threats using a BPMN model as input. However, the output will unlikely represent all feasible attacks, which is significant given the biases encountered in risk management [120]. Therefore, careful consideration is needed when reasoning during threat identification and when communicating the results to other stakeholders. One explicit limitation discovered during the case study is that the prototype did not highlight threats *indirectly* impacting confidentiality, such as the mentioned injection attack, which would breach system integrity and could later lead to a data breach. Thus, additional efforts would be needed to model the transitive effects of attacks in the knowledge base or to employ different reasoning during the filtering step [277].

As mentioned before, the organization's security program had already addressed all the threats. While it might seem that the solution could not identify "novel threats", it also indicates the enterprise's program is mature. For instance, procedural controls like the four-eyes principle were not identified in the previously outlined motivating example detailing the role of insider attacks [242]. This suggests that the identified threats are relevant enough to justify investment in their mitigation [277].

Field Experiment

The second evaluation considered a non-commercial scenario and the inclusion of a domain security expert. This experiment examined an established but not formalized business process (although "business" does not refer to a commercial undertaking here). Specifically, the remote postal voting process described in the security analysis conducted by [149] is used as a scenario. According to the security analysis, the numerous activities and responsibilities involved in a voting business involve several processes and actors. For this experiment, only a subset of stages are considered: casting, storing, and tallying of votes. The experiment questioned whether a *security professional* (*i.e.*, participant one) could use the application to formalize the existing non-formalized voting process with BPMN and conduct a meaningful insider threat identification process. The expert had several years of technical experience in security engineering but no domain knowledge of voting systems and processes [277]. The models created during the experiment are available in [116].

Initially, the security expert was provided with the textual and visual descriptions from [149] – which was restricted to the domain aspects (*i.e.*, excluding the security analysis). Furthermore, the scope was explained, *i.e.*, to focus on *casting*, *storage*, and *tallying*. After launching the web-based application, the expert defined *Integrity* and *Confidentiality* as key objectives. The expert then formalized the process using the built-in BPMN editor according to the modeling language. The expert modeled different voting mechanisms and created three swim lanes representing sets of actors (*i.e.*, voters, postal service, municipality). In the third stage, threats were identified by filtering the suggestions according to his reasoning, yielding the final threat model. Seven unique threats were analyzed, some affecting several steps. In total, the threat model comprised 16 threat events related to 10 assets in the process. Without filtering by security goal or reasoning on its relevance, the knowledge base would have provided 52 threats for the BPMN elements. The experiment lasted two hours, including an introduction to the experiment, familiarization of the security expert with the application, establishing a basic understanding of the voting domain, and execution of the prototype [277].

Once the threat model was completed, the author of [149] – subsequently described as "domain expert"[1] – was contacted to subjectively assess the developed model. Without ground-truth threat models, the author's expertise was used for comparison, treating his judgment as a "gold standard" model. The report was provided to the domain expert as a PDF, with the process model containing the automatically generated color highlighting. Other than that, no explanations of the security expert's reasoning were provided. After 15 minutes of review, the following findings were obtained in a semi-formal interview, summarized according to the four previously used evaluation questions [277].

[1] The expert involved in this experiment was not an author of the publication on this solution.

- **True positives** Although efforts to interpret the attacks' description were needed, the domain expert confirmed that all threats in the report are highly relevant to the remote postal voting process as it is currently being operated in practice. The threat model highlighted the highest number of threats in the areas that the domain expert considers the most sensitive. Specifically, two areas are highlighted in the insider threat model: the temporary storage when votes are continuously received and held until the communal tallying process starts, and the store-and-forward processing in the postal service. Malicious actors could target these areas to view, corrupt, or destroy the voting material. In fact, this attack had previously occurred in practice: a local election worker destroyed voting material in 2020, shifting the outcome of the election [241, 277].

- **False negatives** The author identified multiple false negatives, notably artifact corruption (*i.e.*, modifying voting material), which could be relevant at almost all process steps, though the threat model indicated it in eight steps, with two steps missing the detail. Deletion attacks (*i.e.*, here referring to the destruction or hiding of voting material) were missing. This failure can be attributed to threats' association with the *Availability* property in the knowledge base. Thus, based on the assessment done by the expert, they were excluded, highlighting that properties may only provide initial guidance and should be questioned by iteratively refining the model. Finally, according to the domain expert, introducing malicious code could be more prevalent than modeled by the security expert [277].

- **Are related countermeasures already considered?** The critical threats discovered and modeled by the security expert were primarily located in the postal service's and municipalities' storage facilities. According to the domain expert's investigation, the postal service has been audited against several related standards (*e.g.*, ISO 27001, ISO 22301), implying the existence of information security controls. Nevertheless, it is unclear to what degree the specific threats are mitigated. Still, the domain expert noted that the key areas that are highlighted in the threat model (*i.e.*, , municipal vote storage) might not be covered by physical security controls. This is magnified when considering the fragmented nature of municipal operations [277].

- **Are there assets, aside from the ones mentioned in the threat model where threats could arise?** The security expert's formalization represented the domain well [277].

Based on these four aspects, highlighted by the author and additional views expressed by the author, several findings are emphasized. First, the application of the prototype by an expert who holds security expertise but no domain knowledge can be considered successful since relevant threats were

identified. If deployed in a real-world setting, pragmatic suggestions could be inferred based on the model without having access to internal representations of the systems. For example, the emphasis on the sensitive area of vote storage could foster discussions on whether additional physical and digital control mechanisms (*e.g.*, applying the four-eyes principle, using a two-key lockbox for sensitive material, conducting statistical plausibility tests) are necessary [277].

However, the experimental application of the prototype also led to the discovery of limitations. The time (and thus cost) required to adopt the tool and formalize undocumented processes may be too costly and only applicable for critical procedures where this expense can be justified. Therefore, it can be argued that the tool may be more suitable for enterprises and governments that already hold processes in the BPMN notation, where this step is omitted. Second, threats must be interpreted for the specific context since the descriptions are generic and not domain-specific. For example, in the postal voting threat model, a key threat is that the voting proof card could be stolen from the envelopes. Within the knowledge base, this proof is simply referred to as a "credential" and thus requires contextual interpretation. The third limitation covers threat prioritization in non-commercial settings. The initial prioritization may not be as sensible as in commercial settings since threats may not directly relate to an income stream. Taking this experiment's example, one could argue that integrity (*i.e.*, forging votes) is more impactful than confidentiality (*i.e.*, breaching ballot privacy). However, in practice, different stakeholders may hold different views on those threats' criticality. Thus, the domain expert argues that in the view of the population, "everything is relevant". In addition to these findings related to the experiment, the domain expert suggested usability improvements (*e.g.*, indexing the threats numerically or visualizing them within the diagram) [277].

4.2.4 Key Observations and Discussion

One must use caution when generalizing findings from observational and experimental research methods, especially when they involve a small number of samples, as in this case study and field experiment. Nevertheless, research on the design and evaluation of this artifact is summarized by several observations obtained from the practical instantiation of the approach.

The previous description of the *CoReTM* solution, including non-technical stakeholders and input artifacts, raised the question of whether such threats and data are well understood in threat modeling. This solution investigated this for a specific modeling approach – inquiring whether it is sensible to use procedural models following the BPMN language as an input for automated threat modeling. Based on the results obtained, there are indicators that such models could indeed be used to automatically propose relevant insider threats to an expert modeler. As illustrated, related research argued that process models are useful for other security practices. However, this study was the first to apply them

to real threat modeling scenarios and involve actual stakeholders of those domains. Nevertheless, the key limitations to emphasize are the small scale of the experiments and the discovered necessity of interpreting and communicating abstract threat descriptions for the specific problem domain.

From a technical perspective, the tool implementing the insider threat modeling approach was able to process the existing process models. Furthermore, no additional BPMN extensions were needed to illustrate the value of the prototype. The motivation of the prototype assumes that an organization has already formalized the process according to BPMN (as was the case in the case study). Additional research would be required to investigate whether it is sensible from an economic perspective to formalize processes just for semi-automated insider threat modeling.

Finally, this study argued that the solution was effective in both commercial and non-commercial settings. However, it was found that there are subtle differences in both problem contexts, which question the underlying methodology where modelers iteratively refine security objectives and investigate the related threats proposed by the tool. As noted by the voting security expert, due to the lack of a commercially driven endeavor, the relevance of assets and their sensitivity to threats varies greatly and is highly subjective.

In summary, while the tool can *suggest* several relevant threats based on potentially pre-existing BPMN models, it does not achieve full automation. It requires analysis and translation by a security expert. In that sense, it is safer to conclude that the methodology behind the solution can be guided (*i.e.*, guiding the steps and automating the annotation). Nevertheless, given the prevalence of insider attacks and their potential underrepresentation in security analyses, it is concluded that insider and process threat modeling should receive attention, complementing analysis from an application, hardware, or network point of view.

4.3 Quantifying Threats

Although the previous solutions focused on identifying threats without defining an explicit threat *assessment* approach, they still implicitly assumed that threats can be assessed (*i.e.*, analyzed concerning their sensitivity). This is inherent to the threat modeling discipline since, as the name implies, a model is developed. Thus, certain details from the real world are omitted to discuss the system more effectively. For example, the previous solution on insider threat modeling assumed that its users could gauge whether a particular insider attack was relevant enough to warrant explicit inclusion into the threat model.

Thus, although assessment can be considered a dedicated stage in threat modeling, it is essential to recall that threats are assessed at any stage – even when reasoning on the assets to be included,

which will inherently influence the discussion on threats. In the following, an overarching model and methodology are introduced that support explicit assessment through the quantification of threats, addressing the limited research on threat quantification. Notably, this solution termed *QuantTM* considers threat modeling and risk assessment as two distinct, albeit partially overlapping, disciplines. Common to both is that the quantitative approach has one objective: to elicit from the modelers' heads what they think about an attack's likelihood and degree of impact. The challenge of this activity remains in how this is to be reformulated and communicated externally, using explicit mathematical language [273].

4.3.1 The *QuantTM* Approach

QuantTM tackles the challenge of assessing threat models in a way that preserves the notions of the surrounding business context. *QuantTM* consists of a methodology and a set of factors to model. These can then be used to translate threat abstractions from the technical to the business context – discussing their sensitivity through quantification. To achieve this, *QuantTM* also proposes a method for identifying business impacts and a practical quantitative model to measure threats' potential financial impacts. To design the approach, the nuances between the threat modeling and risk assessment domains must be outlined. While *Threat modeling* is sometimes conflated with risk assessment [248], they are distinct activities. Threat modeling is generally considered an activity applied at the design time of the secure system development life cycle [248, 287]. Nevertheless, there are certainly overlaps with a risk assessment activity. Due to the fact that, in threat modeling, systems are likely not yet built and potentially not even designed, threat modeling aims to identify potential threats and their sources, requiring an abstract view of their impacts during quantification. For example, the specific vulnerability of an asset against a threat requires consideration of higher levels of uncertainty [273].

Given that the decisions made in cybersecurity impact all levels of an organization, *QuantTM* organizes its approach across the three conceptual layers that describe an organization according to NIST [138]. First, the *Organizational Layer* focuses on organization-wide strategic risks, *e.g.*, reputational loss from a data breach. Secondly, the *Business Process Layer* addresses threats to specific missions (*i.e.*, Human Resources or Procurement) that underpin those strategies. For instance, a Denial-of-Service attack could result in revenue loss from the perspective of a process. Thirdly, the layer encompassing *Information Systems* refers to system-specific design decisions. For example, one could implement rate-limiting for clients accessing a particular system component [273].

Figure 4.12 illustrates how the three-layered threat analysis methodology defined by *QuantTM* is decomposed into procedural elements. The methodology comprises several steps: the threat model

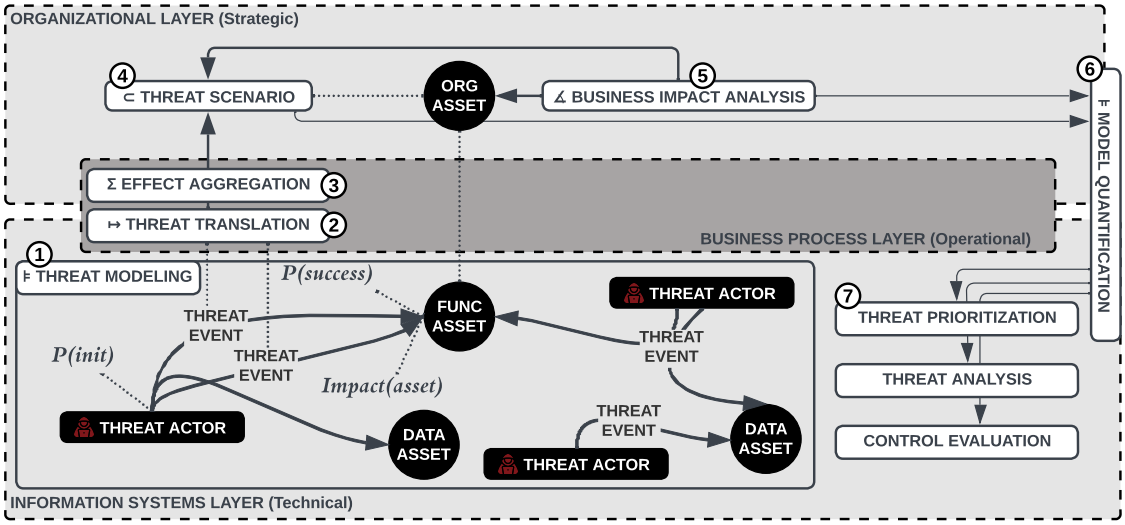

Figure 4.12: Procedural Elements (White) and Artifacts (Black) in *QuantTM* [273]

(Step 1) is defined at the *Information Systems* layer, translated (Steps 2 and 3), and interpreted (Step 4) at the *Business Process* layer, with business impact analysis (Step 5) being performed at the *Organizational* layer. Since *Business Impact Analysis* (BIA) is not yet explored as a source of information in threat modeling, *QuantTM* introduces a tailored method and specific set of impact measures. *QuantTM* integrates and thereby explores the viability of BIA and quantification into threat modeling. To achieve quantification, a generic set of properties (*i.e.*, applicable across domains) is defined at each procedural step, forming input variables for the quantitative model (Step 6) that provides economic impact measurement. Downstream activities (Step 7) utilize the output for *e.g.*, prioritization and control evaluation. In the following, each step of the process, the overarching model, and the prototype guiding BIA are introduced. Table 4.2 summarizes the terminology used throughout the procedure [273].

Information Systems Layer

The initial phase of the *QuantTM* methodology involves threat modeling (Step 1) at the (technical) *Information Systems* (IS) level. Here, a system-centric approach is employed to identify *threat actors*, potential *threat events*, and *target assets* that are affected. In the context of ERM, this asset identification stage could be aligned with the strategic definition of key assets. Hence, the overall scope and purpose of the threat modeling process should be clarified beforehand. Here, the first variables for the threat model are compiled: Along with documenting descriptions of threat events (*i.e.*, the set of

Table 4.2: *QuantTM* Terminology [273]

Term	*Explanation*	*Example**
T	Set of Potential Threat Events	{ DDoS Attack, Ransomware }
P_i	Likelihood of a Threat's Initiation	$P_i = 0.5$, Experiences a DDoS attack every 2 years
P_s	Likelihood of a Threat's Success	$P_i = 1.0$, Every initiated DDoS attack is successful
d	Duration of a Threat's (Technical) Harm	$d = 2d$, DDoS attacks last for 2 days
A	Set of Functional or Data Assets	{ Payment Service, Sales Data }
TM	Threat Model: Rel. between T and A	Ransomware Attack on Sales Data
E	Set of Harms to Assets	{ Data Loss, Service Unavailability }
deg	Degree to which a Harm affects an Asset	$deg = 0.8$, 80% of sales orders unprocessed
TS	Threat Scenario: Rel. between T and E	Data Loss due to Ransomware Attack
TS	Threat Scenario: Rel. between T and E	Data Loss due to Ransomware Attack
C	Set of Cost Factors	{ 1K of daily losses, 2K of one-time costs}
BIA	Relationship between TS and C	1K of daily losses and 2K of one-time costs due to a Ransomware Attack on Sales Data

*Depending on the availability of data and the quality of estimates, realistic examples would require additional parameters to describe their distributions. For example, d could be modeled as a 90% confidence of the downtime being between 4 and 8 hours.

threats T) and assets (*i.e.*, the set of assets A), an initial understanding of the actors can be modeled using the probabilistic variables P_i and P_s. These variables represent the likelihood of such attacks being attempted and their chances of success. Actual values can be established through calibrated estimates [117] and supplemented by empirical methods based on existing data sources [90]. Depending on the nature of the attack, an optional value d may be included to indicate the duration of a threat event's perceived impact on the continuity of the operation. However, the nature of various events will yield that $d = \infty$, as some impacts persist quasi-indefinitely. For example, while data exfiltration can be detected – the data's confidentiality would remain compromised and cannot realistically be restored. A DDoS attack, on the other hand, might only affect data availability temporarily. Note that this duration is modeled from a technical viewpoint on the asset's impact (*e.g.*, data integrity, service availability) rather than the business impact duration, which will be assessed in subsequent steps. For example, a data breach could imply a permanent technical loss, but reputational harm might not last as long. Conversely, a small-scale DDoS attack could result in business impacts (*e.g.*, weakened customer relationships) outlasting the technical duration of the attack. Thus, only the effects on the system's operations are described in this stage [273].

$$
\begin{aligned}
T &= \{t_1, t_2, \ldots, t_n\} \\
A &= \{a_1, a_2, \ldots, a_n\} \\
TM &= T \times A
\end{aligned} \tag{4.1}
$$

The resulting threat model is a relation between threat events and assets. Once a threat model *TM* is established at the technical level, a relation exists between the threat set and the assets. The aforementioned probabilities are recorded for each relation (*cf.*, Equation 4.1), yielding a five-tuple (*i.e.*, threat, asset, likelihood of initiation and success, and duration). This technically formulated threat model is translated and interpreted within the *Business Process* (BP) layer in the following step [273].

Business Process Layer

This layer focuses on translating (Step 2) each previously modeled threat into one or more threat effects. To do so, each technical asset is associated with the business asset it operationally supports. Modelers review each threat event and its impact on the business function's security property (*i.e.*, Confidentiality, Integrity, Availability, or Accountability). This step aims to strategically frame each technical description through its effects on the mission. For instance, a ransomware attack modeled at the IS layer for a particular system could translate to temporary service unavailability or, in consideration of extortion (*cf.*, Section 2.3.1), to confidentiality being permanently lost [273].

In addition to abstracting assets and threats, an additional reduction of the threat model can occur since the engagement of the next layer (*i.e.*, organizational layer) may require that threats have a relevant likelihood of occurrence and business impact processes. Thus, alongside defining each threat's effects (Step 3), an optional variable, *deg*, represents the degree of impact on an asset, considering its role in the process. For example, both a DDoS and ransomware attack might disrupt data availability. However, a DDoS attack may only limit the accessibility of public-facing interfaces, and internal employees may still operate on the data. In a hypothetical example, 20% of sales revenue is directly generated through back-office employees using another channel. The effect of the DDoS attack would be described as "data unavailability," with the business impact set to 80% since only external sales are impacted. With respect to gauging the likelihood, reasonably probable threats should be included. Still, biases, such as excluding rare but highly impactful threats, should be considered since even those threats qualify as strategic risks [120]. The goal is to include threats that may potentially occur in the coming years (*i.e.*, excluding highly theoretical threats). As illustrated in Section 4.3.3, this is a crucial difference from other methods [273].

$$
\begin{aligned}
E &= \{e_1, e_2, \ldots, e_n\} \\
TS &= T \times E
\end{aligned}
\tag{4.2}
$$

Gathering and transforming the required properties necessitates collaboration between business mission representatives and a technical representative involved in creating the IS-level threat model. The output here is a set of threat effects summarized by E associated with items in the threat model (see Equation 4.2). This results in an aggregated set of threat scenarios TS, enabling organization-wide, strategic discussions of mission impacts [273].

Organizational Layer

Once threat events, their likelihoods, the related organizational assets, and degrees of system impact are mapped to threat scenarios (Step 4), the overall business impact of these scenarios can be evaluated. While certain factors could be assessed on the BP layer (*e.g.*, revenue loss, investigation cost), certain intangible factors, especially those relating to strategic threats, may be invisible to mission managers. For instance, a manager responsible for one process might estimate the direct impact on sales figures. Intangible financial impacts, such as damage to reputation, relationships, employee morale, or opportunity costs, require a broader perspective than the one established by the mission. The research within the space of BIA has demonstrated how to quantify the costs of related incidents [297]. Additionally, emerging cyber risk management practices demonstrate the value of empirical data for approximating organizational impacts [88, 90, 273]. Since BIA is seldom applied directly in threat modeling, Section 4.3.2 provides an in-depth introduction to a novel, tailored BIA approach and prototypical solution.

$$
\begin{aligned}
C &= \{c_1, c_2, \ldots, c_n\} \\
BIA &= TS \times C
\end{aligned}
\tag{4.3}
$$

With respect to *QuantTM*'s overarching methodology, this BIA method is exemplified. However, the overall approach is agnostic to the specific assessment method as long as it generates a numerical cost measure for each identified threat effect (*cf.*, Equation 4.3). As highlighted, the approach records a set of cost factors, ideally represented as distributions of potential cost outcomes for the threat scenario. This structure involves a one-to-many mapping. Thus, a single threat scenario could result in multiple outcomes where impacts are characterized by different costs [273].

Quantitative Model

The quantitative model associates the underlying pillars once cost factors have been established (Step 5) and linked to threat scenarios related to specific threat events. As described, these events can then be quantified (Step 6) by generating a discounted cost metric, scaled according to probability and degree of impact, as represented in Equation 4.4. For example, cost factors C may be replaced with the single loss expectancy L of the threat effect, adjusted for the duration of the threat event [273].

$$Q(t) = \sum_{e_1}^{e_n} P_i(t) \times P_s(t) \times L(e_i, d = \infty, deg = 1.0) \tag{4.4}$$

All variables must be clearly documented so that they can be communicated back to the *Information Systems* (IS) layer since threat modeling is fundamentally a design activity. Thus, the output can influence the identification and evaluation of countermeasures (Step 7) using economic considerations. This systematic approach allows applying established economic metrics, such as the *Return on Security Investment* (ROSI) and the Gordon-Loeb model [30]. Access to this data on both the IS and BP layers is crucial to facilitate discussions with strategic management on cybersecurity budgeting to mitigate threats' impacts effectively [273].

Nevertheless, certain scenarios (*e.g.*, ones relating to critical risks or requiring substantial investments) may require the strategic management's awareness for optimal treatment. For example, the impact of malicious data encryption might be mitigated on the IS level with a read-only off-site backup policy. In contrast, negotiating a DDoS filtering agreement with the *Internet Service Provider* (ISP) might require strategic recognition of the dependency across systems and increased budget for procurement. The proposed quantitative model delivers explicit, numerically expressed impact assessments to address such diverse scenarios, enabling informed decision-making [273].

4.3.2 Prototypical Implementation

The previously outlined methodological steps of *QuantTM* are designed to establish a common framework for business-oriented threat quantification without narrowly defining how they shall be implemented. Still, demonstrating the feasibility of this approach hinges on an actual implementation. The following outlines implementation considerations for each layer to illustrate how the approach could be instantiated. Given that BIA is considered the least well-known component, a bespoke methodology and guiding prototype are introduced [273]. An open-source implementation of *QuantTM*, along with a publicly accessible instance, is provided at [178].

IS Threat Model

As illustrated by the previously described solutions, practical challenges exist already on the IS layer. Nevertheless, practical methods and tools are available to guide and support this stage. Although they exhibit methodological differences, they commonly involve identifying threat events and assets. Likely, the estimation of the other factors, such as likelihood and duration estimates, is still required and not supported by existing tools and methods. For example, *ThreatDragon* directly leaps to recording the impact of the threats using three qualitative levels. From a practical viewpoint, the adoption of a flexible threat modeling method [274] and the inclusion of diverse stakeholders would be necessary to provide the *QuantTM* factors [253, 273].

Listing 4.2: Modeling the distribution of P_i [273]

```
from scipy.stats import poisson

# Build mean for P_i
P_i_mean = min_incidents
    + (max_incidents - min_incidents)
    * prob_confidence

# Build Poisson distribution, sample 100'000 probabilities
n_samples = 100 * 1000
P_i_samples = poisson.rvs(P_i_mean, size=n_samples)
```

To assess the parameters beyond threat events and assets (*i.e.*, P_i, P_s, d), the authors propose modeling the initiation and success rate using probability distributions. Poisson distributions, as described in Section 2.4.2, are conceptually well-aligned for this task. Implementing such a distribution in Python simply involves collecting pessimistic and optimistic estimates within a user-defined confidence level, describing the expected occurrences. In Listing 4.2, this convolution is achieved using a Monte Carlo simulation using readily available Python libraries [273].

Operational Translation

After developing a threat model and describing its threat events, these details need to be interpreted within the context of the surrounding business process. This stage involves translating threats in collaboration between technical and non-technical stakeholders. Since the literature lacks coverage on interpreting threats during modeling, methods from ERM are evaluated. [120] highlights interviews and workshops as practical tools, although instances need to account for well-known biases. These must be considered when choosing a method to build the threat scenarios. For example, if stake-

holders consistently feel (or felt) their input is ignored, they may avoid participating or sharing relevant details. Additionally, these methods come with varying costs. Thus, one needs to consider the criticality of the related asset. In an ideal scenario, non-technical stakeholders are already included when collaboratively identifying threats in the previous stage, as highlighted in the *CoReTM* case study [273].

Business Impact Analysis Procedure

BIA is positioned as a central element. Previously, no details on how organizations should conduct such analyses were presented. Since BIA is not established in threat modeling, a specific BIA method is proposed and implemented, highlighting that it can be applied to outputs generated by the threat modeling and translation steps. Specifically, *QuantTM* proposes a new process to integrate BIA and a customized set of impacts aligned with abstract threat event descriptions. A guiding BIA prototype is also developed, providing a web-based instance of the approach [273].

BIA naturally takes a business continuity-oriented view on service disruptions. Besides identifying potentially disruptive events, the resulting impacts are considered, encompassing the analysis of a wide range of factors such as financial, legal, and technical. In practice, two challenges must be addressed: *(i)* the effects of threat scenarios on the business service must be identified, and *(ii)* the magnitude of those effects must be measured or estimated [273].

To conduct an impact analysis, a series of steps underline the proposed BIA process, yielding the desired quantified cost factors. A bespoke process is proposed to align the steps with the output of a threat modeling procedure. The proposed process and steps are outlined below, followed by prototypical considerations. Figure 4.13 overviews the steps and their resulting output [273].

The proposed BIA approach in *QuantTM* comprises six steps to model the impact of a given threat. Initially, a previously identified threat from the technical layer of the threat model presents the entry point. The first step, *i)* scenario definition, considers the threat within its mission context. For instance, a DDoS attack would be contextualized by the targeted information system and the supported business mission (*e.g.*, a DDoS attack on a web-based sales channel). To contextualize, the threat scenario is *(ii)* categorized based on its impact described by CIAA principles of the involved functional or data asset (*e.g.*, unavailability of the sales channel or data). Using these effect categories, various factors that *(iii)* map to the principles are derived from a knowledge base that was compiled from the literature. The BIA approach in *QuantTM* covers 16 different business impact factors, which were gathered from reports such as [176] and [63]. For example, product quality loss during a dis-

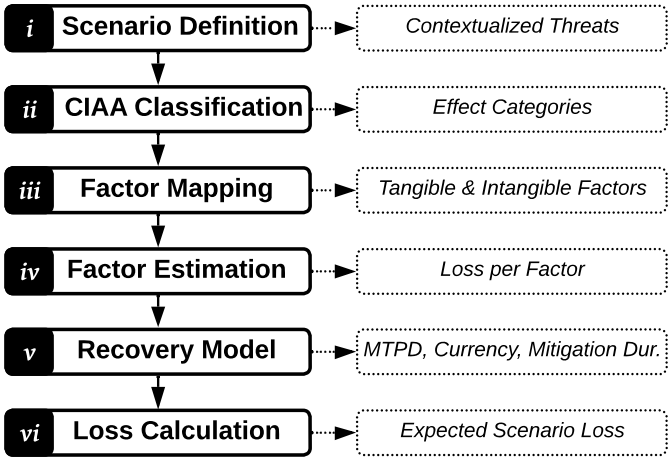

Figure 4.13: Steps and Artefacts of the BIA Approach [273]

ruption represents a tangible factor. An intangible example is the degradation of a customer relationship [273]. The complete set of factors can be found in the source code [178].

As an output of the current step, a set of proposed but extensible factors guides the analysis, acknowledging two assumptions: First, users can reason about the relevance of suggested impacts based on their domain knowledge. This is assumed to be realistic since business continuity managers are already familiar with BIA due to previous contingency planning [240]. Other practical methods include the bow-tie method to distinguish factors and impacts, which is described to be well-understood by non-technical stakeholders [120]. Secondly, the approach hinges on the assumption that users understand that the suggested impact factors are non-exhaustive and that the recommendation system cannot enumerate all potential impacts. As emphasized before, operators and managers of the approach would need to account for common biases in risk management [273].

The developed impact library offers tangible and intangible factors based on their relevance to security principles. For instance, in the case of an availability attack, the process would suggest considering the effect of *revenue loss* but not the impact of a *customer breach notification*, as data may not be leaked in this specific threat scenario. It is crucial to understand that these presented effects are merely guidelines; users can flexibly incorporate additional effects that may be specific to their business, industry, or use case. With those factors explicitly defined, *(iv)* loss estimation defines the quantification factors. The proposed BIA method suggests that those are modeled based on the type of threat and modeled using persistent and one-time loss functions. Once the critical business impacts are identified, *(v)* attack recovery, currency, and MTPD are recorded. These are compiled to facilitate the *(vi)* loss calculation for each threat [273]. Additional information may be required at this

stage. For instance, their daily loss and estimated duration are aggregated for persistent impacts (*i.e.*, time-dependent ones). To improve the accuracy of the loss trend, the recovery level can be modeled, which is defined as the relative degradation of a given situation compared to the baseline [273].

$$\begin{aligned} \text{Loss} = \sum_{i} \left(\sum_{t} \text{BI_pers}_{i} \times (1 - \text{rec}_{t}) \times \text{days}_{t} \right) \\ + \sum_{j} \text{BI_ot}_{j} \end{aligned} \tag{4.5}$$

The model in Equation 4.3.2 illustrates the relationship between threat scenarios and costs as outlined earlier in Equation 4.3. However, it does not yet incorporate the likelihood of events and only models the magnitude of impacts. In simple terms, the loss of a threat is modeled based on two components: one-time loss and persistent loss. Here, *i* and *j* represent individual persistent and one-time impacts, respectively, while *t* denotes the recovery stages of each persistent impact. `BI_pers` captures the loss that occurred through persistent impacts, `rec_t` indicates the recovery level of the business over time, and `days_t` represents the duration required to reach the baseline `rec_t`. Thus, it aggregates all persistent losses over recovery time, along with one-time losses [273].

Business Impact Analysis Prototype

The guiding prototype for the BIA approach enables users to execute the methodology for a specific threat scenario. This tool guides the analysis with partial automation while automatically gathering all relevant user input. The collected data is used to compute the aggregated losses of defined business impacts and visually break down the contributing factors [273].

Similar to the previous two prototypes, the *QuantTM* prototype addresses privacy by operating entirely on the client side (*i.e.*, within the user's user agent). No third-party services or components are involved. The web-based user interface is composed of several static HTML files, and the dynamic functionality is implemented in JavaScript. The only operational dependencies are the browser-level services, such as the *localStorage* persistence API [273].

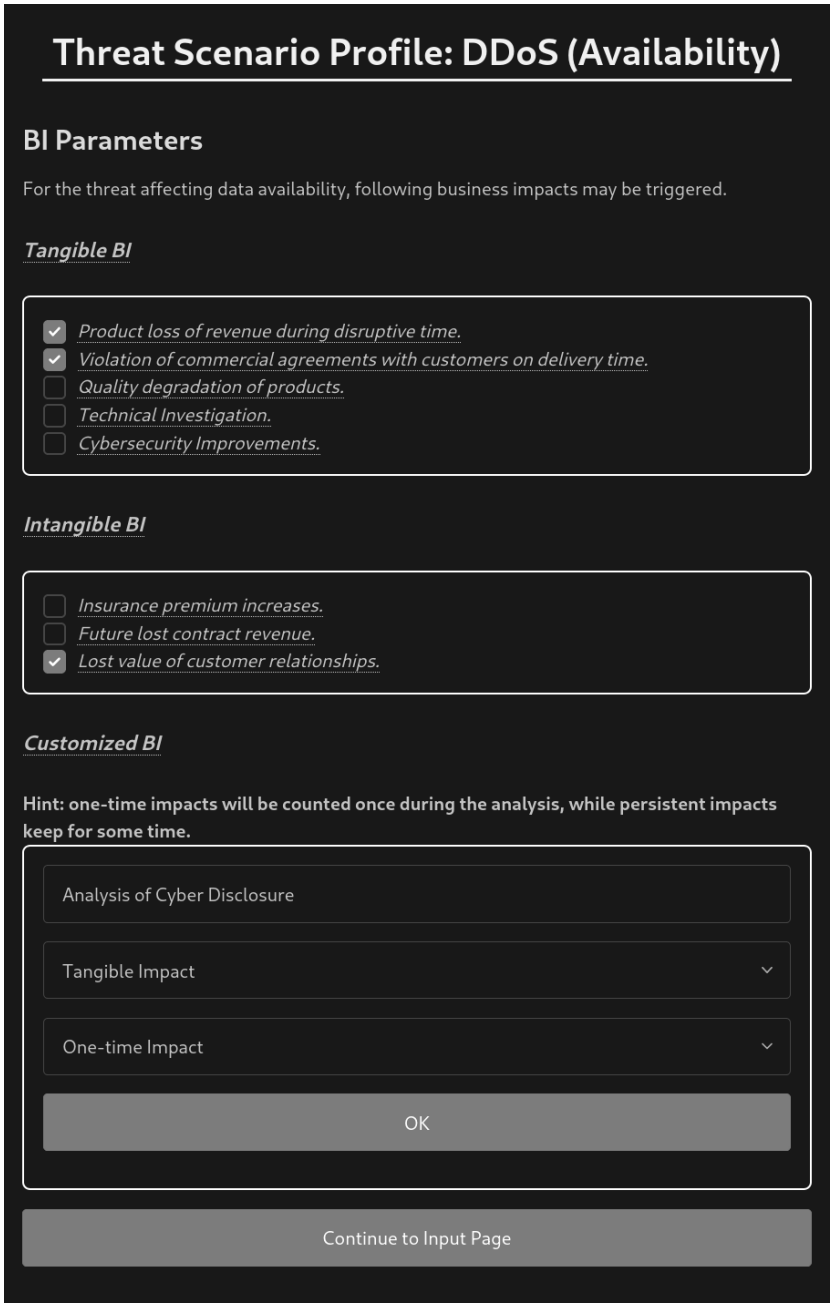

Figure 4.14: Defining Exemplary Impact Factors [273]

Users enter one or more threat scenarios on the *home page* to begin the BIA process. This page heuristically attempts to provide an initial mapping to the CIAA principles based on a predefined configuration. This output guides the selection of impact factors. The CIAA classification is performed by matching a threat description against a set of more than 20 commonly observed threats, which the user can extend. Consequently, tuples of threats and associated CIAA principles are stored

locally. The user is then redirected to sub-pages corresponding to each of the CIAA properties. On these pages, tangible and intangible factors are sourced from the library and proposed for each threat. These can then be activated or deactivated for each property, indicating whether the user wants to model the impact. Additionally, users can manually integrate other impact factors. For example, for the previously illustrated attack threatening the *data availability*, the following tangible (T) and intangible (I) impact factors are suggested [273].

- Product revenues loss during the duration of service disruption (T)
- Violation of commercial agreements with customers (T)
- Regulatory penalties (T)
- Quality degradation of products (T)
- Technical investigation cost (T)
- Defense improvements (incident response, post-mortem analysis, mitigation) (T)
- Insurance premium increase (I)
- Lost future contract revenue (I)
- Customer relationship degradation (I)

Each business impact is associated with either a one-time or persistent loss function. For instance, a regulatory penalty might be applied only once. In contrast, losses in product revenue likely depend on the duration of an outage and are thus modeled as a persistent loss. After collecting the applicable factors through the UI, which is exemplified in Figure 4.14, the *input page* guides the user in estimating the actual losses (*i.e.*, defining values for the proposed loss functions). For one-time impact factors, each factor is displayed, and losses are collected accordingly. For persistent factors, the recovery function can be formulated as pairs of recovery level and time along with the estimated loss magnitude. Users can also simply define tuples with extreme values to simplify the modeling. For example, the following formulates a complete service outage over four working days, avoiding fine-grained modeling [273].

$$(\text{recovery} = 0.0, \text{timeframe} = 4.0) \tag{4.6}$$

Once all input data (*i.e.*, impact factors and recovery functions) is collected, the model is applied, and the user is redirected to *home page*, where the overall impact is computed, and the resulting losses of each scenario are visualized. Bar charts are used to compare the impacts of different threats. The total loss for each threat is explained by breaking it down into its factors. First, pie charts display the relative distribution of tangible and intangible losses. The distribution of impacts is visualized for each threat to gain a deeper understanding of these factors. This visualization could be used to mitigate the impact of threats, thus enabling non-technical measures to supplement the mitigation of threats from a technical perspective. For instance, a hypothetical threat might cause significant losses due to a *violation of commercial agreements with customers*. If all technical measures have been exhausted but the residual risk of the threat scenario remains unacceptable, business owners could gauge the risk exposure and renegotiate the details of the commercial agreement to reflect the risk appetite. Finally, Figure 4.15 illustrates the continuity visualization of attacks. Here, line charts visualize different recovery levels by plotting the persistent losses over time for each threat. This shows how different impacts evolve over time, facilitating discussions on achieving a smoother recovery curve by reconsidering operational resilience [273].

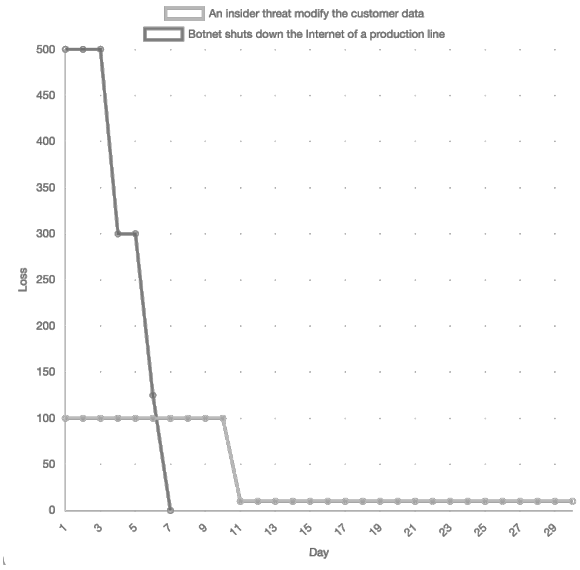

Figure 4.15: Visualizing Two Hypothetical Persistent Loss Scenarios [273]

4.3.3 Evaluations

Testing the contributions of *QuantTM* yields even greater complexity than the previous methods that targeted threat modeling: because threat assessment is almost exclusively practiced qualitatively, obtaining and testing the model against ground-truth data is infeasible. Thus, the following evaluations do not focus on *proving* that the output of the model is correct. Instead, they aim to illustrate how the methodology and model may exhibit *utility* for particular practical contexts. More specifically, it is being investigated whether *QuantTM* is *systematic*, whether it supports the inclusion of *multiple viewpoints*, whether it is *applicable* to real scenarios, and whether it is *usable* for real users. Furthermore, it is illustrated how the approach differs from related approaches. The following presents observational and analytical methods, including case studies, focus groups, and expert-based experiments.

Case Study

This case study considers a real-world Swiss SME transitioning from a manual order-by-phone process to an e-commerce solution. The main goal of the case study is not to produce a comprehensive list of threats and vulnerabilities but to deploy the approach to this case. The project's objective is to derive recommendations that *(i)* are relevant, *(ii)* comprehensible to the business, and *(iii)* present opportunities for actions and improvements. The e-commerce solution in question replaces an existing, yet not fully online and web-based information system: customers use electronic mail and telephone to place orders [273].

The company involved in the case study is characterized in Table 4.3. From a national perspective, this case is relevant as SMEs constitute 99.72% of the economy in Switzerland by the number of companies. Furthermore, they employ 66.90% of the workforce [238]. Unfortunately, even such small enterprises face cyberattacks due to particularities of their business (*e.g.*, lack of dedicated security staff, limited IT budgets) [84]. While the selected case involves a small company within the Swiss economy (SMEs typically have fewer than 250 employees), only 4.56% of Swiss SMEs have more than 20 employees, illustrating that the enterprise provides a relevant scenario from the Swiss perspective [238]. The company was contracted via e-mail. The company agreed to participate in the threat modeling workshop and to execute a business impact analysis. The first stage, focusing on the IS layer, was executed participatorily, where the authors introduced the method. Direct communication was not necessary after establishing the threat model. Consequently, interactions were conducted via e-mail to request the impact analysis, providing the essential details [273].

Table 4.3: Case Context [273]

Information	*Value*
Sector	Trade (B2B, Machinery)
Size	10–20 Employees (SME)
Country	Switzerland (Production, Logistics), USA, Spain
Information Systems	ERP, CRM, Storage & Domain Services

Phase 1 – IS Threat Modeling Initially, a threat model was developed based on the technical specifications of the information system in question. The e-commerce solution is developed in PHP and serves as standard software that can be customized for businesses while operating off-premises in a Swiss data center. For the analysis, it was modeled as a web-based application assuming a typical server-side rendering architecture. Hence, this stage followed the IS layer's high-level methodology while considering technology-specific attack libraries. Although the threat model is likely not exhaus-

tive, a foundational set of threats was selected for evaluation. On the IS layer, technical representatives identified five primary threats for the Web-based system under analysis. The first column in Table 4.4 summarizes the initial threat model. Then, probabilities for each threat event were established based on estimates and supplemented with historical data. For instance, *DDoS attacks* were predicted to occur every five years, with each attack being entirely successful (*i.e.*, $P_s = 1.0$) since there was no knowledge of DDoS protection services being implemented. These estimates were based on the service provider's management expertise and cross-referenced external reports [90]. As outlined in the Table, effects are still formulated in a technical language; for example, the DDoS attack leads to the unavailability of the application's outfacing interfaces [273].

Table 4.4: Information System Threat Model [273]

Threat Event	P_i	P_s	*Duration*	*Degree*	*Effects*
DDoS	0.2	1.0	48	1.0	Service unavailability (external)
CSRF	0.4	0.1	∞	1.0	Malicious input
XSS	0.4	0.5	∞	1.0	Malicious input
XXE	0.4	0.1	∞	1.0	Malicious input
Deserialization	0.2	0.2	72	1.0	Lateral attacks, file deletion, unavailability
Ransomware	0.6	0.2	72	1.0	Unavailability (int., ext.), data leak

Industry reports supported the estimation that 48 hours would be a pessimistic yet realistic duration for the DDoS threat event. Additionally, given the SME's operation of the information system, the event was characterized as a total loss of the system's service availability. The following three threats were assessed as residual threats with low initiation likelihood based on the service provider's and system representative's expertise. For example, *XML External Entity* threats were deemed to have a low success likelihood since XML was not used for external message representation. In contrast, an attack employing *Vulnerable deserialization* was considered more likely due to the potential existence of multiple sub-vectors [273].

Collaborative discussions and the technical team's insights on potential threat effects (at the technical level) were essential for this phase. A threat event leading to remote code execution was modeled as a series of derivative threat effects lasting 72 hours. Ransomware was also examined as a potential threat. In fact, the company had been previously breached by such an attack, leading to adverse effects even though the process was only partially digitalized at that stage. Up-to-date information on threat actors' behaviors is crucial for appropriately assessing the IS level. As described in Section 2.3.1, this threat now often involves data leakage and extortion as additional damage functions [273].

Technical threats pertinent to the information system and the analysts' assumptions were explicitly documented. Thus, they were associated with technical assets, including functional assets (*e.g.*, the shop's ability to execute services and employees' ability to process orders) and data assets (*e.g.*, confidentiality of orders and customer information) [273].

Table 4.5: Threat Model Interpretation [273]

Threat Event	*Duration (d)*	*Degree*	*Business Disruption*
DDoS	48	1.0	*(i)* Customers can not order online, only via phone
CSRF	∞	1.0	*(i)* Malicious orders to existing or new customers
XSS	∞	1.0	*(i)* Malicious orders to existing or new customers
XXE	∞	1.0	*(i)* Malicious orders to existing or new customers
Deserialization	72	1.0	*(i)* Employees cannot process existing orders *(ii)* Customers cannot order online, only via phone
Ransomware	72	1.0	*(i)* Employees cannot process existing orders *(ii)* Customers cannot order online, only via phone *(iii)* Existing orders are publicly accessible

Business Process Translation Building on the now-established descriptions of the threat model, threat effects were then translated and aggregated from a technical perspective to reflect their impact on the business process layer. Following the methodology, this step captured how the threats affect the business process without delving into technical specifics. This was achieved in a discussion involving close collaboration with a business representative and the system administrator from the SME. Table 4.5 collects the result of that discussion. Importantly, probabilities were not shared with the managers to avoid biases [273].

The key output from this stage is the identification of impacts on the business process. While some were modeled as tangible losses, several intangible losses were explicitly modeled for the SME. Interestingly, technical and non-technical personnel agreed on the outcome, although the technical personnel were unaware of the business' sensitivity to specific effects. For example, the SME generates a significant portion of its revenue from direct sales. Nevertheless, substantial revenue is sourced through other channels, such as consulting services and training. It was crucial to recognize that the business follows a Business-to-Business (B2B) model, characterized by long-term customer relationships and high trust. Additionally, the market segment is served by only a few players. Since the customers of the SME have a critical reliance on the products for their own operations, any interruptions lead to financial losses for those customers, too, thereby threatening customer retention [273].

Organizational Interpretation On the organizational level, the model was evaluated to assess the influence of technical threats affecting missions from the organization-wide perspective. A manager was tasked with assessing and quantifying the business impact stemming from the modeled disruptions. This impact was quantified using estimations (*e.g.*, anticipated customer reactions to trust breaches or service unavailability) and complemented with historical data. In this case study, estimates were represented by only recording the expected value instead of modeling distributions of potential outcomes [273].

Table 4.6: Modeled Business Impacts [273]

Threat Event	*Business Disruption*	*Attack Impact (USD)*
DDoS	*(i)* Customers can not order online, only via phone	1,620
CSRF	*(i)* Malicious orders to existing or new customers	0
XSS	*(i)* Malicious orders to existing or new customers	0
XXE	*(i)* Malicious orders to existing or new customers	0
Deserialization	*(i)* Employees cannot process orders	3,240
	(ii) Customers cannot order online	1,620
Ransomware	*(i)* Employees cannot process orders	3,240
	(ii) Customers cannot order online	1,620
	(iii) Orders are publicly accessible to both customers and competitors	1,080,000

Table 4.6 presents the modeled business impacts, which are based on estimated disruptions and their durations. All figures had been modeled in Swiss Francs (CHF) but translated to US Dollars (using the conversion rate from March 26, 2023: 1.08 USD for 1 CHF). The first threat event was modeled as a direct loss, yielding a financial impact of 1,620 USD. Here, the reasoning is that customers would resort to the existing sales channel (*i.e.*, via phone). Still, costs are modeled to restore functionality and conduct a basic investigation. The following three threats were not deemed to have a realistic impact when considering the business process, as the company initiates contact during the onboarding of new customers—the threat events associated with Deserialization first address the issue of orders remaining unprocessed. Here, the business assumes that in the worst-case scenario, the company would pay for expedited delivery fees to resolve customer complaints, estimated at 3,240 USD for the duration of the incident. Concerning ransomware, the risks of data leakage and extortion were perceived as significant. Here, understanding the particular business situation demonstrates why preserving this information could be advantageous when assessing threats: Each customer and even upstream competitors have customized purchasing conditions. In cases where that were made

public due to an extortion threat, this pricing policy would be questioned. In the best-case scenario, profit margins would merely decline due to the necessity of adjusting prices for certain customers. In the worst case, the company could lose its advantageous upstream procurement conditions to competitors, resulting in potential losses amounting to hundreds of thousands of US dollars. This threat poses a significant and, thus, strategic risk to the long-term operation of the enterprise. With this analysis, the *QuantTM* model quantified each attack through its impact on a technical behavior based on the conducted impact analysis. Table 4.7 displays the final aggregate impacts $Q(t)$ per threat, *i.e.*, aggregating the impact factors to the granularity level employed on the IS level. The values are based on the assumption that estimates are applicable for the specified threat duration [273]. Applying the formula constitutes the concluding step in the case study where *QuantTM* was deployed within the organization.

Table 4.7: Quantified Threat Model [273]

Threat Event	*Threat Duration*	$Q(t)$ (USD)
(Distributed) Denial-of-Service	48	324
CSRF	∞	0
XSS	∞	0
XXE	∞	0
Deserialization	72	194
Ransomware	72	13,543

Discussion Using the output obtained by applying the methodology and model, one can arguably inform downstream activities. In this case, the output could be interpreted for prioritization, indicating that three threats may be treated with low priority. This conclusion is drawn from the business-centric view established through *QuantTM*. The contributing factors to this assessment are less of a technical nature, but rather due to the presence of non-technical controls at the process level. For example, new customers are registered in a personalized process, making a purely "digital" attack complex. Two threats (*i.e.*, *DDoS* and *Deserialization*) are identified to result in business-significant consequences. Thus, one would likely discuss how to adopt feasible controls for which a quantitative approach facilitates the application of cybersecurity economic models. For instance, the ROSI could be utilized to contrast the reduction of risk exposure and control cost. Considering a hypothetical DDoS prevention service costing USD 540 annually, it may not optimize the overall financial outcome. However, an actual discussion would likely require assessing the effectiveness of the protection and the range of possible attack impacts (*e.g.*, does this service completely mitigate DDoS

attacks? Could there be attacks that are less likely but with higher impact?). In addition to discussing control efficiency, the estimates serve as a basis for prioritization, allowing threats to be ranked according to their impact or potential return on investment to identify the most significant threats and opportunities [273].

In this specific case, the application of *QuantTM* highlighted one threat (*i.e.*, *Ransomware*) being potentially existential and thus a strategic risk. For this, the variance between $Q(t)$ and the established attack impacts also illustrates that cyber-attacks are commonly viewed as long-tail risks. As such, organizations would need to base their decision-making on the expected value and the estimated impact of a rare event. Nevertheless, stating impacts using a quantitative approach facilitates various discussions. For instance, on a technical level, evaluating additional security controls leads to variants in system design. However, there may be other entry points to increase resilience. For example, business processes spanning different systems could be optimized while adopting business continuity methods. For this case study, one exemplary measure could involve implementing transparent pricing policies. While this would initially reduce profit margins, it could also enhance customer loyalty, decreasing the risk of customer losses. Strategic vulnerabilities can be discussed at the organizational level, as this threat illustrates the firm's dependence on one business function, potentially highlighting dependency on particular income streams [273].

To summarize, insights gained from instantiating *QuantTM* are interpreted. First, the adoption appears feasible even for small organizations with limited technical expertise or without prior knowledge of the adopted techniques (*e.g.*, BIA or BCM). In this case, *QuantTM* explicitly supported integrating diverse *information sources* [253], representing various organizational *viewpoints*. Based on this data, critical threats are identified based on business impact. It is argued that *QuantTM* follows a systematic methodology grounded in a clearly defined methodology and quantitative model, especially when comparing it to existing qualitative approaches focusing on modeling with externally defined factors or gut feeling. For example, the *Ransomware* threat was rated as having a high potential impact based on particular effects grounded and communicated in business-oriented arguments. In this case study, the magnitude of the threat's impact arose from the emerging data extortion subthreat. Here, a declarative factorization of threats' effects ensures that quantification is both explicit and repeatable. A notable observation arises when comparing outcomes with other methodologies. In a risk matrix – which will be experimentally tested in Section 4.3.4 – such an impactful but improbable threat might be qualitatively rated using the *medium* category. This opposes the outcome of the *QuantTM* approach, which described a threat scenario implying potentially catastrophic consequences [273].

Another observation highlights that following an explicit and cross-functional process yields findings for both technical and non-technical stakeholders. Specifically, during discussions of threat effects, the impact of data exfiltration was deemed as catastrophic – a business sensitivity that technical modelers were unaware of. This finding not only highlights that threats can be mitigated through technical and organizational controls but may also imply a lack of awareness on the technical level. However, this underlines the challenges of the approach: organizations must have the means to include these stakeholders, which may be challenging. Furthermore, non-commercial endeavors working on less tangible products (*e.g.*, open-source projects or e-Gov providers) may not hold data on cost factors. Furthermore, the limited generalizability of results obtained when evaluating artifacts using observational case studies must be restated. Nevertheless, the case study presents the first results of the solution's feasibility in *Small and Medium-sized Enterprises* (SMEs) [273].

Focus Group Experiment

The previous evaluation observed the usage of the methodology and model with limited experimental control against a real-world scenario. In the following, the objective is slightly shifted, focusing on the usability of the implemented prototype (*i.e.*, not only the methodology and model), which is experimentally tested involving a set of users. Thus, a higher degree of control is enacted to ensure the experimental setting. Thus, the experiment focused on executing a business impact analysis to assess the tool's effectiveness on an existing and thereby controlled threat model.

Methodology The focus group discussion consisted of a small group of five participants communicating online. Participant recruitment did not pose the requirement of having experience in threat analysis or business impact analysis. At the beginning of the session, an overview of the topic and approach was provided. This comprised an introduction to the solutions workflow and its prototypical design, a theoretical introduction to key concepts, and an outline of the experiment's scope. Finally, the specific tasks were assigned to participants, and a questionnaire – leveraging the *System Usability Scale* (SUS) and open feedback forms – was provided [273].

After introducing the solution, the discussion employed a hypothetical case study composed of a scenario description, task definitions, and data elaboration. Participants were tasked with conducting a threat assessment for a fictitious technology manufacturer. The stakeholders were advised to focus on three threat scenarios: malware affecting a central server, an insider threat altering customer data, and a botnet infecting devices involved in the production line. In addition to this technical information, business-related data of the company was provided. For each threat, participants were required to complete a series of tasks. First, they needed to translate and interpret the threat in relation to the process. Then, participants received access to the web-based prototype and applied it to identify

and quantify potentially relevant business impact factors. The following seven discussion questions assessed the effectiveness of navigating the prototype for the defined scenario [273]:

Q1) "When choosing the business impacts, do you understand what it evaluates?"

Q2) "Which threat costs the most to address, and which is most urgent?"

Q3) "At what time does the loss caused by the malware threat drop sharply?"

Q4) "Do you have an understanding of the proportion of tangible and intangible impacts?"

Q5) "Which threat costs the most to address? And which threat is most urgent?"

Q6) "Do you have a clear picture of the business impact composition caused by the insider threat?"

Q7) "How do participants rate the usability?"

In addition to guiding the discussion with these questions, feedback was openly collected, and the perceived usability was assessed using the SUS [273].

Execution Participants were recruited using a questionnaire assessing their background, education, and prior information security or BCM knowledge. Fourteen candidates registered, and five participants were selected for the experiment (*cf.,* Table 4.8). Only one participant with a finance background had encountered previous methodological exposure to BIA, and two participants held practical experience in application testing. All participants had theoretical education on information security [273].

The focus group experiment was executed as a 57-minute online discussion. Their screens were captured to assess the participants' performance interacting with the prototype. Tasks were completed sequentially: the methodology was described verbally, and no live demonstrations were provided to investigate if users could navigate the web-based tool. Then, the moderator presented the scenario and assigned tasks to the participants. Participants individually solved the tasks, relying solely

Table 4.8: Focus Group Participants [273]

#	*Background*	*BIA Knowledge*	*Testing Experience*	*Security Knowledge*
1	Master Student	No	Yes	Yes
2	Software Engineer	No	Yes	Yes
3	Master Student	No	No	Yes
4	Master Student	No	No	Yes
5	Finance Expert	Yes	No	Yes

on the supplied scenario data. They were encouraged to thoroughly analyze and utilize every component of the tool to gain a comprehensive understanding. During the execution, the moderator imitated the role of a business representative: questions or comments unrelated to the scenario were simply recorded. All participants completed their tasks within 35 minutes, followed by moderated questions to assess the application's effectiveness and usability [273].

Analysis Two sources of information were analyzed: *(i)* the collected data and screen captures, including notes and transcripts, and *(ii)* questionnaire results. The seven guiding questions were used to interpret the data. Participants produced consistent results for the six functional questions (*i.e.*, textitQ1-Q6). However, there was one task failure in response to *Q5*, where participants had to discuss threats considering both impact and urgency. One participant misinterpreted the graphs, yielding an inverted result – confusing the longest tolerable period with urgency. Nevertheless, the positive outcome for other participants and tasks suggests that the tool is effective in guiding the recognition of impact factors associated with technical threats. It is argued that this is applicable even when users have limited BCM knowledge. Here, the tool could guide the application and familiarization of the methodology, requiring users to have access to organizational data or build estimates. The results on *Q1* and *Q6* indicate that participants identified key impact factors influencing the risk exposure. Data on *Q2* through *Q5* illustrate that participants successfully analyzed, interpreted, factorized, and quantified threats for malware attacks, botnets, and insider threats [273].

The usability analysis, based on the SUS questionnaire, the moderator's notes, observations, participants' comments, and audio transcripts, are summarized by five observations. The first two focus on improvements regarding actionable user interface components (*e.g.*, buttons and input forms). Participants expressed confusion regarding the actions associated with particular elements, indicating a need for additional visual guidance within the tool. This need is supported by the following observations, where users explicitly suggested that HTML tooltips and labels could provide supplementary hints contextualizing terms and actions. Hence, this feedback is highly relevant if the tool is positioned to provide not just automation but methodological guidance. Despite some uncertainty, users demonstrated that they can effectively utilize the tool. Lastly, participants were unsure how the tool stores the provided data, which could be addressed through additional explanations on the *home page*. It is essential to acknowledge that users received minimal training and no design decisions were explained – although this privacy concern is addressed through client-side data storage [273].

Table 4.9: Usability Scores (SUS) [273]

Participant	*1*	*2*	*3*	*4*	*5*
Score	75	85	55	72.5	72.5

The computed SUS scores, presented in Table 4.9, were derived after normalizing the values. Four out of five participants rated the usability as sufficient, while one rated it as excellent, leading to an overall score qualified as "acceptable". The interpretation of the SUS score's influencing categories requires caution, as it is non-diagnostic: the application's learnability yielded "good" results, while the time required to utilize the tool highlights conflicting opinions, suggesting either a potential deficit or a mismatch between the participant group and the intended target audience [273].

4.3.4 Expert-based Experiments

So far, experiences from operating *QuantTM* were obtained from a real-world scenario and provided to real yet untrained users for a hypothetical case. In the following, it will be illustrated how the output of the approach may differ when experts with cybersecurity knowledge apply the *QuantTM* approach, contrasting it with two other commonly employed approaches.

Qualitative Matrix The first point of reference is established by applying the still commonly used risk matrix: a qualitative, two-dimensional table where both occurrence and impact are subjectively rated using abstract ordinally-scaled terms (*e.g.*, low, medium, high). This representation was then employed by a participant[2], who holds a Master of Science degree in Data Science and more than seven years of practical experience in the IT sector (*e.g.*, network engineering, data science, project management, security research). Thus, the participant had worked on cybersecurity-related matters in projects but never held a position focused exclusively on cybersecurity [273].

In the experiment, the participant received contextual information about the scenario gathered in the previously documented case study. The participant was presented with all the technical information and system documentation concerning the system – the resulting output of the *QuantTM* case study was removed to avoid influencing the reasoning. Subsequently, the participant was asked to prioritize the described threats from the model in Table 4.4. The prioritization shared by the expert was documented and illustrated in Figure 4.16. A comparison of this result and the output of applying *QuantTM* illustrates how the approaches differ. It is important to reiterate that due to limited data for this case, it is not appropriate to claim the correctness of either model [273].

[2]The expert involved in this experiment was **not** an author of *QuantTM*

Four notable differences are described. First, the qualitative analysis *(i)* indicates that all threats received the same *medium* score. While it cannot be proven to be false, it is arguably unlikely that all threats share the same priority. Secondly, assuming this score was correct, its utility would be diminished by the need for interpretation and granularity. For example, what would a *medium* score represent? How much worse would a *high* score be? Both DDoS and Ransomware threats are rated with a *high* severity. The results of the business impact analysis explicitly communicate a direct financial loss of a few thousand Swiss Francs – the other, including intangible consequences, leading to potentially significant long-term financial losses. Thirdly, closely related to the subjective and implicit formulation, this qualitative approach offers limited diagnostic value regarding its contributing factors and remediation. In contrast, the business impact factors utilized by *QuantTM* communicate weaknesses in strategic resilience. Based on the output of *QuantTM*, both technical and non-technical organizational measures (*e.g.*, how to decrease the impact of an attack? How to achieve continuity in case of an attack?) can be evaluated. Lastly, while professionals may prefer qualitative, subjective estimates for measuring probability due to data scarcity, this case study indicates that relevant data is available for specific threats. Hence, modeling using actual probabilities may be more expressive, even if some probabilities are estimated or derived from historical public data. Effectively, this fundamentally yields a different language of current estimates, which may not be "more correct" – but at least more explicit so that they do not require downstream interpretation that could lead to errors – for example, modeling the assumption that "with a 90% confidence, between zero and two large-scale DDoS attacks are expected per year" is more expressive than rating it as an attack with "medium likelihood" [273].

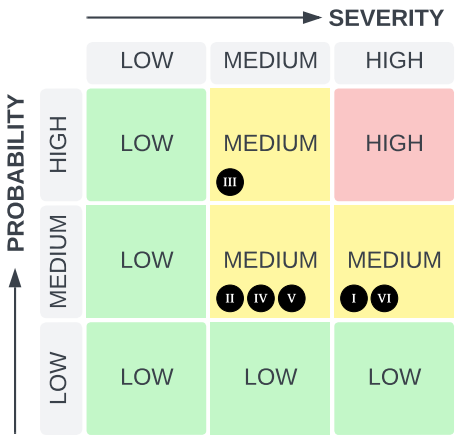

Figure 4.16: Qualitative Analysis [273]

DREAD In addition to using an unguided matrix-based approach, the expert was requested to prioritize threats using the still widely-established DREAD methodology and model (*cf.*, Section 3.1.6). DREAD differs from simple matrices since it provides a specific methodology to model a set of factors with a pre-defined scale. For instance, a data compromise would yield a damage score of 9, while completely destroying the system would produce a maximum score of 10. In that sense, it carries particular opinions: for example, a DDoS attack is to be modeled as more damaging than

Table 4.10: DREAD-based Analysis [273]

Threat	*D*	*R*	*E*	*A*	*D*	*Sum*	*Grade*
DDoS	10	10	10	10	8	48	C
CSRF	8	0-10	5	6	0-5	19-34	M-H
XSS	8	0-7.5	5	6	0-5	19-31.5	M-H
XXE	5	5	5	0-10	5	20-30	M-H
Deserial.	5	5	5	0-10	0-10	15-35	M-H
Ransomw.	10	5	2.5	10	0-10	27.5-37.5	M-H

L=Low, M=Medium, H=High, C=Critical

a data breach. Ultimately, scores are cumulated and summarized by four levels: low, medium, high, and critical [273].

During the experiment, the expert inquired about the methodology and then developed the scores for each threat. As indicated in Table 4.10, the expert encountered challenges in settling a specific score and recorded ranges of values – although this is not directly encouraged by the method. The final score was calculated by summing those scores, representing the lower and upper bounds. Similar to the risk matrix, most scores ended up with a similar rating. The exception here is the critically rated DDoS threat. Although a DDoS attack could arguably be critical, the results obtained from *QuantTM* argue that the organizational factors of the business would not justify this criticality (*i.e.*, the company claimed that in case of the sales system being unavailable, customers could and would likely order through other channels) [273].

Additionally, the categorization into the groups (*i.e.*, medium to high) questions the granularity of the scoring system, which may fail to convey the actual impact, questioning the actionability of the output. Focusing solely on the DDoS attack could be an obvious action – but aside from that, it is unclear how to proceed and which threats and mitigations to prioritize. Similar to the previous qualitative method, the resulting scores require interpretation. For example, the output of *QuantTM* questioned whether the economic impact of the DDoS threat warranted expensive countermeasures. The qualitative DREAD scores do not enable economic assessment without an intermediary interpretation step. In contrast, *QuantTM* offers specific business impact factors that encapsulate both tangible and intangible consequences against which threats and countermeasures can be discussed [273].

4.3.5 Key Observations and Discussion

Several observations are summarized regarding the design of the *QuantTM* approach, which embodies an overarching methodology (*i.e.*, a set of steps that yield a set of factors that are quantitatively

recorded), a quantitative model, and a specific prototypical implementation to assess threats in its business context by leveraging BIA. All observations are backed by practical tests (*i.e.*, observational, experimental, or illustrative methods) using either the high-level approach of *QuantTM* or its web-based implementation.

1. Employing *QuantTM* enables the assessment of threats using a quantitative "language". Statements are formulated precisely, explicitly, unambiguously, diagnostically, and without the need to interpret values.

2. Since the lack of data is a common counter-argument to quantitative threat modeling and risk management, the model and methodology are flexible to accommodate different data sources, enabling a "best data approach" – whether that data is based on estimates or historical records. Hence, even proponents of qualitative threat modeling could employ the model by "translating" their assumptions. As will be demonstrated when introducing the subsequent solution, subjective estimates may become critical for novel technological trends where historical data is unavailable due to their novelty. Other factors for a lack of data may include the staleness of historical data or the unwillingness of companies to reveal sensitive data. Since random decision-making is hardly an option, the relevance of the "best data approach" is emphasized, even if internal and potentially subjective sources are used.

3. The feasibility of using the underlying approach and the specific prototype in a threat modeling context has been demonstrated in small-scale deployments and experiments. However, not all factors of the model appear equally useful: modeling the occurrence of attacks raised fewer learnings than translating their impacts in consideration of the business context. From a conceptual perspective, it is argued that translating impacts from a technical viewpoint to a business-oriented one is the most fruitful contribution of *QuantTM*.

4. In practice, threat modeling is not the same as performing a risk assessment. It is argued that threat modeling comprises not just one "threat assessment" stage but that the mere fact of modeling assets and threats is intrinsically based on assessing and reducing uncertainty. Hence, even when using *QuantTM*, there are inevitable stages where experts use their intuitive feeling. For example, modelers working on a system architecture involving an OpenSSH server could reasonably add the control mechanism involving "disabling password-based access" without explicitly modeling the impact of "SSH brute forcing" on the business domain. Still, the discussion of residual and strategic threats may benefit from quantitative modeling.

5. These considerations are based on the study's underlying limitations: the small number and scale of experiments and the inability to prove the model's correctness quantitatively. Furthermore, the specific, practical limitations discovered during the experiments (*e.g.*, usability improvements, extending user guidance) are reported. During the discussion of *QuantTM*, it was argued that prioritization of attacks depends on the surrounding business and the objective of this prioritization (*e.g.*, discussing a control measure, establishing overall exposure for a strategic discussion). Hence, this approach requires access to those scenario-specific factors.

4.4 Modeling and Quantifying Threats to AI

This study shifts the focus to a particular technological perspective. Before, several solutions described how to model and quantify threats, focusing on technical and non-technical trends and challenges. The proposed solution, *ThreatFinderAI*, focuses on extending these two problems to the domain of AI-based systems. The motivation for this advancement is the scarcity of approaches and experiences navigating the threat modeling solution space for the AI domain. Additionally, the problem domain of integrating AI technologies into systems presents an interesting scenario due to its novelty (*e.g.*, data and knowledge on the prevalence and magnitude of attacks are still forming). Thus, *ThreatFinderAI* explores the viability of viewpoints from previous solutions (*e.g.*, providing expert-based guidance, leveraging visual collaborative modeling, employing a quantitative assessment that does not exclude subjective estimates) and proposes a methodology and architecture for a prototypical solution that is then deployed within this problem domain.

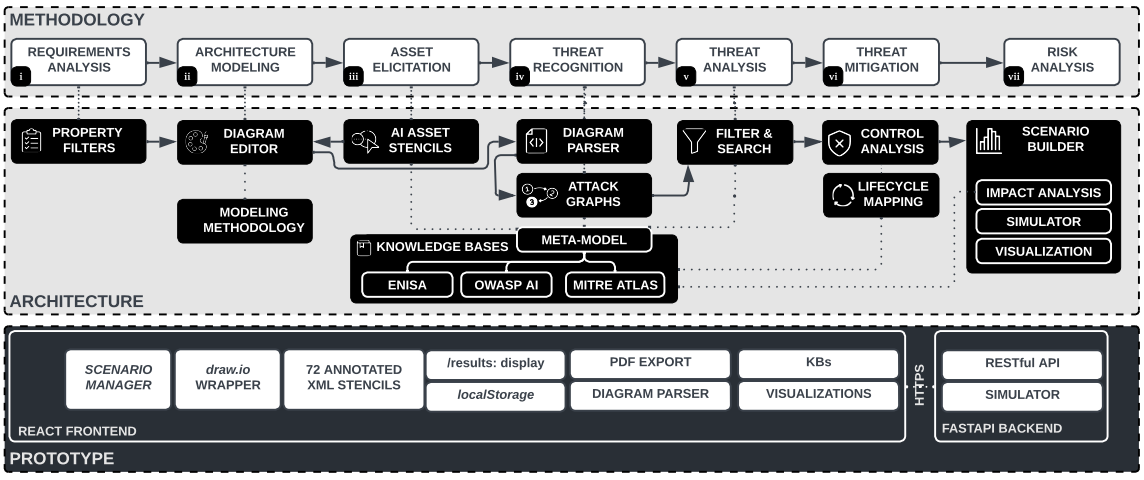

Figure 4.17: *ThreatFinderAI* Architecture and Prototype [269, 280]

4.4.1 The *ThreatFinderAI* Approach

The design of the *ThreatFinderAI* considers AI-based systems (*i.e.*, an information system integrating one or more technologies as defined in Section 2.3.2). It aligns the architectural semantics of these information systems with common procedures in threat modeling. The architecture of the *ThreatFinderAI* approach is illustrated in Figure 4.17. The architecture highlights how components relate to an abstract threat modeling procedure. These steps are aligned with established research in threat modeling, as described by the generic 5-step threat modeling process summarized in Table 4.11 [280].

Table 4.11: Generic 5-Step Threat Modeling Process [160, 179]

	Goal	***Description***
1	Objective Identification	Determine system security goals
2	Assessment	Identify system assets and interactions
3	Decomposition	Select relevant assets
4	Threat Identification	Categorize threats to assets
5	Identify Vulnerability	Analyze threats and determine vulnerability

To apply threat modeling to the AI domain, *ThreatFinderAI* explores the viability of previously investigated approaches (*e.g.*, visual modeling, asset-centric data extraction), methods from ERM, and *QuantTM* to extend threat identification towards strategic risk identification and communication. This integration yields a systematic approach that serves as a tool to test guided and quantitative threat modeling within AI security, addressing both academic and practical challenges. The seven stages that bound *ThreatFinderAI* are supported through an architectural framework comprising eleven macroscopical components, as detailed in the middle layer of Figure 4.17. A prototype was developed to assess the feasibility and effectiveness of the proposed strategy, as presented in the concluding section. The supporting elements of each step are described below [280].

Architecture

The first phase mandates defining an objective that guides the overall procedure. In *ThreatFinderAI*, an asset-centric viewpoint is suggested in combination with key properties, which had already been explored for insider threat modeling and threat quantification. Defining such an objective does not immediately bring value; however, besides providing relevant input for filtering and assessment, it can help ensure awareness of the system's business relevance. The step for object identification is framed through the scoping of a specific architecture aligned with a business mission, along with identifying a

key security requirement and an abstract architectural asset. For instance, users may outline a scenario such as "LLM-based medical data analysis," identifying "data" as the asset, and "confidentiality" as the key property. The approach leverages the CIAA principles and six abstract asset families synthesized from a public report [75]: actors, data, models, artifacts, processes, environments, and tooling [280].

After clarifying the objectives, the methodology requires analyzing the system from an architectural perspective by relying on system decomposition. Preserving the context for the threat modeling process is vital (*i.e.*, whether an architecture is newly developed or a risk assessment of an existing instance is created). The *ThreatFinderAI* methodology leverages visual asset modeling to achieve this step. In ideal cases, existing architectural models already exist from non-security-related activities. If this is not the case, it is assumed that formalizing the system visually aids the system understanding for involved participants. To achieve a holistic security perspective on the resulting AI-based system, related components must be explicitly modeled from different viewpoints (*i.e.*, looking beyond data and models). As one guiding knowledge base, the pipeline model of the AI life cycle from [75] serves to identify all activities and systems involved. Hence, users model any component, actor, and environment involved in any stage. To support and partially automate this process, the *ThreatFinderAI* tool leverages a diagram editor and a procedure that iteratively instructs users to model each life cycle stage. For instance, even when employing a pre-trained model, it is crucial that users model the data collection process. Unrelated to the actor and the transparency with which the step is fulfilled, it contributes to the system's attack surface [280].

Practically, a meta-model that abstracts resources detailing AI attacks is proposed to structurally and conceptually assist the diagram editor. Key assets, including processes, environments, data, and models, from [75] are formalized for architectural modeling. Consequently, each component in the diagram is classified by category (*cf.*, the previously introduced asset classes). For each elicited asset, a stencil is introduced to ensure that diagrams remain machine-processable. Thus, any system component in an architectural model could be extracted and related to the meta-model. As will be illustrated in the evaluations conducted on *ThreatFinderAI*, assets are formalized at a high level, making it viable for different flavors of AI. For example, an instance of the "training data" stencil could apply to ML, DL, or even LLMs [280].

Once the user concludes the system modeling, functional and data assets can be extracted while considering their relevance to the previously defined key asset families. For example, in a healthcare project where data confidentiality is critical, it is essential to identify threats to the training data. As mentioned in the previous stage, asset modeling was achieved using stencils. In addition to providing visual guidance, diagrams created using the stencils allow the recovery of annotations supplied by each stencil, as explored in the *CoReTM* solution. Thus, annotating each element with references

to the meta-model allows these diagrams to be analyzed automatically. This output is used for threat identification, which provides threat events related to extracted assets. For example, sourcing training data from an untrusted source may indicate vulnerability to a data poisoning attack. The system identifies threats using an overarching attack graph that combines multiple knowledge bases. In this context, attack libraries built from reports provided by ENISA, the OWASP AI Exchange, and MITRE ATLAS can be linked through properties such as asset category and life cycle stage to the meta-model. Thus, the meta-model synthesized from [75] enables the usage of various knowledge bases to query threats of the previous asset families [280].

A set of threats can be suggested by relying on abstract asset notations and knowledge bases. However, not all identified threats will be equally relevant (or even applicable) from the modeler's perspective. Therefore, the threat analysis requires user involvement to evaluate the suggestions, with the adopted threats forming the threat model. Users are guided to review threats related to the initially stated security objective. For instance, a data inference attack could be launched directly against the data or indirectly by accessing the model. Depending on the architecture, the relevance of each threat may differ. The next phase discusses countermeasures requiring the identification of technical, organizational, or strategic mitigations. Related knowledge bases can be queried to streamline this process, where the meta-model facilitates control identification based on the life cycle stage and asset family. For instance, monitoring the usage of a model serves as a countermeasure during the production stage, *i.e.*, it may not directly apply to data acquisition. This illustrates how the dimensions of the meta-model – in this instance, the life cycle stage – conceptually align threats, assets, countermeasures, and impacts [280].

Although practitioners draw the scope of a threat modeling activity differently, the current stage's output could already represent a viable threat model. Again, the scope, objective, and the participants involved must be considered. In a secure software design context, articulating strategic risks may not be desired, and participants mainly seek awareness of practical controls. Conversely, in a risk assessment context, the created model serves as a baseline to identify and discuss strategic risks [280].

As motivated by the practices in ERM and the findings of the *QuantTM* solution, the assessment stage follows a purely quantitative philosophy. As previously argued, data can be obtained from two sources: historical data and subjective expert-based estimates. AI security is a developing risk in information security – which is already considered an "emerging risk" [120]. Thus, *ThreatFinderAI* assists experts in quantifying, visualizing, and communicating the uncertainties associated with AI threats using estimates or historical data. To guide this activity, *ThreatFinderAI* identifies business impacts through a set of impact factors drawn from the literature and mapped to security properties that are

present in the meta-model. For example, training data breaches could result in notification penalties. The quantification of those factors established by BIA comprises the following three steps [280]:

1. Residual risk is characterized through risk scenarios. First, impact identification is performed for the threats in the scenario. *ThreatFinderAI* comprises tangible and intangible business impacts that are suggested based on key assets and the defined security principle. For example, a membership inference attack may lead to a data breach (*i.e.*, breaching *confidentiality*). Hence, tangible impacts, such as a data protection fine, or intangible impacts, such as loss of reputation, are suggested from the set of impacts.

2. Each business impact is quantified by building a distribution of financial impacts. Estimates can be developed by eliciting I_{min} and I_{max}, which frame the potential losses for a particular scenario. In addition, the analyst expresses the confidence level at which the scenario is expected (*i.e.*, defining I_c).

3. Estimating the number of occurrences is carried out in a simplified approach when compared to *QuantTM*. The best and worst-case number of occurrences (*i.e.*, o_{min}, o_{max}, and o_c) are stored in addition to the framing confidence level.

The identified business impacts and the associated estimates about occurrence and impact serve as the basis for quantifying scenarios. *ThreatFinderAI* applies Monte Carlo simulations by first building two distributions from the expert estimates: I_{min}, I_{max}, and I_c are used to build a log-normal distribution of financial losses. This distribution is widely employed to model operational losses in risk management, as it can model a potential "long-tail" area of the distribution (*e.g.*, incorporating that impacts can I_{max}) as introduced in Section 2.4.2. Modeling occurrences, on the other hand, hinges on the collected discrete values. The distribution of the scenario's occurrences is then modeled using a Poisson distribution characterized by o_{min}, o_{max}, and o_c [280].

After conducting several simulations (*i.e.*, drawing n values from the two distributions), a distribution of loss exposure can be obtained based on the two simulation results. Various metrics and visualizations to evaluate residual risk exposure are applicable. For example, optimistic (*i.e.*, losses exceeding a value with $p = 0.2$), pessimistic ($p = 0.8$), and expected loss ($p = 0.5$) scenarios can be computed. Figure 4.18 highlights how the probability density function of the losses and the loss exceedance curve can be graphically displayed [280].

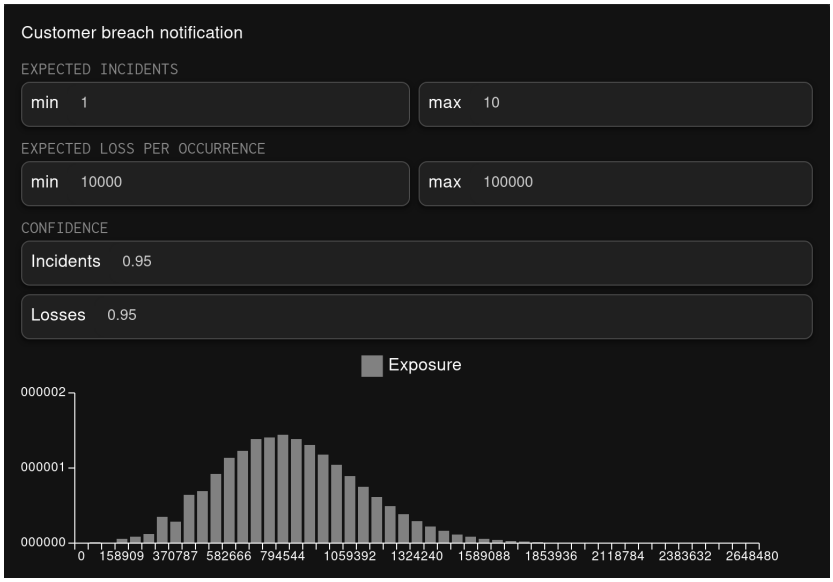

Figure 4.18: Quantifying the Business Impact on an AI System [269]

Prototype

A web-based solution was implemented to test the *ThreatFinderAI* approach, as summarized by the bottom layer in Figure 4.17. The graphical user interface was designed as a *Single-page Application* (SPA) leveraging the `React.js` library to simplify user interaction while providing a completely client-side implementation. Users initiate a new project by creating a description and providing a security objective (*i.e.*, defining asset family and security principle) on the `home` page. If the user already holds a diagram, it can be uploaded here. In the absence of an existing model, the `modeling` page provides an interactive diagram editor to draw it from scratch. The diagram editor is embedded within an inline HTML frame; the underlying drawing engine is supplied by the *diagrams.net* diagram editor [134]. The editor's functionality is extended by pre-loading stencil libraries and continuously storing the diagrams in the browser's storage via the `localStorage` API. This design enables architectural modeling without data leaving the browser and with minimal dependency on third-party interfaces. A custom set of XML-based stencils was developed to facilitate asset elicitation and threat identification. The resulting library includes one stencil for each asset elicited from the [75]. In total, generic stencils (*e.g.*, trust boundaries, actors) and AI-related artifacts are represented by 72 stencils. As depicted in the left modal of Figure 4.19, the stencils are categorized into six groups, facilitating the modeling of static software elements, processes, infrastructure, and actors [280].

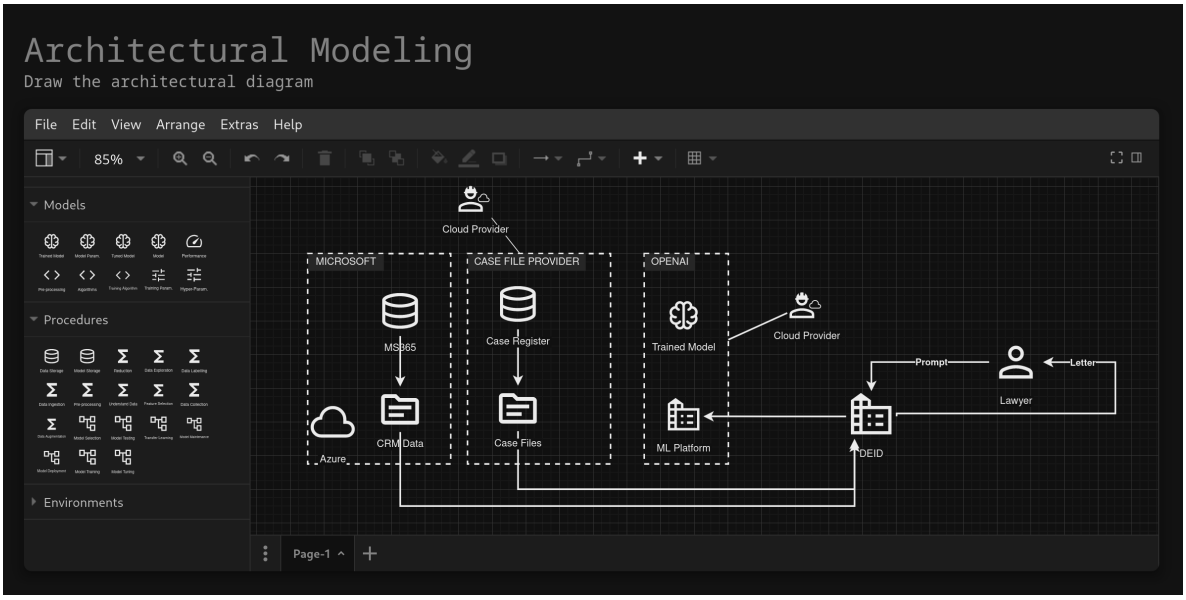

Figure 4.19: Architectural Modeling and Asset Annotation in *ThreatFinderAI* [269, 280]

Threats are then identified by parsing the diagrams, extracting assets based on their XML annotations, and querying one of three knowledge bases. Each knowledge base corresponds to a manually formalized version of the unstructured or semi-structured reports provided by ENISA, OWASP, and MITRE. The knowledge bases are formatted as a JSON file, linking threats through the asset-based meta-model that incorporates high-level steps and asset categories from [75]. This hierarchical structure allows users to explore threats per asset, assess their relevance, and incorporate them into a threat model. Once a threat model is created, the set of assets relating to the events serves as a basis for identifying controls by querying the knowledge bases. To support the strategic assessment of threats, users quantify residual risk (*i.e.*, the existence of the threat events, while considering the adopted controls). First, a knowledge base of business impacts is queried against the abstract effect of the threat (*e.g.*, which business impacts could relate to a loss of "availability"?). Users review the suggested impacts and then define the estimates for each impact, which are then quantified using simulations. To support the simulations described in the approach, data leaves the browser in the form of the six values framing the two distributions. Monte Carlo simulations are executed in the backend by building the distributions using readily available Python libraries (*e.g.*, numpy and scipy). The FastAPI-based backend executes $n = 100,000$ simulations and downsamples the resulting distribution of losses. Then, they are returned to the frontend, where they can be visualized, as demonstrated in Figure 4.18. Thus, at that stage, users have modeled AI-related assets, reviewed threats relating to them, adopted a set of control mechanisms, and quantified the residual impacts based on impacts [280].

4.4.2 Evaluations

Several evaluations have been performed during the iterative development of *ThreatFinderAI*. First, a controlled experiment using real users was performed – testing whether untrained users could leverage the prototype to recreate a threat model that experts initially built. This evaluation leveraged the first version of the prototype, which excluded quantification and did not fully automate the approach. Participants had to manually install the stencil library into a public (*i.e.*, not integrated) instance of *diagrams.net* and export the resulting diagrams into the *ThreatFinderAI* prototype for threat suggestion. The remaining evaluations represent the final version of the prototype, as described in the previous section, which will be subsequently illustrated through a demonstration.

Demonstration

The *ThreatFinderAI* approach and its prototype implementation were deployed into a production environment to demonstrate to the research community how it can be used for a realistic scenario [269]. The frontend is provisioned as a static web page to the *Cloudflare Pages* content delivery network, secured with strict HTTPS-only access. The backend merely provides the Monte Carlo simulations and is hosted in a Linux-based virtual machine. The machine operates within the University of Zürich's infrastructure behind an HTTPS reverse proxy and is equipped with 2 GiB of memory and two vCPUs [269]. Access to the publicly running instance is provided through [279], and a video demonstration of the following description is available in [278].

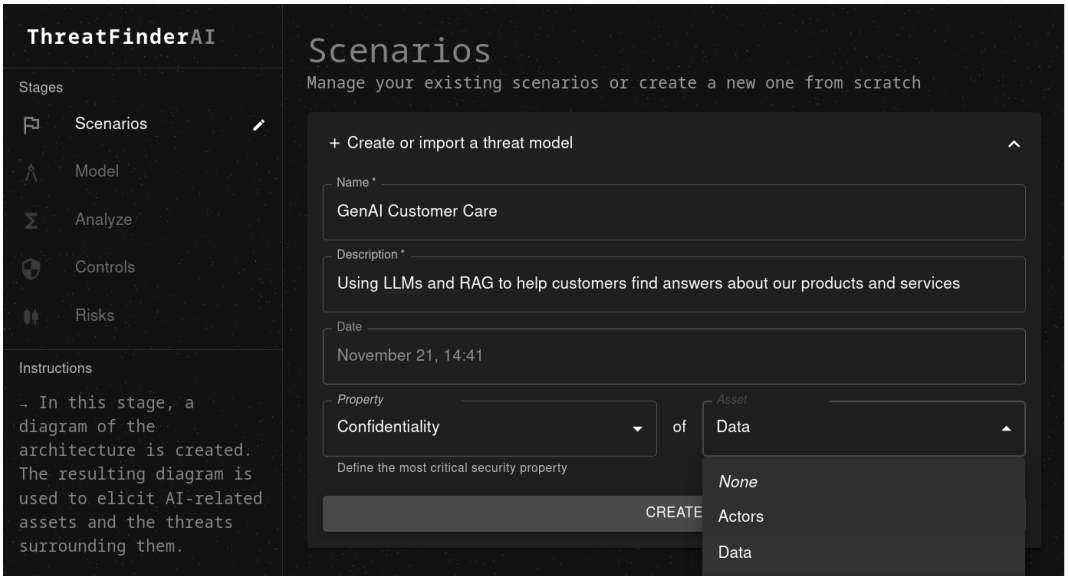

Figure 4.20: Demonstration: Scenario Definition and User Interface Overview [269]

To demonstrate the underlying approach, the presenter showcases the full scope of the platform through a scenario that resembles a system architecture developed by Swisscom, Switzerland's largest telecommunications provider, as part of their research on digitalizing customer care processes [77]. The synthesized demonstration scenario involves a fictitious organization that has historically utilized conversational agents to assist customers with inquiries related to products, services, and billing. Given the limitations of conversational agents and the growing interest in the usefulness of *Large Language Models* (LLMs), the organization is considering whether an LLM-based system architecture could optimize the relative number of customer requests whose initial request is resolved through an agent (*i.e.*, the resolution rate). A security analysis of the architecture employed in this scenario is conducted in the following demonstration. Using *ThreatFinderAI*, the demonstration illustrates *(i)* which threats exist for such an LLM integration, *(ii)* how the risk could be reduced through control measures, and *(iii)* whether the residual threat corresponds to a strategic risk for the endeavor [269].

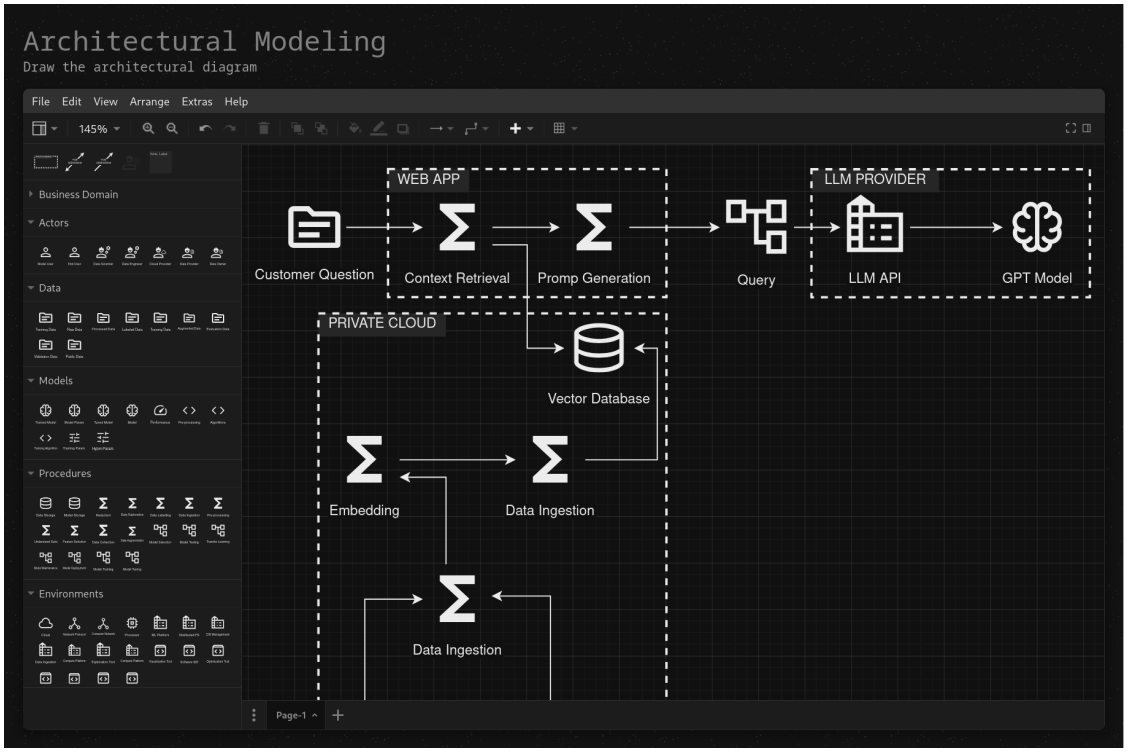

Figure 4.21: Demonstration: Modeling the Architecture [269]

Initially, the presenter establishes a new scenario based on key concerns of the problem description as shown in Figure 4.20. The project is described, and an initial security objective is identified to guide the threat identification process. Importantly, this represents an initial prioritization according to the business context and should not serve as an absolute filtering measure. The specific use case

does not relate to a critical mission (*e.g.*, sales, network management). Still, there are concerns about personal customer data held by the user and the company. Hence, *data* is identified as a key asset and *confidentiality* as the key principle [269].

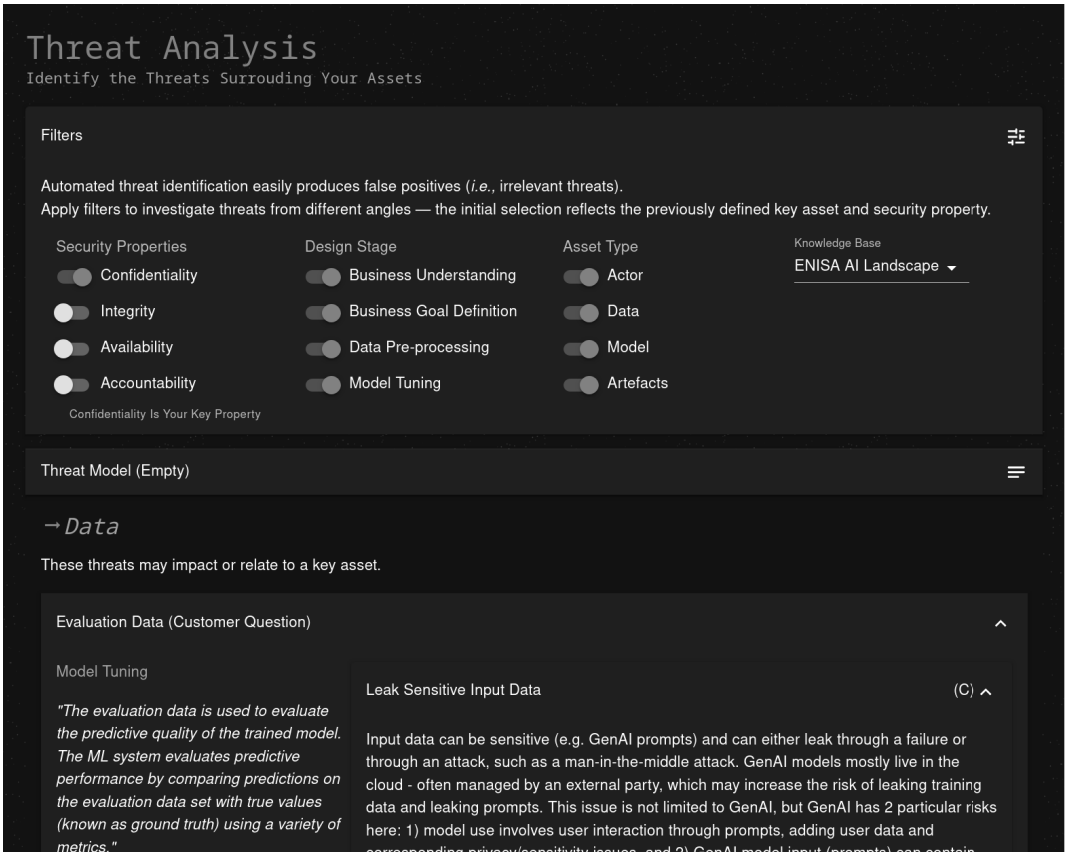

Figure 4.22: Demonstration: Identifying Threats [269]

Then, Figure 4.21 demonstrates how the architecture is manually modeled, assuming that a new diagram must be created within the platform. The integration architecture – mainly following a typical LLM-based *Retrieval Augmented Generation* (RAG) pattern – is modeled with three trust domains: the publicly accessible web application, the company's private cloud where information about products and services is stored, and the third-party LLM is modeled as an API-based ML platform. Relevant components are represented using dedicated stencils, including data assets (*e.g.*, customer requests in natural language, the embedded context retrieved from indexed knowledge bases), models (*i.e.*, the LLM operated as a service), and processes and infrastructures (*e.g.*, the procedure of splitting the text from existing knowledge bases, generating embeddings for the model, retrieving contexts from an embeddings database). The architectural pipeline is comprehensively modeled, beginning

with customer requests and detailing which internal aspects are used during development and operation. This leads to the resulting output being sent back to the customer [269].

After modeling the architecture, *ThreatFinderAI* automatically extracts assets by parsing the diagram. Based on the asset's representation in the meta-model, properties such as lifecycle stage and asset family are extracted (*cf.,* Figure 4.22). Incorporating the key asset definition, the platform suggests various threats by querying the knowledge base. For instance, *customer questions* are identified as a potential key asset since they relate to the previously prioritized asset family. The presenter illustrates the interactive stage of threat analysis, where the proposed threats are filtered based on criteria (*e.g.*, , prioritizing threats at the inference stage over the development stage). Different threat representations are compared across knowledge bases (*e.g.*, comparing threat descriptions in the ENISA report with the OWASP AI Exchange). Then, key threats, such as *leaking sensitive data through the prompt to the LLM provider*, are adopted and thereby added to the newly developed threat model [269].

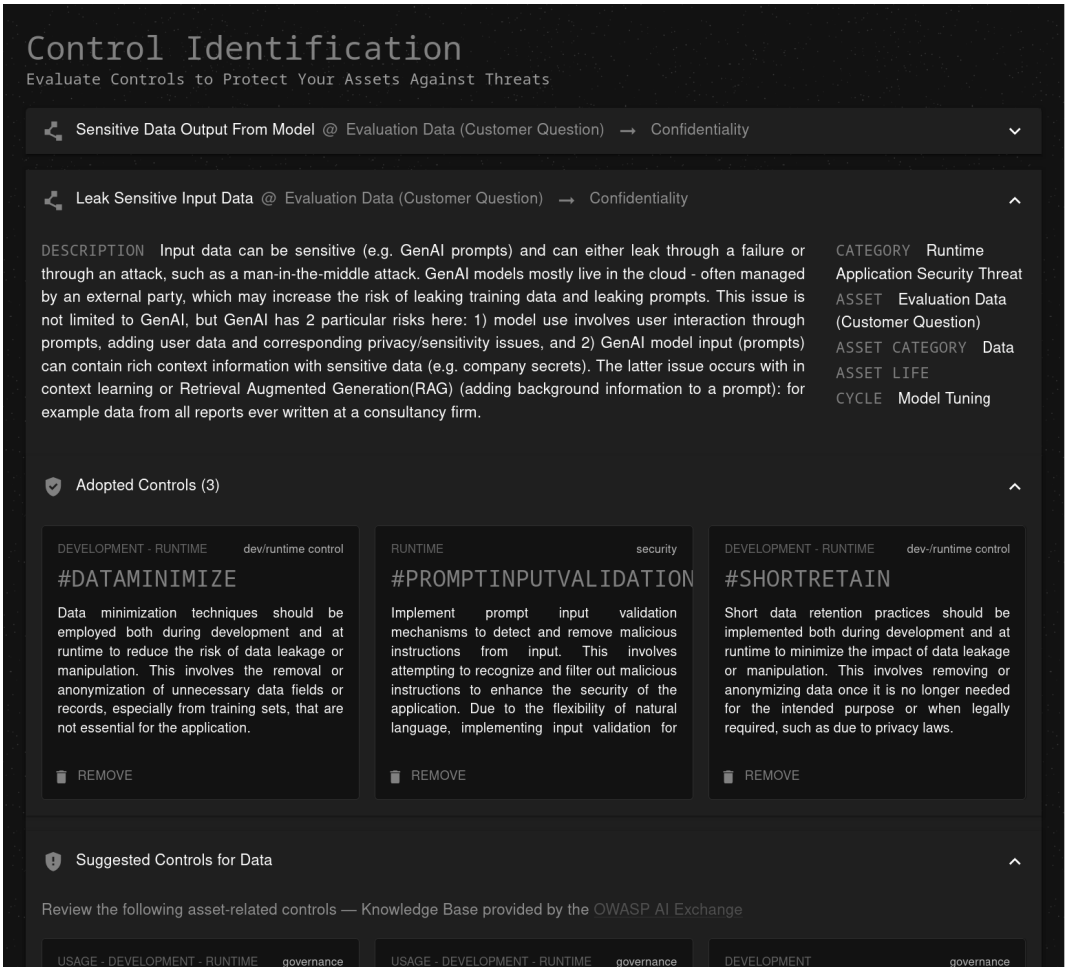

Figure 4.23: Demonstration: Identifying Controls [269]

To elicit control measures, the metadata associated with the threat model (*e.g.*, target asset family, lifecycle stage) is utilized to query the knowledge graph internally. Each suggested countermeasure is described through descriptive text and category, as shown in Figure 4.23. The demonstration illustrates various controls, including development-time and governance measures, highlighting the necessity of adopting multiple controls. The presenter then adopts several of the suggested controls (*e.g.*, *data minimization* and *anonymization*) to prevent scenarios where customers enter personal data into prompts since the prompt would be transmitted to the LLM provider. At this point, the technical part of the analysis highlighted actionable insights for architectural design [269].

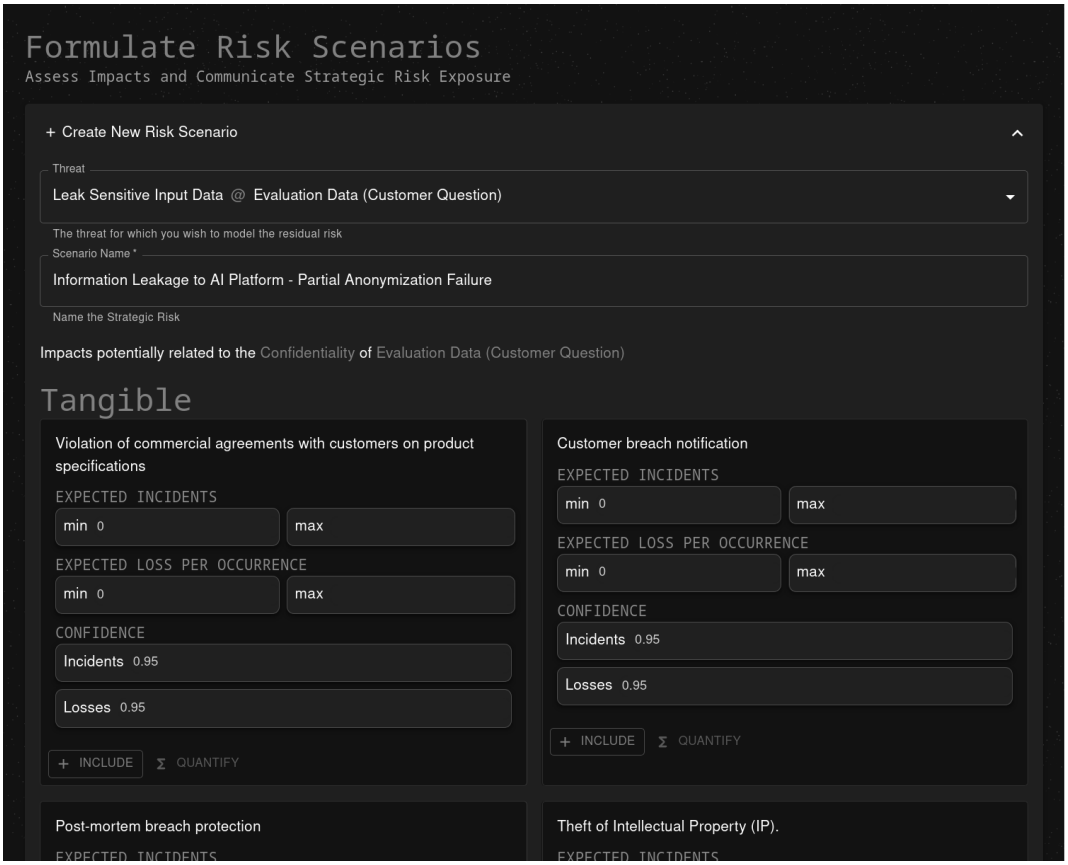

Figure 4.24: Demonstration: Identifying Impacts [269]

The presenter conducts a risk analysis using the threat model as input to extend this outcome and use it for a strategic discussion. Due to the confidentiality requirement, data minimization failures (*i.e.*, exposing personal data to the model provider) are considered an exemplary risk scenario. The presenter selects the threat in the user interface, prompting the tool to suggest potential business impacts (*cf.*, Figure 4.24), such as the *violation of data protection regulations*. To quantify the impact

of this event, minimum and maximum estimates for occurrences and losses are surveyed. Estimates are defined: between one and three incidents could occur over multiple years, where each incident would cause losses ranging from 10,000 to 100,000 USD. The visualizations derived from the simulations' results are indicated in Figure 4.18, illustrating the range of expected losses for this impact. By explicitly defining impacts, computing simulations, and visualizing them, *ThreatFinderAI* contextualizes threats' business sensitivity [269].

User Experiment

A similar experimental approach to the one used in the insider threat modeling study was used to test the ability of users to employ the tool. Therefore, this scenario-driven user experiment incorporates controlled elements (*i.e.*, an explicitly defined user task) and real-world aspects (*i.e.*, a baseline threat model and architecture of an AI-based system). Security and data science researchers manually created the threat model within a project that aimed to provide the basis for secure developments for an AI healthcare system (*cf.*, Section 3.1.4). For the experiment, the expert-created threat model serves as a "gold standard" model against which the output of the users can be compared. As in previous studies, it cannot be argued that the expert model is perfect (*i.e.*, all threats being relevant and no threats missing). The platform architecture used for the experiment emphasizes data privacy through architectural elements implementing anonymization techniques, attribute-based privacy measures, and reliable tracking systems [280]. Thus, data confidentiality and privacy are apparently highly relevant for business stakeholders. For example, a high-level misuse case threat model would highlight that data breaches (*i.e.*, a direct opposition to the platform's main mission of data sharing) are highly sensitive since they would not only lead to regulatory fines but to the reluctance of partners (*e.g.*, clinicians, hospitals, data brokers) to adopt the platform – diminishing its overall usefulness [280].

Execution First, security researchers created the baseline threat model. This team comprised two individuals with conventional cybersecurity expertise, one with practical knowledge in AI security, and a fourth with a mathematical background and data science experience. The experts used the web-based *diagrams.net* tooling, without any add-ons, *i.e.*, the authors resorted to a general-purpose diagramming solution. With this tool, basic shapes were used to represent key system components and their threats, actors, and trust areas. Ten trust domains were described, and relevant threats were identified. The experts relied on external threat information, which was manually gathered from academic and industry sources, including the ENISA report, the OWASP AI Exchange, and several academic publications. Each threat was associated with explicitly modeled threat actors (*e.g.*, malicious registered users, external human actors, automated bots, malware, and infrastructure administrators). The model included various security concerns, addressing threats during development, runtime, and

Table 4.12: Participant Overview [280]

Part.	*Educational Background*	*AI Knowledge*
1	Master of Data Science	Practical experience
2	Master of Data Science	Practical experience
3	B.Sc. Software Systems	Theoretical knowledge
4	B.Sc. Software Systems	Theoretical knowledge
5	B.Sc. Information Systems	Little to no understanding
6	M.Sc. Pharmacy	Little to no understanding
7	Master of Law	Little to no understanding

usage. The threat model listed 44 distinct threats – some affecting multiple components. In comparison, *ThreatFinderAI*'s knowledge base holds 96 distinct threats, covering and even extending and refining the ones found by experts. As such, it represents a superset, and participants could – in theory – find every threat using the data provided by *ThreatFinderAI* [280].

In the second phase, the actual experiment took place. Seven participants were recruited for a threat modeling workshop. Participants received a pre-recorded video tutorial conveying the usage of the tool. Furthermore, knowledge transfer about the existing architecture was provided by explaining the architectural model. In that sense, the inclusion of the participants resembled the onboarding of consultants to provide a security analysis once a basic architectural model is established. Arguably, this presents a pitfall for the experiment since the success of the experiment hinges on this domain knowledge transfer (*i.e.*, conveying the business case and the architecture). However, in a realistic scenario, *ThreatFinderAI* assumes that users can redraw an architecture in *ThreatFinderAI* is present, too, making it necessary to factor in this step [280].

Then, participants were tasked to apply the initial implementation of the tool – following the underlying methodology. Afterwards, the resulting threat models were collected by exporting them for analysis at a later stage. Additional qualitative data were collected by asking participants to complete a questionnaire, targeting their perceptions of the tool and its methodology. The participants' expertise and educational background were elicited at the beginning of the questionnaire, as indicated in Table 4.12. Although the experiment was based on a small number of participants, their academic and professional backgrounds differed. Notably, none reported cybersecurity expertise, and only two had an academic degree in Computer Science – majoring in Data Science. These two participants were the only ones who reported practical experience working with AI-based systems. The remaining users only had theoretical exposure [280].

Subsequently, the assumption made during the design of *ThreatFinderAI*was questioned: are IT professionals already familiar with the *diagrams.net* editor to avoid additional knowledge transfer? Based on responses on a Likert scale, all technical participants confirmed their familiarity with the tool. However, none of the participants reported awareness of threat modeling tools, highlighting that data science experts may have some exposure to tooling used in software engineering but not necessarily to security-related tools. Indeed, this finding is strongly limited due to the number of participants, but it could motivate future research investigating this observation [280].

The following questions aimed to elicit how participants perceived their ability while navigating *ThreatFinderAI*. All participants reported that they felt that they were navigating the tool successfully. The participant without a Computer Science background (*i.e.*, participant Six with a background in pharmacy) reported having experienced uncertainties when identifying assets and reporting limited comprehension of AI system architectures. When evaluating the clarity of the instructions, six out of seven participants expressed that they perceived them as understandable. All participants reported that their understanding of the architecture and scenario provided was solid. More specifically, three participants found it easy to comprehend, while the remaining four considered it sufficiently understandable [280].

Table 4.13: System Usability Score [280]

Part.	***Educational Background***	***Score***
1	Master of Data Science	55
2	Master of Data Science	70
3	Bachelor of Science Software Systems	85
4	Bachelor of Science Software Systems	52.5
5	Bachelor of Science Information Systems	52.5
6	Master of Science Pharmacy	75
7	Master of Law	45
Average		**62.14**

Finally, the usability of *ThreatFinderAI* was established using the SUS as summarized by the overall scores in Table 4.13. When the experiment was conducted, the average score indicated acceptable usability for the prototype. Among participants with a Computer Science background, Participant One's score was notably low, with individual responses varying across the ten questions. For instance, this participant expressed that he needed further assistance from a technical expert while stating that the tool was easy to learn. Additionally, Participant One voiced dissatisfaction with the underlying diagram editor while acknowledging the platform's positive features. As previously noted, the proto-

type version used in the experiment did not integrate the underlying system components – motivating this integration in the currently available *ThreatFinderAI* prototype [280].

Analysis Because it is possible to analyze the resulting threat models while grounding the output in the commonly used meta-model, it is argued that the proposed modeling process is guided and, in certain steps, automated. Due to this, the reasoning of the participants and the resulting threat models were compared to the initial model. Considering the security objectives formulated by the participants, the ones leveraged by the experts were congruent. The task success is assessed by comparing the baseline model and the models of the participants. Participants One, Two, Three, and Six identified all relevant threats in the baseline model. One potential explanation is that their previous practical knowledge of AI-based systems contributes positively to creating the threat model. Although the authors expected this, even one participant without practical knowledge achieved this and identified all relevant threats. In contrast, Participants Four, Five, and Seven did not exhaustively enumerate all threats, indicating that either those participants are not a realistic target group or that the *ThreatFinderAI* solution would require further improvements [280].

Hence, participants with a data science background demonstrated the ability to create effective models. However, concerning efficiency, it has to be emphasized that this result is the basis of both the tool's recommendations and the manual interpretation during threat analysis. As such, the approach does not fully automate the process. In addition to manual efforts, the tool suffers from the common issue of producing a considerable number of false positives – since the tool itself *suggests* many more than the 44 threats in the expert-based model. In summary, participants successfully identified all threats from the baseline model based on the provided architecture, attesting that the initial version of *ThreatFinderAI* can be used to systematically create AI-related threat models without requiring users to survey threats manually. However, efficiency can be enhanced by further refining the filtering of threats. The tool's usability at that stage was subjectively rated as "acceptable", motivating additional integration efforts of the *ThreatFinderAI* architecture – which serves as a basis for the previously provided demonstration and the subsequent case study [280].

Case Study

The second version of its prototypical implementation is employed in the case study conducted using *ThreatFinderAI*. Compared to the previous experiment, the following key additions were made. First, all architectural components are integrated into a single application – users do not need to use *diagrams.net* as a separate tool, install the custom stencils, create a model, export it, and import it to *ThreatFinderAI* for threat identification. Secondly, the strategic functions – identifying controls and related business impacts for a threat model and quantifying their impact – were added. As such, the

state of the prototype used for the case study represents the one illustrated through the demonstration provided in Section 4.4.2.

The case to be studied is a scenario where *ThreatFinderAI* is represented by an integration project in the legal field involving an LLM-based architecture. One of the services offered by the company frequently entails the creation of letters based on a case file, which is considered a highly repetitive yet necessary part of the process. Customer information, including the case files, is already stored in a cloud-based *Customer Relationship Management* (CRM) system. Additionally, the company can access legal information from a third-party service that commercially provides access to such information (here referred to as a legal research platform). With this information, the company envisions integrating an LLM-based architecture while being particularly interested in using an API-based LLM platform. No access to implementation details (*e.g.*, design decisions, exemplary data, system access) is provided, and the threat model and its subsequent risk analysis must be developed using that information. While this arguably represents only a fraction of available data, this is not necessarily an unrealistic scenario since not every security analysis has full information access. The theory on threat modeling would postulate this as one of the strengths of threat modeling: working with abstractions. By relying only on this information and feedback provided by the company, the following questions are addressed [280]:

Q1) Which threats are relevant?

Q2) Which countermeasures appear sensible?

Q3) Are the threats and their residual risk strategically sensitive?

Execution First, a new scenario is defined in *ThreatFinderAI*. Following the domain expert's opinions directly, *confidentiality* of *customer data* is defined as objective since other flaws could be mitigated procedurally. For example, if the content were influenced (*i.e.*, pertaining to *integrity*), manual reviews would likely catch them. If the system's *availability* was not ensured, letters could still be crafted manually. In the next step, an integration architecture diagram is created using the stencils. The resulting architecture encompasses two cloud platforms: one hosting the existing CRM and the other supporting a service-based LLM. Additionally, the third-party API facilitates access to legal information. Although the real architecture is not publicly available, it resembles the figurative one provided in Figure 4.19 [280].

After modeling the architecture, *ThreatFinderAI* extracts each asset automatically. The *OWASP AI Exchange* knowledge base was selected for threat identification. *ThreatFinderAI* elicited ten assets – four assets were associated with threats related to the security objective: *customer information*, *case*

files, *prompt input*, and *output*. The case files refer to data provided by the legal research platform; as such, their confidentiality is not assumed to be critical. *Customer information* is seen as the key asset to be protected and was further analyzed by investigating related threats. *ThreatFinderAI* identified six threats that could affect its *confidentiality*. Development-time threats were excluded using a life cycle filter, as customer data is only employed at inference. Development-time attacks would require closer attention if that data were used to retrain an LLM. For this asset, the following three threats were identified as highly relevant based on their description in the knowledge base [280]:

T1 Leakage sensitive data through the prompt.

T2 Inclusion of sensitive data in the output (*e.g.*, copyrighted text or customer information).

T3 Data exfiltration within the CRM.

All three threats are significant for a comprehensive discussion. Nevertheless, the third threat *T3* is already known since data is already stored in the CRM. Nonetheless, other threats, such as *T1*, highlight that the integration extends this attack surface (*i.e.*, data could then be exfiltrated from the CRM or LLM platform). Thus, further investigation focused on these newly formulated threats, especially since the goal was to assess the strategic risks of introducing these new technologies. To understand the risk exposure, countermeasures had to be discussed after adding *T1* and *T2* to the new threat model. Yet again, the knowledge base is queried by *ThreatFinderAI* using two contextual parameters. The stage in the MLOps life cycle (*i.e.*, inference, and not development-time) and the emphasis on data assets yielded 22 suggested control measures. The control measures summarized in Table 4.14 were included in the defense model [280].

Table 4.14: Identified Countermeasures [280]

Threat	***Countermeasures***
T1	Data Minimization, Anonymization or Deidentification
	Control Access to Customer Info
	Encrypting the Communication Channel
T2	Detect and Filter Sensitive Fields
	Output Monitoring and Retention
	Manual Review of Output

Data minimization, anonymization, and verifying the LLM's outputs are essential controls to reach the security objective. Thus, the application of *ThreatFinderAI* highlighted practical insights on how to

Table 4.15: Estimating Impact Factors [280]

Scenario	*Occurrence*	*Losses (CHF)*	*Confidence*
Data Protection Fine	$[0, 2]$	$[1E4, 1E5]$	0.9
Customer Losses	$[0, 5]$	$[1E5, 5E5]$	0.9
Legal Dispute	$[0, 1]$	$[2E5, 3E5]$	0.95

Occurrences Are Modeled Over a 10-Year Horizon, Losses are Defined Per Event

adapt the architecture: incorporating *(i)* anonymization of context data during prompt creation, 2) adding means to monitor input and output of the LLM service, and 3) including a procedural control where output is reviewed, before embedding it into the letter. Since some controls, in turn, use other systems (*e.g.*, monitoring systems, deidentification algorithms), which are unlikely to be perfect, there is a concern for residual risk exposure, represented by two scenarios. These scenarios were subjected to the BIA implemented in *ThreatFinderAI*. First, an anonymization failure would leak personal data. Hence, potential business impacts are approximated by scenarios such as degraded customer trust or the violation of regulations on consumer data. Two impacts proposed by the platform are analyzed: fines for data protection violations and customer attrition. The second scenario involves the output of the model: since the output is created using data provided by the legal research platform, the service provider could deem that this is an inappropriate use of its service. This scenario is analyzed through the impact stemming from a hypothetical legal dispute [280].

Table 4.16: Simulated Losses [280]

Scenario	*Optimistic*	*Expected*	*Pessimistic*
Data Protection Fine	69'162	139'647	233'629
Customer Losses	1'177'150	1'822'151	2'667'088
Legal Dispute	0	292'027	582'141

Based on loss exceedance with $P = 0.2$, $P = 0.5$, and $P = 0.8$

The business impacts of those two risk scenarios were quantified using simulations and subjective estimates of their occurrence, minimum and maximum losses, contextualized through confidence level. The applicability of Swiss privacy law, the potential for fines, and the option of adopting an insurance product were considered when formulating estimates. Table 4.15 summarizes the estimates. Internally, *ThreatFinderAI* computed exposures based on 100,000 Monte Carlo simulations and calculated the annualized losses, provided in Table 4.16. The estimates convey considerable loss variance.

When aggregating the overall risk exposure across scenarios, one could argue that due to the magnitude, the introduction of the technology corresponds to a strategic risk – although this would require knowledge of the company's risk appetite. Overall, the application of *ThreatFinderAI* to a realistic scenario highlights that it yields technical measures (*e.g.*, automated anonymization, monitoring) while also providing insights for non-technical actions. For example, the company could minimize its impact by reconsidering the use of technology. As such, the service could be offered transparently, where customers would have to explicitly agree on the technology's usage while continuing to provide the manual process in a dedicated product [280].

Analysis The initially formulated questions are discussed by studying *ThreatFinderAI* in the conveyed case. First, the approach systematically facilitated the identification of relevant threats through architectural modeling (relating to $Q1$). Naturally, these results must consider the fact that the completeness of the threat model cannot be ensured and that its creation depends on human interaction. Nevertheless, this case study illustrated that, although the methodology targets AI systems generically, the proposed meta-model applies to specific "flavors" of AI-based systems, including LLM platforms in this case [280].

Secondly, based on the input of domain experts and the usage of *ThreatFinderAI*, the contextual analysis of the threat model revealed that *ThreatFinderAI* can automatically propose relevant, practical countermeasures (*i.e.*, addressing $Q2$). For example, data minimization is identified as a potential control mechanism. While some identified threats are unique to the particular AI architecture, traditional control mechanisms also remain relevant, indicating that AI security encompasses "traditional" threats and controls [280].

Finally, *ThreatFinderAI* the case study demonstrated that *ThreatFinderAI* can be used to conduct a BIA and quantify the losses that occurred due to the impacts through simulations of estimated data. Although the accuracy of the estimates cannot be asserted, it is argued that *ThreatFinderAI* aids in identifying strategic risks. Specifically, the previously presented exposures would explicitly communicate that customer losses are a strategic risk scenario for the use case. Furthermore, experts can utilize the estimates and visualizations to articulate uncertainty and variance through the modeling with confidence levels and optimistic or pessimistic loss scenarios. Thus, these functions provided by *ThreatFinderAI* demonstrate that answers for $Q3$ can be formulated with precise, explicit, business-oriented arguments [280].

4.4.3 Key Observations and Discussion

The following observations are raised as a result of developing the *ThreatFinderAI* approach and its prototypical tooling. These observations carry findings for this chapter as a whole, too, since the

tool combines concepts from previously described solutions (*e.g.*, visual semi-automated modeling, business impact analysis, and quantification) and applies them to the AI security domain:

1. Basing threat modeling on visual architectural models presents the ability to automatically suggest relevant threats and practical countermeasures. Full automation appears questionable based on the subjective experiences of involved participants and the author's subjective experience. For example, in the case study, development-time threats were excluded because confidential data is only used at the inference stage, and the confidentiality of the model's training data was not critical. It is argued that this contextual information cannot be extracted from assets or data that would be available at that stage.

2. Although full automation appears not feasible with limited data – and as argued in the *CoReTM* study, creating and formalizing additional data could be counterproductive – *ThreatFinderAI* appears to be successful at guiding real users through the creation of a threat model. Still, the limited scale of experiments and the inherent challenge of proving the correctness of threat models weakens this claim.

3. The value addition from creating a threat model is not inherently given. *ThreatFinderAI* bridges this by directly tying this stage to two downstream activities: finding practical countermeasures and discussing residual risk.

4. For the communication of risk, the simplified approach of *QuantTM* involving BIA was tested for AI-based systems. Yet again, its correctness is not proven; however, desirable properties, such as the explicit and quantitative communication of risks that do not require interpretation, were illustrated. For example, the threat scenario formulated in the case study clearly communicates why a threat scenario is considered strategically dangerous through explicit measures and factors. Hence, not only system-oriented decision-making but also strategic discussions are supported.

5. Concerning threat modeling in AI security, additional observations are raised from practical experiences made in the field. First, a lack of data on actual attacks motivates the need for subjective estimates in quantification. Furthermore, experiences from deploying *ThreatFinderAI* question whether the stakeholders building such a system had exposure to security considerations. When leading application security discussions, builders of "traditional" information systems commonly have basic knowledge of information security. Future research could address and verify this knowledge gap.

To conclude this chapter, the author argues that proper tooling can help make threat modeling more systematic and, thereby, both more effective and efficient – to which the underlying functions (*e.g.*, guided visual modeling, semi-automated knowledge-based threat identification, impact analysis, quantification) contribute. Nevertheless, the studies in this chapter acknowledge the limitations of fully automating those aspects and testing them to prove correctness. Furthermore, the value of threat modeling is deemed to strongly depend on downstream actions: are controls actually implemented? If so, are they tested to mitigate the threat events properly?

The practice of cybersecurity is part engineering and part inference.

— Daniel E. Geer (2016)

5 Detecting and Mitigating Threats

Threat Modeling in itself does not reduce risk exposure. Instead, it presents an approach to navigate risk-reducing measures more effectively and efficiently. Nonetheless, concrete measures have to be taken to reduce the risk, an activity that can be guided by the threat model: vulnerable areas must either be removed or additional controls placed in front of those vulnerable areas. The following studies on mitigation focus on how to defend against attacks by building and testing approaches for *prevention* and *reaction*. As such, *detection* is employed merely as a means to an end since it does not cause risk reduction by itself. Thus, detection is investigated from a defensive perspective. Figure 5.1 contextualizes these solutions, using the familiar framing to highlight the challenges they address and the contributions they add to the solution space. It is argued that control mechanisms require closer proximity to a valid testing scenario to demonstrate their value. These solutions are therefore developed and evaluated in use-case-driven design research, making heavier use of quantitative and experimental research methods (*e.g.*, controlled experiments, simulations). First, a framework for lightweight MTD is developed for the threat model pertaining to the *ElectroSense* scenario. Based on those results, additional limitations are addressed by the following solutions. For example, *SecBox* investigates how the data used for the framework's reactive orchestration could be obtained more systematically and securely. *MTFS* aims to overcome the inefficiencies exhibited by the proactive MTD techniques.

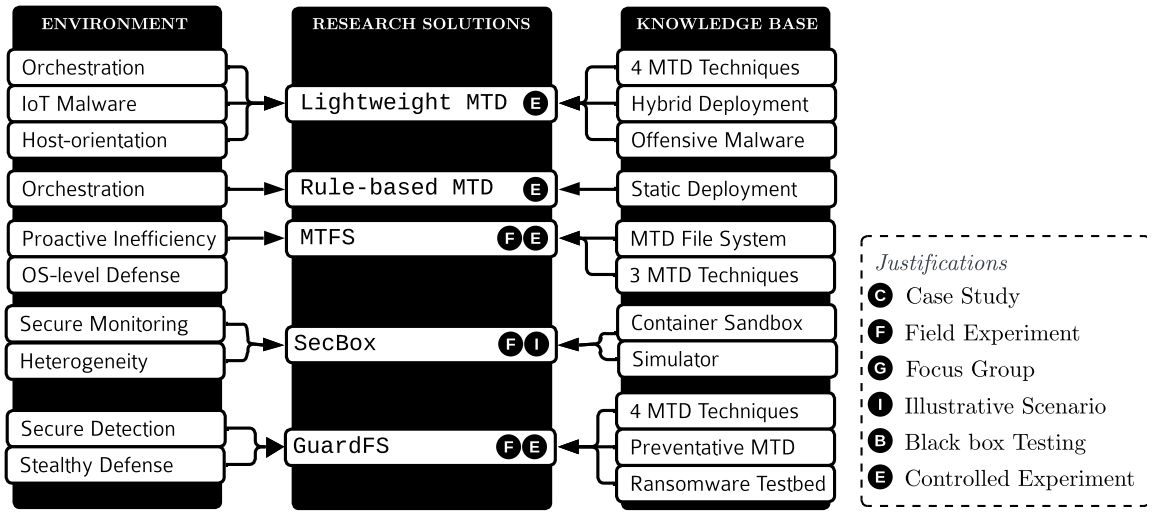

Figure 5.1: Solutions' Context for Studies on Detection and Mitigation

5.1 Lightweight Moving Target Defense

The first defensive design study is located in the IoT paradigm to elicit specific environmental challenges. More specifically, this study focuses on the *ElectroSense* scenario, described in Section 2.3.2. The threat model inherited in this scenario exemplifies and accentuates many of the security considerations commonly observed in IoT security. For example, although *ElectroSense* operates on rather powerful IoT hardware, computational resources are still a concern. Moreover, the defense of this scenario is not only bound by technology: due to the operation of crowdsensing nodes in residential networks, defense is also limited from a management perspective. For example, although accurate detection could raise alerts to the operator, it is unlikely that an *ElectroSense* operator is inclined to spend time investigating security alerts raised by the device. The application of MTD for this scenario also accentuates the limitations found in the solution space: few MTD frameworks operate autonomously on the device while considering the proactive and reactive orchestration of defense mechanisms against multiple attacks (*i.e.*, limitations *L6* and *L7*). By developing a novel framework and testing it in *ElectroSense*, this study contributes to the overall lack of experiences stemming from practical deployments and tests. In the following, the MTD framework is introduced, which comprises *(i)* several concrete MTD mechanisms countering common malware-based threats, *(ii)* a reactive and proactive orchestration approach, and *(iii)* the evaluations performed against real-world malware on *ElectroSense*.

5.1.1 Framework Architecture

Figure 5.2 highlights the components of the proposed MTD framework. The key objective of the framework is to hold a set of MTD techniques that can be executed locally (*i.e.*, without interacting with a third-party service). It is assumed that those techniques mitigate a broadly formulated behavior (*i.e.*, a high-level threat). Hence, the framework's techniques could be mapped to the threats in a threat model. For example, crypto-ransomware would require the development of a dedicated technique that can counter the malicious behavior [267].

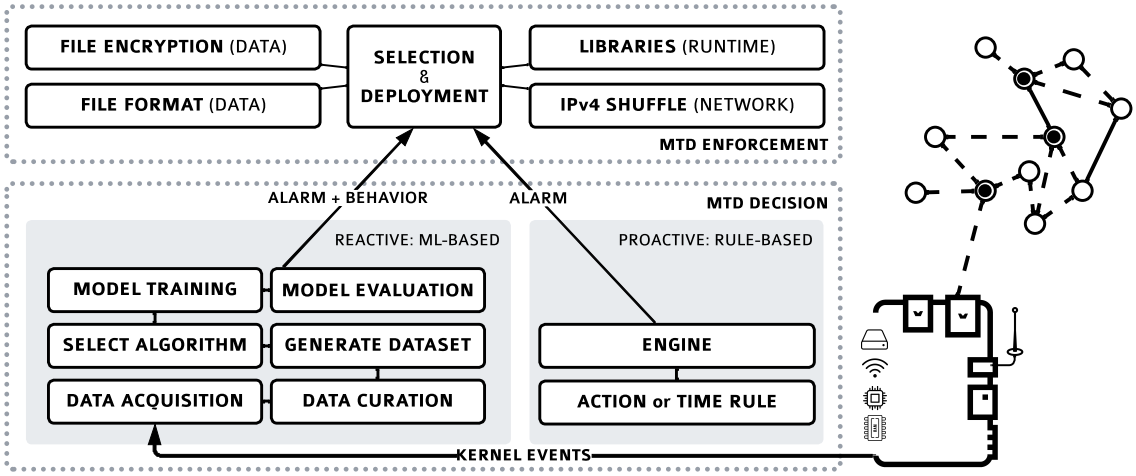

Figure 5.2: Framework Architecture for Lightweight, Host-based MTD [267]

Based on those assumptions, the framework is composed of two macroscopical areas. The *decisioning module* decides not only when to start the defense but also which of the locally available techniques to deploy. The other area deals with the actual *enforcement* and abstracts technical details of the defense [267].

To implement the *decisioning module*, the fundamental deployment approaches in MTD apply: proactive deployment relies on an initial start definition, whereas reactive MTD uses a runtime signal. As such, decisions are taken before the system is deployed (*i.e.*, how the system should behave at a later point in time, without considering its environment). In the proposed framework, proactive deployment requires that the operator define either a playbook of actions or a time-based rule upon which techniques are deployed. The rule-based engine then evaluates those configurations at runtime and interfaces with the enforcement layer [267].

The reactive approach differs here since it leverages environmental signals and moves the decision-making to the runtime of the framework. Signals obtained from inferencing supervised ML models

trained on the device's behavior could provide accurate signals about the system's state. Since the framework is designed to be completely autonomous, dynamic features of the device are obtained by the *Data Acquisition* component, which monitors OS-level parameters (*e.g.*, resource consumption, kernel events). Similarly, the components denoting *Data Curation*, *Dataset Generation*, *Algorithm Selection*, and *Model Training* implement a common ML pipeline based on those behavioral features. The *Model Evaluation* component should be deployed on the device to ensure autonomy – the interim stages where the model is built could be operated offline. Ultimately, this module interacts with the enforcement layer, much like the proactive module [267].

The enforcement module is controlled by a centralized selection and deployment component. Since the proactive module does not present additional information, a default behavior, such as the deployment of all parallelizable techniques, must be defined. Alternatively, if it is informed by the reactive decisioning module, it can take a decision based on both time and the detected malicious behavior (*i.e.*, the classification result *clf*). In the following, a concrete set of MTD techniques countering common malware-based threats is introduced. Those techniques are registered by the selection component with a simple rule-based configuration. A simple mapping composed of six rules is used, and no information flows from the techniques (*e.g.*, defense success, errors) to the enforcement layer [267].

R1) If [*clf* == Rootkit] → deploy [Libraries MTD]

R2) If [*clf* == Ransomware] → deploy [File Encryption MTD]

R3) If [*clf* == Botnet] → deploy [IP Address MTD]

R4) If [*clf* == Backdoor] → deploy [File Format MTD]

R5) If [*clf* == action] or [time] → deploy [File Format MTD] and [IP Address MTD] and [Libraries MTD]

Defense Techniques

Before instantiating and deploying the overall framework to the *ElectroSense* scenario, four conceptual MTD techniques are developed in accordance with the threat model and the available defense capabilities. The techniques are designed for a scenario with constrained resources. However, they mainly depend on a Linux-based operating system and could likely be ported to other environments.

File System Honeypot This technique broadly encompasses malicious encryption of sensitive files, as may be the case with ransomware samples. Related studies have attempted to mitigate

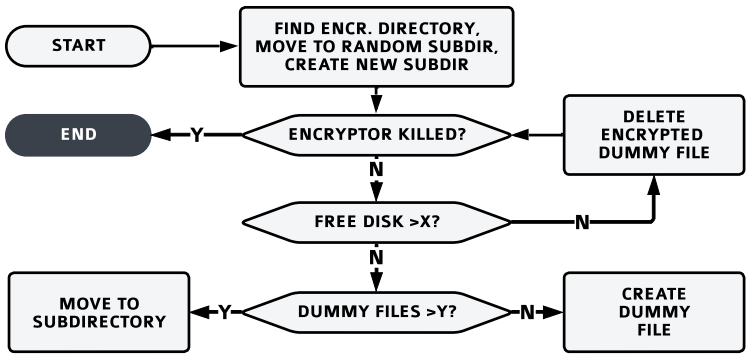

Figure 5.3: File Encryption Defense [267]

such samples at other steps in the CKC (*e.g.*, at key creation, during C&C orchestration); this technique assumes such efforts have failed in a previous line of defense. The technique aims to minimize data loss under the assumption that a breach has already occurred. Conceptually, the technique exploits the behavior assumed to be involved in file encryption: recursive file identification and its subsequent encryption. Since the adversary's goal is to provide damage by making system assets unavailable through encryption, a contradicting objective is followed: executing encryption while ensuring the system remains usable to a certain degree (*e.g.*, complete the encryption, inform the user about the ransom). Thus, samples discover files in a specified directory, while potentially targeting only a subset (*e.g.*, encryption PDF files, ignoring linked libraries). Then, the same behavior is executed recursively by moving to the sub-directories and applying the same behavior [267].

To diminish the success of this behavior, the technique, visualized in Figure 5.3, creates a set of dummy files distributed across directories. The technique uses those files to trap an executing ransomware sample by creating and destroying such files dynamically. By "trapping" the sample's execution, file loss could be minimized while gaining additional information. From the deployer's perspective, the proposed MTD is first deployed in the directory where the ransomware is executing. The technique then relocates to a randomly chosen subdirectory, creating a new subdirectory filled with dummy files. Once a predetermined number of dummy files is created in each directory, the MTD transitions to another subdirectory and reiterates the process. Although the files do not contain actual user data, they must be cleaned regularly to avoid wasting disk space. In addition to maintaining the honeypot files, which can only delay or distract the ransomware, the technique involves monitoring the processes. By doing so, the process executing within the honeypot can be terminated, halting the

encryption process. It does so by analyzing compute-intensive tasks, excluding whitelisted processes, and tracking the number of files held by each process [267].

File Suffix Obfuscation As previously illustrated using the case of ransomware, malicious behavior may encompass only certain file types. This may likely be the case for other malware families (*e.g.*, data exfiltration, rookits), too. Thus, this technique aims to exploit the adversarial vulnerability of relying on file identification through its file path suffix [267].

Combatting malware relying on this vulnerability involves generating pseudo file suffixes, where alphanumeric strings are generated and appended, concealing the real file type. This implies that user data can be recovered later, necessitating state management to map modified file names in a dictionary. When a new pseudo extension is generated, the technique ensures that it leads to a unique file name, preventing its reconstruction. Naturally, this technique cannot terminate malicious execution, assuming another technique can do so while restoring the original state. At runtime, three parameters are needed, since it is unfeasible to apply the technique to the full system. The technique accepts the target directory, a flag indicating whether files in the hierarchy are included, and a set of file extensions to obfuscate [267].

Dynamic Linking This defense method is more specific to the selected execution environment, actively manipulating the Linux runtime to reduce malicious alterations of system libraries. To prevent attacks involving code execution, this technique sanitizes corrupted dynamically linked system libraries. The design of the technique targets a specific property in the Linux runtime environment, where libraries can be preloaded by specifying an environment variable (*i.e.*, `LD_PRELOAD`). By setting `LD_PRELOAD`, dynamic libraries are linked from user space, overriding kernel-space libraries. Rootkits commonly exploit this to transparently preload malicious libraries through the variable. Additionally, rootkits can unlink and replace the `/etc/ld.so.preload` configuration. Since writing to `/etc` commonly requires elevated privileges, the adversary is already assumed to have obtained such privileges [267]. To defend against the behavior, the proposed technique applies sanitization, thereby restoring both the `ld.so.preload` specification in the configuration directory and by restoring and linking a previous `LD_PRELOAD` value [267].

Private IPv4 Address Shuffling The fourth technique adopts a network perspective, as done by other studies in the MTD field. However, due to the necessity of autonomously defending (*i.e.*, being unable to rely on Software-Defined Networks or Firewalls), the approach is modified. Incorporating a network component is potentially useful since C&C interaction is common to various malware families, including cryptojackers and botnets. A key distinction from related studies is the operation with private IP addresses, as is commonly the case when operating in residential networks, where *Network Address Translation* (NAT) is highly common. The main goal of the defense method

is to disrupt connections with C&C servers, irrespective of their location (*i.e.*, on the device, in the local network, or externally) [267].

More specifically, the mechanism compiles a list of all unused private IPv4 addresses within the local area network of the node. An initial set of addresses is derived from the local subnet configuration. Using existing network discovery approaches, the addresses of active devices are identified on the data-link layer by scanning using the *Address Resolution Protocol* (ARP). Following this, a random and unused IPv4 address is selected, migrating the network configuration to the new source address. The technique then verifies network access, either restarting legitimate processes or reverting to another address until network reachability is given [267].

5.1.2 Evaluations

In the following experiments, a prototypical implementation of the framework is implemented and deployed to the *ElectroSense* platform. Then, both modules are tested, therefore providing details on the attack detection and mitigation performance.

In the *ElectroSense* platform, several IoT devices can be used. For the experiments, the *ElectroSense* sensor leveraged a Raspberry Pi 4 (1.5 GHz CPU, 3.7 GB memory). The device executes the official sensing workload and is connected to a residential LAN. In the following, the initial breaching stages needed by malware are omitted – simulating a scenario where a previous line of defense (*e.g.*, SSH authentication, web-based authorization) failed. To test the MTD against several damaging payloads, the operating system is directly infected with malware samples of four families. This included botnets (*bashlite*), ransomware (*Ransomware_PoC*), rootkits (*Beurk*, *Bdvil*), and backdoors (*TheTick*, *PythonBackdoor*, and *httpBackdoor*). Thus, several malicious behaviors (*e.g.*, remote orchestration, data destruction) were observed after infection. The malware samples are available through [47]. In addition to those malicious payloads, the *ElectroSense* workload was constantly operated [267].

Reactivity

The ML-based detection module of the framework starts the detection approach outlined in [47]. These experiments do not necessarily contribute new knowledge to the solution space, *i.e.*, this study does not aim to answer how to effectively detect attacks. Nevertheless, to assess the performance of the reactive defense as a whole, the detection performance must be established. Due to the particular threat model, autonomous defense requires that the *Data Acquisition* component leverages host-based sources. In the deployment, kernel events, which are inherent to any Linux-based operating system, have been leveraged. Monitoring was achieved using common CLI tooling, leveraging the

`perf` utility. This tooling enabled the extraction of approximately 80 metrics and, as such, presents information about the device's behavior from different perspectives (*e.g.*, networking, memory, file system access, CPU, process scheduling, input and output devices). To balance the monitoring overhead and data granularity, every 5 seconds, a snapshot of those dimensions was gathered, with additional time required for processing. This implies that some amount of computational resources is spent to collect and process the data to have a more accurate view of the system compared to a proactive deployment. Table 5.1 highlights the computational overhead incurred when monitoring these time windows, which, including processing, are obtained with a total duration of ≈10 seconds [267].

Table 5.1: Monitoring Efficiency [267]

Monitoring Time	***Processing Time***	***CPU Usage***	***Memory Usage***	***Storage Usage***
5 s	≈5.21 s	1-4 %	6.14 MB	6.98 kB

To develop an initial model that can distinguish between normal and malicious behavior, both of those behaviors must be recorded in order to leverage supervised learning methods. Thus, the different configurations (*i.e.*, the normal behavior, and each of the malware samples) were operated for more than six hours. The dataset obtained from monitoring and preprocessing the aforementioned dimensions yielded a dataset with 70 features (*i.e.*, 12 related to the network, 21 to memory, 12 to file system, 4 to CPU, 14 to process scheduler, 7 to drivers). In total, the dataset consisted of 21,600 vectors with roughly 2160 vectors corresponding to each behavior [267].

Table 5.2: Model Performance [267]

Algorithm	k-NN	SVM	XGB	DT	RF	MLP
Avg. F1-Score	0.88	0.94	0.98	0.97	0.98	0.95

An initial detection system was built offline, by leveraging the obtained dataset and testing various models using a common splitting strategy (*i.e.*, training on 80% of data). The models included *k-Nearest Neighbors* (k-NN), *XGBoost* (XGB), *Support Vector Machine* (SVM), *Decision Tree* (DT), *Random Forest* (RF), and *Multi-Layer Perceptron* (MLP). The resulting models were compared offline, computing and comparing the F_1 score of each model, as indicated in Table 5.2, which presents the classification outcomes for each model, measured by the average F_1 score. Two models (*i.e.*, RF, and XGB) achieved the highest observed F_1 score of 0.98. Due to superior training time and resource consumption at inference, the RF model was deployed within the framework. In fact, it was later

demonstrated to be feasible to execute the full reactive pipeline on the device, which required 39.46 seconds for training and 0.0427 seconds for inference. In terms of resources, persisting the model on disk or loading it into memory requires 89.55 MB [267].

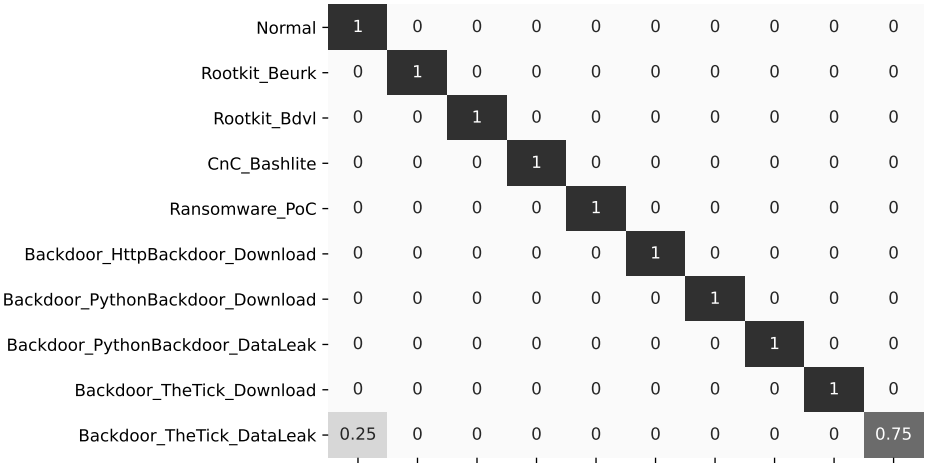

Figure 5.4: Confusion Matrix of the RF Model [267]

To obtain a better intuition of which attacks can be detected (and thus, presumably, be reactively mitigated), Figure 5.4 highlights that only data exfiltration for *TheTick* was misclassified in 25% of cases. The other behaviors, including normal operation, were detected almost perfectly. Based on these results, it is assumed that reactive deployment could provide accurate detection for the MTD framework. However, experiments involving the enforcement module are needed to understand the utility, especially since MTD is considered a second line of defense (*i.e.*, a malware sample is already executing) and $\approx$ 10 seconds are needed to inform the defense [267].

Defense Effectiveness and Efficiency

Using the trained models, the overall MTD framework was assessed, analyzing both effectiveness and efficiency during attack and normal operation. Table 5.3 demonstrates the result of deploying the framework and infecting the device with each sample. The presented results indicate the average of ten rounds of execution, where the *length* is the most important to gauge the defense effectiveness since it establishes for how long the malicious payload executes. The remaining metrics indicate the overall resource consumption of the system, which is reported as an aggregate to highlight that both the MTD techniques and the malicious execution contribute to efficiency. Finally, the last column

Table 5.3: Enforcement Efficiency and Effectiveness [267]

Scenario	***Length*** [s]	***CPU*** [%]	***RAM*** [%]	***I/O*** [Blks/s]	***Sensing*** [KB/s]
Baseline					
Sensor	1035	23.37	15.53	0/0	16.34
MTD Framework	1035	23.94	15.99	0/0.3	15.86
Reactive MTD					
Botnet	42	22.75	8.44	0/1.5	16.31
C&C MTD	12	19	8.43	0/6	15.10
Data Leak	112	29.33	14.27	2/1	15.28
File Suffix MTD	56	35.20	14.29	3.60/1.50	15.14
Backdoor	112	22	17.27	0/0.25	15.85
C&C MTD	14	21.5	17.46	0/1	9.37
Ransomware	84	83.86	9.85	79/572	15.50
Encryption MTD	70	90.66	9.89	93/667	15.64
Rootkit	42	27.50	13.43	0/0.5	15.40
Libraries MTD	12	34.00	13.45	0/2	15.30
Proactive MTD					
Rootkit	29	40	11.71	0/25	16.51
Libraries MTD	1035	25.82	11.63	0/5.49	16.26

implies the impact on the normal workloads' usability, here measured as the throughput being sent to the crowdsensing platform [267].

The first two scenarios represent the baseline resource consumption when no malicious behaviors are present. No MTD techniques were deployed due to a misclassification. Thus, the scenario executes for the maximum time allowed per experiment (*i.e.*, 1,035 seconds). As indicated, the overhead of introducing the framework running in a passive state (*i.e.*, no MTD techniques being deployed) is minimal. When limiting the framework to deploy in a purely reactive or proactive manner, the durations of each experiment indicate the superiority of reactive MTD. Only one proactive scenario is presented; all others did not improve over the reactive scenario. Still, the MTD framework is able to mitigate all attacks, with the reactive scenario improving in terms of effectiveness. For instance, the data exfiltration scenario runs uninterrupted and undetected for 112 seconds. Then, the *File Format* MTD is reactively deployed, acting actively for 56 seconds and concluding the experiment. Here, 300 MB of benign PDF files represented sensitive user data – out of which only 11 MB were exfiltrated. Concerning the impact on the *ElectroSense* service, the techniques do not exhibit a considerable effect

on sensor data transmission. Overall, reactive mitigation appears to optimally deploy the techniques that effectively defend against attacks while keeping resource impact to a low number [267].

Nonetheless, by accepting a few monitoring cycles of malicious execution time and then purposefully wasting system resources (*i.e.*, applying MTD), there is a noticeable effect on system resources, especially for impactful payloads. For example, ransomware and the related File Encryption MTD strongly impact computational resources, especially CPU consumption. Figure 5.5 visualizes this impact for three stages of an experiment involving ransomware, where (1) indicates the start of the malware, (2) denotes the deployment of the MTD, and (3) locates the termination of the sample. As can be seen, the defense further increases resource consumption, presenting a tradeoff between resource usage and defense efficacy, since the MTD framework limited execution to 84 seconds and the file-based defense limited data loss to 7.1 MB. While this does present a strong increase over an "uninterrupted" scenario, where the full 300 MB would be lost, operators would have to gauge for their particular threat model whether this loss is permissible [267].

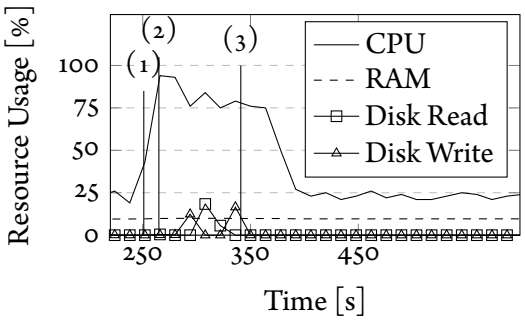

Figure 5.5: Defense Example [267]

Offensive Reactivity

Reactivity, as provided by the *Decisioning Module*, did improve the defense effectiveness. Thus, it did positively contribute to the overall field of research that uses ML-based methods for attack detection – showing that not only is detection feasible but that it does actually improve defense down the line. However, in this last experiment, *RansomAI* is developed to assess how the usage of AI-based attack methods changes this narrative, assuming that both defense and offense would use them [266].

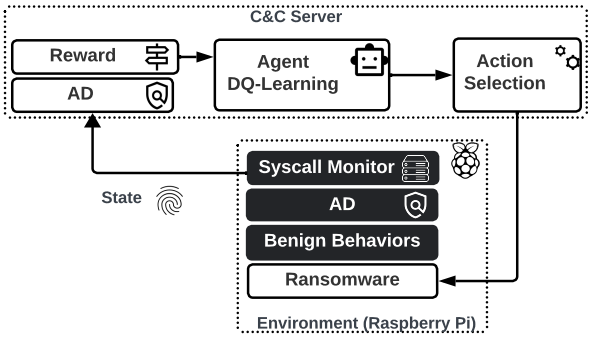

Figure 5.6: *RansomAI* Architecture [48]

Figure 5.6 presents the architecture of the framework that leverages many of the same concepts as the defense framework. At the core, an RL agent learns how to adapt its defense under the assumption that the defender has some reactive mitigation system. It exploits one specific flaw of the previously presented framework: attack mitigation is not completely stealthy. For example, the previous MTD employing a file-based honeypot terminated the encryption process, thus presenting a clear signal to the attacker about the success of its action. The problem to be solved by the agent is to leverage that signal, contextualize it with respect to the action taken (*i.e.*, the encryption behavior) and the current behavior of the device, and learn a more optimal attack policy [266].

Table 5.4: Ransomware Configurations [266]

Configuration	***Algorithm***	***Rate*** [B/s]	***Burst Duration***	***Burst Pause*** [s]
1	ChaCha20	16	1 file	60
2	AES-CTR	565'565.65	unlimited	0
3	Salsa20	632'834.80	unlimited	0
4	AES-CTR	500	10 s	5
5	ChaCha20	200	20 s	40
6	Salsa20	200	120 s	30

To provide a prototype of the ransomware's architecture, *RansomAI* assumes the learning environment to be a Raspberry Pi 4 that executes the *ElectroSense* workload, as indicated in the defense scenario. To circumvent the reactive mitigation system, the Ransomware-PoC sample is adapted to have six encryption configurations (*i.e.*, the potential actions the agent can take). The configurations are indicated in Table 5.4. Additionally, a C&C server is introduced, which learns and orchestrates the execution. *RansomAI* employs the same features used by the defender to build the *States* representing the environment, leveraging the same commands to obtain these features. Furthermore, an *Anomaly Detector* is built to replicate a generic detection system assumed to be present on the defensive side. Based on the inference of the AD, the rewards are computed. Since the agent aims to maximize two conflicting goals (*i.e.*, maximizing data loss and minimizing detection), both the encryption rate and the output of the AD are considered in the reward function. Based on the triples of each iteration (*i.e.*, current state, action taken, reward obtained), the *Agent* learns the optimal policies using a Deep Q-Learning algorithm, with a hidden layer of 25 neurons and the output layer representing the six possible actions [266].

To optimize the hyperparameters used by the agent, several factors were assessed, including the exploration parameters, the number of neurons in interim layers, the activation functions, and the

strategy to initialize the model's weights. A baseline of the agent was built, and testing included varying one parameter each. The baseline was obtained by running 5,000 Episodes and leveraging ε = 0.4, δ = 0.01, α = 0.0050, γ = 0.10, interim neurons = 40. Furthermore, the *Log* and *SiLU* activation functions were tested, and initial weights were derived using the *He* method. After optimization, the optimized agent leveraged the following hyperparameters: $\varepsilon = 0.20$, $\delta = 0.01$, $\alpha = 0.005$, $\gamma = 0.30$. 25 neurons were used and *Log-SiLU* as an activation function. For a more in-depth explanation of the optimization of exploration settings, learning rates, discount factors, number of neurons, and activation functions, the reader may refer to [266]. A study building on top of *RansomAI*, optimizing it, and testing it in more challenging defensive settings is presented in [48].

Based on this instantiation of the agent, the initial question – whether an RL-based agent could leverage similar mechanisms as a reactive defender and learn a stealthier behavior – can be answered. To do so, ground-truth information was defined against which the ransomware selection can be assessed. As outlined in Table 5.5, the agent can learn the optimal offensive policy with an accuracy of $> 95\%$ in just 300 episodes (*i.e.*, $\approx$ ten minutes) [266].

Table 5.5: Accuracy of Offensive Reactivity Agent [266]

Episodes	***Learning*** [min]	***Accuracy*** [%]
100	2.0	91.43
200	5.1	94.86
300	6.5	96.32
400	8.1	96.21
1'000	23.9	98.61
2'000	66.4	99.07
5'000	172.2	99.71

To summarize, it must be made clear that *RansomAI* relies on rather strong assumptions with respect to the defender. For example, the approach assumes that the defender uses a similar behavioral detection and mitigation system or that the agent can continuously test multiple configurations. Most importantly, it is argued that the existence of flaws in the defender's model is the key assumption here. If the defender's model can detect every configuration perfectly, even ones encrypting at a low rate, there would be nothing for the attacker to learn. For example, Table 5.6 highlights the performance of the Anomaly Detection system when confronted with normal behavior and the six ransomware configurations [266]. Apparently, only *Conf. 2* and *Conf. 3* are detected by the AD with a low number of false negatives. However, those are also the two configurations with the highest encryption

Table 5.6: Anomaly Detector Performance [266]

Behavior	*Normal*	*Conf. 1*	*Conf. 2*	*Conf. 3*	*Conf. 4*	*Conf. 5*	*Conf. 6*
Performance	88.94	91.62	0.62	0.21	80.18	82.05	77.38

Normal denotes TPR, *Conf. n* FNR

rate. *Conf. 4*, is less accurately detected; however, it encrypts at merely 500 B/s. Thus, *RansomAI* can stealthily encrypt, albeit at a low rate. Nevertheless, this emphasizes the limitations of the reactive MTD framework: if attacks are not detected in the reactive setting, they remain completely unmitigated, arguing for the inclusion of proactive measures, too. Furthermore, while the framework demonstrated strong defense against existing samples, dynamic, self-adapting malware (or even the manual adaptation) would reduce that defense effectiveness [266].

5.1.3 Key Observations and Discussion

Developing and testing the MTD framework and the offensive counterpart raises several observations. First, it appears that both a reactive and a proactive deployment approach demonstrate utility. Proactive approaches appear especially suitable for malicious behaviors that could be hard to detect, such as passive or low-impact behaviors (*e.g.*, persistence, logging). However, for many malware-based attacks, a reactive deployment component may be both more effective and efficient. It is hypothesized that the skew of this tradeoff could strongly depend on two factors: proactive deployment frequency and defense cost. If an MTD technique could incur close to no overhead, it could be deployed frequently, thereby improving over a reactive scenario, which would still be prone to misclassifications (*i.e.*, unnecessary deployments or missed deployments). Secondly, it appears that the reactive deployment has a net benefit on overall resource consumption (*i.e.*, comparing to an unmitigated case). However, even the simplified models used here require manual rule-based configuration (*i.e.*, which action to trigger for a classification outcome). Furthermore, the monitoring system does require computational resources, such as the roughly 90 MB model, to be stored on disk and loaded into memory. Finally, it must be restated that the evaluations consider a static scenario, where neither the attacker nor the normal behavior is adapted. This assumption appears as the key limitation of the work, where attackers could adapt the malware – or develop a self-adapting malware, as was demonstrated using *RansomAI*. In that scenario, the threat model of the data collection systems would need to be re-evaluated, too. After all, data is collected on a compromised system, which presents a questionable assumption.

The subsequent studies are directly motivated by the results of these experiments. For example, *MTFS* aims to develop more lightweight versions of the MTD techniques that target ransomware, with the goal of being cheap enough to be deployed proactively. *SecBox* analyzes how the data – that in this study was collected on a compromised host – could be extracted securely and scalably using containerized sandboxing. *GuardFS* aims to do the same but leveraging a host-based perspective where data is extracted in the kernel to reactively defend against a user-space threat. Finally, *R-MTD* aims to replicate the decisionsing module, hypothesizing whether a completely rule-based reaction could maintain effectiveness while increasing efficiency. All of those systems are then evaluated from both offensive and defensive perspectives. Importantly, different aspects were selected to be tested based on the objective of each solution. These aspects inevitably influence the degree of control enacted in each experiment and, thus, the generalizability and relevance of results. For example, *SecBox* tests whether malware samples affect the underlying host system. To ensure numerous samples were tested, samples had to be executed non-interactively. These results may be easier to generalize than those of other studies where only one sample is executed manually, however, at the loss of relevance.

5.2 Rule-based Reactive Moving Target Defense

R-MTD builds on top of the previous study (*cf.*, Section 5.1), focusing on an alternative path to overcome the same limitations discovered in the literature (*i.e.*, *L6)* and *L7)*). In addition, it considers the limitations of that previous study: rules to decide which technique to deploy were assumed to exist without presenting a concrete methodology, and the resource consumption of the ML-based reactivity module could be further optimized.

5.2.1 The *R-MTD* Approach

At the core, *R-MTD* proposes and validates a methodology to build a rule-based reactivity system aligned with a particular threat model. Figure 5.7 highlights the four stages of the methodology that are to be connected with an established threat model. By following these stages – which are subsequently introduced – a rule-based MTD framework following the architecture indicated in Figure 5.8 is obtained.

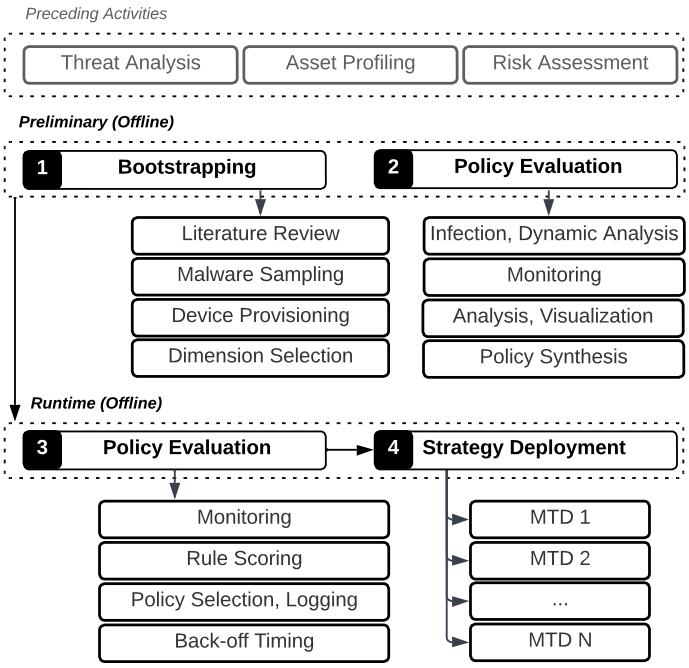

Figure 5.7: Methodology to Develop Rule-based MTD [270]

On the left of the architecture, a set of techniques must be available for deployment. The prototype of *R-MTD* leverages the four MTD techniques proposed by [267]. It is assumed that the techniques are applicable to a family of malware and not just one particular sample. To properly decide about not just *when* defense is necessary, but also *which* of the techniques to deploy, each technique must be referenced by at least one *policy*. Each policy is composed of one or more rules. For example, a hypothetical behavioral rule for ransomware could require $CPU > 25\% \wedge WRITE > 5MB/s$, *i.e.*, that at least one of four cores is fully utilized and that there are writes to the disk. To evaluate those rules, an *Observer* must gather those values from an *Agent* monitoring the device. If the rules in any of the policies are fully met, the *Deployer* takes care of technical details involved in deploying the techniques (*i.e.*, how to start them, which parameters to hand over to the defense) [270].

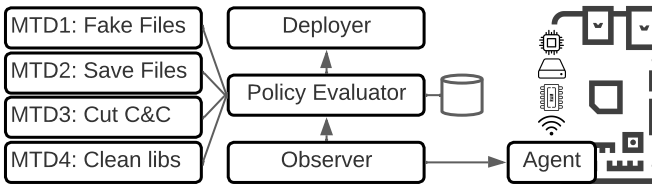

Figure 5.8: Rule-based MTD Framework Architecture [270]

To instantiate the framework, users start in the *Bootstrapping* phase; malware samples relevant to the threat model have to be identified and obtained (*e.g.*, from commercial providers, open-source databases). Since the approach relies on a behavioral approach, a device that is comparable to the target device must be obtained and provisioned. Before gathering that data, relevant features have to be identified, explored, and analyzed. Based on the previous study, behavioral aspects, such as system metrics (*i.e.*, I/O throughput, CPU workload), are suggested as an initial scope. As a starting point, the 80 metrics described by [218] or [47] can be reviewed [270].

Once an initial hypothesis of usable dimensions is established, the *Policies* are *generated* by evaluating the behavior of samples using those dimensions. Employing a behavioral approach implies that samples are dynamically analyzed and, thus, executed. Measurements of benign and malicious workloads are obtained through infection and execution. Then, data must be analyzed to build rules and thresholds that are composed of policies. This can be done by computing and analyzing statistical properties (*e.g.*, minimum and maximum values, standard deviation) and by leveraging visualizations. For each MTD technique, an appropriate time window for monitoring and processing, and a threshold must be defined. To build sensible thresholds, metrics that do not notably differ across behaviors are discarded. Here, the analyst must also analyze the effect of the MTD techniques – values should return to a normal stage, otherwise they would trigger additional false deployments [270].

Once the policies are built, the framework can be deployed online. *Policy Evaluation* mandates that the same monitoring is executed and the policies are evaluated. It is crucial to build records of the actions that are taken to properly orchestrate the deployment of techniques in a stable manner. For example, after deploying a specific technique, the monitoring in the next time window may still lead to a positive result. However, depending on the MTD technique, subsequent deployments should be spaced apart [270].

The final stage involves the actual deployment of the technique, which is informed by the techniques available to the particular device. Technical delays and constraints imposed by the techniques should be aligned with the orchestration. In the long term, it is advisable to analyze the logs after deploying the framework to assess whether the policies operate effectively and efficiently or whether a policy refinement is needed [270].

5.2.2 Prototypical Implementation

A Python-based prototype of the framework was developed, which can be found in [65], along with the data that was leveraged to synthesize the policies. The framework is instantiated using a similar threat and defense model, leveraging *ElectroSense* as the benign scenario. By focusing on three malware families, the methodology is executed for Command-and-Control-based malware, Rootk-

its, and Ransomware. Table 5.7 presents an initial hypothesis indicating potentially useful features for those three behaviors, as described by related literature [270].

Table 5.7: Initial Feature Hypothesis [270]

	Processes	*Disk I/O*	*Network conn.*	*Network I/O*	*CPU*
C&C	✓	✗	✓	✓	✓
Rootkit	✓	✓	✓	✓	?
Ransomware	✓	✓	✗	✗	✓
Reconnaissance	✗	✗	✓	✓	✗

✓ = Potentially Affected ✗ = Unaffected

To start the implementation of an initial prototype leveraging those features – informed by the threat model of *ElectroSense*– seven malware samples were acquired (*i.e., httpBackdoor, backdoor, The Tick, BASHLITE, RansomwarePoC, BEURK,* and *bdvl*). Using a real *ElectroSense* sensor, a subset of 33 metrics falling into the four categories was collected using the `dstat` CLI tool. To generate policies, the metrics were captured for both normal behavior and infected phases, each monitoring experiment lasting ten minutes. The `dstat` command yielded data in one-second time windows [270].

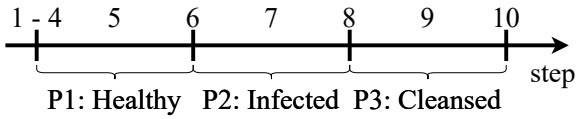

Figure 5.9: Three-stage Dynamic Experiments [270]

360 measurements of each scenario were available for data analysis. Each dataset represented various stages to model how the introduced payload affects the features. As demonstrated in Figure 5.9, each infection is preceded by non-interrupted and benign execution and proceeded by a post-mitigation phase. The latter is needed to gauge whether a technique – introduced during infection – leads to a return to normal behavior. In addition to those stages, the *ElectroSense* sensing and data transmission workloads are run in parallel [270].

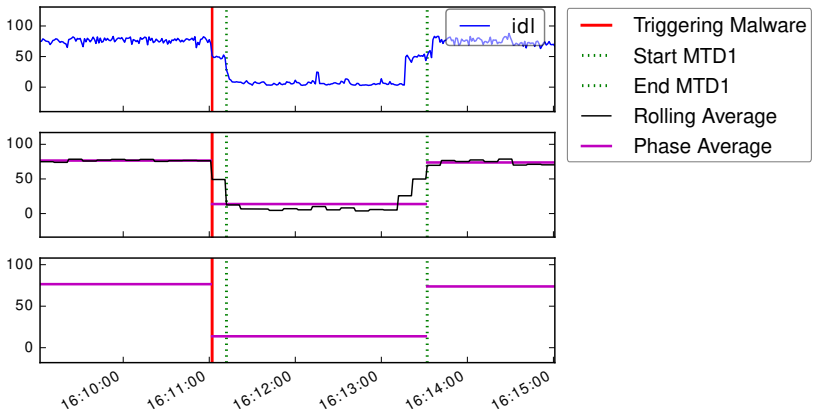

Figure 5.10: RansomwarePoC and MTD1: CPU Idle Time [270]

In the next stage, the effect of the four available MTD techniques (*cf.*, Section 5.1.1) on these stages had to be visually analyzed: by means of a simple Jupyter notebook, the datasets were loaded, preprocessed, and the features plotted uniformly, by rendering the relative metric (y-axis) over time (x-axis). By comparing the experiments with varying degrees of smoothing (*i.e.*, averaging over 1-second, 5-second, and per phase), a threshold was visually identified. This is exemplified in Figure 5.10, where an experiment involving *RansomarePoC* and *MTD1* is shown for the metric indicating the time the CPU spends idling [270].

Table 5.8: Defense Policies [270]

Metric	***Condition***	***Threshold***	***Action***
idl	<=	50	MTD1
sys	>=	24	MTD1
usr	>=	10	MTD1
writ	>=	200000	MTD1
tim	>=	6	MTD2
new	>=	17	MTD3
recv	>=	2500	MTD4

By analyzing the plots, the majority of metrics fitting the initial hypothesis (*cf.*, Table 5.7) were discarded by testing whether they yield a noticeable change after deploying the sample (*i.e.*, the red vertical line in Figure 5.10). From the remaining subset of metrics, ones that differ across payloads were further selected. Finally, from this set, the ones that are robust were selected: if values did not

return to the initial stage, they were discarded (*i.e.*, when the third stage did not resemble the first stage). Finally, thresholds were manually defined based on one of the granularity levels, and rules were synthesized into the policy set indicated in Table 5.8. Specifically, 10-second intervals were selected to provide robust detection, and 60 seconds was determined as a realistic buffer. As such, this buffer should be respected by the *Deployer* even when the policies sequentially evaluate to true [270].

To deploy these components in a productive environment for the subsequent evaluations, the policies were stored in a CSV file. A Python-based agent was developed to continuously monitor the target device using the same Linux tools, compute the smoothed values, and evaluate the policies. In cases where a policy indicates MTD deployment, the respective component interacts with the interface of the previously introduced MTD framework. In addition, the prototype tracks the orchestration to avoid subsequent deployments (*i.e.*, respect the aforementioned buffer of 60 seconds) [270].

5.2.3 Evaluations

In the following, the prototypical implementation is used in the context of *ElectroSense* to establish the detection performance, the ability to mitigate a multi-purpose attack, and the efficiency of the overall system.

Detection and Mitigation

In the following, the policies are tested by individually testing the malicious samples in a controlled environment. At the beginning of each experiment, the device is restored to a completely clean state, running only the productive *ElectroSense* workload. Thus, the experiments cannot be considered to be fully controlled since the *ElectroSense* sensing imposes some external factors that are not controlled. Similarly, the malware itself may interact with third-party controllers, and its behavior may vary over time. The experiment sacrifices some level of control in favor of relevance (*i.e.*, leveraging real samples in a productive scenario) [270].

The results of these experiments are displayed in Table 5.9. The results do not explicitly indicate any false positives or false negatives in detection – instead, those are incorporated into an overall picture represented by the starting point of the malware and the MTD. This establishes the actions taken, the time needed for detection, and the subsequent mitigation until the device returns back to normal operation [270].

The key results are that, except for *BASHLITE*, which exhibited mostly passive behavior, all samples are successfully detected. Upon detection, all samples are successfully mitigated after a few it-

Table 5.9: Detection and Mitigation Effectiveness [270]

	Trigger point		Duration (s)		
	Malware	***MTD***	**Action**	**Detection**	**Mitigation**
Ransomware-PoC	07:15:00	07:15:26	1	26	143.78
httpBackdoor	07:26:00	07:26:26	2	26	8.94
BEURK	07:38:00	07:38:04	3	4	0.3
The Tick	07:50:30	07:52:34	4	124	9.01
backdoor	08:10:00	08:10:38	4	8	9.02
bdvl	08:22:00	08:22:04	3	4	0.25
BASHLITE	08:32:00	-	-	-	-

erations. For example, *Ransomware-PoC*, *httpBackdoor*, and *The Tick* were not initially detected, requiring more than two monitoring cycles. The remaining of the successfully detected samples were discovered in the first cycle. To explicitly report detection performance, the monitoring window was shortened, and no actions were executed. Aside from the highly passive *BASHLITE*, most samples were eventually detected, as indicated in Table 5.10. Still, there are strong differences in how accurate and early certain behaviors are detected. Especially passive ones, including backdoors and rootkits, that spend most of their time in a passive state, are likely easier to detect when considering the combination of many features, as demonstrated by the previous ML-based detection system. Nevertheless, once those samples start any actions, the MTD deployment is initiated, as demonstrated by Table 5.9. In that sense, the rule-based system does provide a baseline defense system that eventually detects most techniques in a simpler yet less reliable way [270].

Illustrative Defense Scenario

The previous experiment analyzed whether the framework successfully detects singular attacks and deploys a suitable technique. The second experiment involves multiple diverse infections in series (*cf.*, Table 5.11), testing if those can be detected and whether the framework can recover to the normal state. *BEURK* is injected into the device after a brief phase of benign sensing. After seven seconds, the behavior is detected, prompting a 1-second defense through *MTD-3*. After another minute of benign behavior, *RansomwarePoC* infects the device, running undetected for 11 seconds. *MTD-1* cleans the devices in 143 seconds of operation. *httpBackdoor* does similarly for 30 uninterrupted seconds, which is finally mitigated by *MTD-2*. In sum, each behavior is eventually mitigated, and the device returns to a normal and presumably benign state without any false deployments being issued [270].

Table 5.10: Detection Performance [270]

Behavior	*Metric*	*P1*	*P2*	*P3*	*P4*
healthy	TNR (%)	100	100	97.59	100
	FPR (%)	0	0	2.41	0
"Ransomware-PoC"	TPR (%)	25.32			
	FNR (%)	74.68			
"httpBackdoor"	TPR (%)		96.43		
	FNR (%)		3.57		
"BEURK"	TPR (%)			3.61	
	FNR (%)			96.39	
"bdvl"	TPR (%)			5.33	
	FNR (%)			94.67	
"backdoor"	TPR (%)				100
	FNR (%)				0
"The Tick"	TPR (%)				100
	FNR (%)				0
"BASHLITE"	TPR (%)		0		1.20
	FNR (%)		100		98.80

TPR = True Positive Rate, FPR = False Positive Rate
TNR = True Negative Rate, FNR = False Negative Rate

Resource Consumption

MTD serves as an additional layer of defense; thus, it is important to assess the overhead introduced by each layer, particularly in the context of constrained devices. The effects of the strategy selection agent are assessed using the actual device during operation to analyze the performance impact of the strategy selection agent. The deployed selection framework was monitored over a one-hour period. Memory usage and relative CPU consumption were recorded every ten seconds during this time frame. Since this work focused on rule-based selection, reusing the previously introduced MTD techniques, measurements focused solely on the monitoring, policy evaluation, and deployment aspects of the framework. The measurements involved both normal behavior, where the framework yields pure overhead and attack scenarios [270].

Figure 5.11 visualizes the result and contextualizes it based on the *RansomwarePoC* infection. The selection agent consistently utilizes 44 MB of memory, corresponding to $\approx$1% of the overall memory available on the system. Considering the consumption of CPU cycles, an initial spike (*i.e.*, 11% relative load) was observed for the first ten seconds. In the long term, CPU consumption stabilizes, requiring $\approx$ 1%. Internally, the policy database occupies only a few hundred bytes, making the ap-

Table 5.11: Defending Multiple Malware Infections [270]

Index	***Status***	***Phase***	***Duration*** [s]	***Description***
1	healthy	healthy	60	pause
2	infected	detection *b*	7	trigger *r*
3	infected	deployment *b*	0.33	MTD 3
4	healthy	healthy	60	pause
5	infected	detection *r*	11	trigger *r*
6	infected	deployment *r*	143.44	MTD 1
7	healthy	healthy	41	pause
8	infected	detection *h*	30	trigger *h*
9	infected	deployment *h*	9.07	MTD 2
10	healthy	healthy	238	pause

b = "BEURK", *r* = "Ransomware-PoC", *h* = "httpBackdoor"

proach transferable to less capable devices. Moreover, the agent was not yet optimized for efficiency; for example, an implementation in another language or runtime could still considerably reduce the overhead [270].

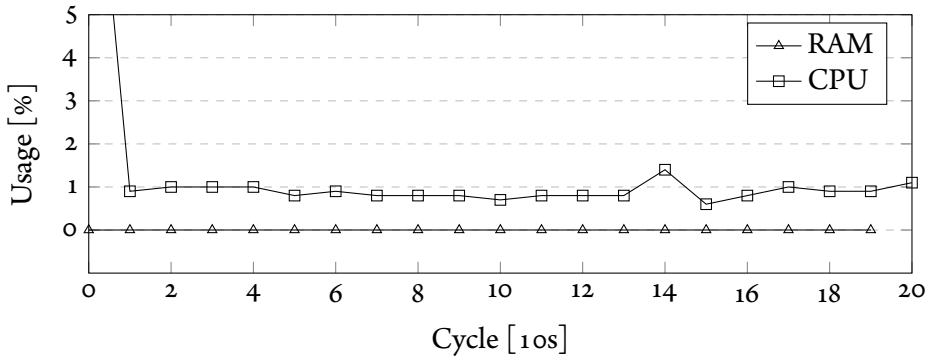

Figure 5.11: Framework Efficiency [270]

5.2.4 Key Observations and Discussion

Based on the experience drawn from developing *R-MTD*, deploying it in a defense scenario based on *ElectroSense*, while contrasting the results with the previously outlined hybrid defense framework, the following observations are raised:

1. Based on the specific threat model that was employed for testing, a rule-based, reactive MTD framework can be used to add an effective line of defense. For several behaviors, mitigation appears successful while potentially improving the efficiency of the *decisioning* module. This

was demonstrated to be the case for not just one but several malicious behaviors, providing further evidence of MTD being effective against multi-purpose attacks.

2. Due to the human-in-the-loop approach when developing policies, several limitations arise. First, a human operator may not find optimal "orthogonal" values that clearly pinpoint one attack. Thus, there are non-negligible development efforts.

3. Closely related to the last observations, the resulting engine is unlikely to be able to discover highly stealthy attacks – as exemplified by the passive behaviors tested. Here, an ML-based system could likely identify the behavior by learning patterns from a higher number of dimensions.

4. Similar to the last study, this framework assumed a non-adaptive adversary. As such, the results are coupled to the threat model used in these experiments.

5.3 Obtaining Data Through Dynamic Execution

The previous solutions already assessed that behavioral sources of data collected through dynamic execution are useful for detection, as validated through defensive scenarios. However, these solutions operated with the assumption that it is efficient to perform "the infect – collect – detect" cycle by relying on the actual devices to be protected. While this may yield realistic data, it is not necessarily a safe assumption: after all, malicious payloads are executed in the development environment. In addition, while this could work for a small threat model, targeting a few malware families and only a limited number of samples, it might not scale for a larger viewpoint. For example, Section 2.3.1 exemplified that even one malware exchange platform recorded more than a dozen new submissions per day. If organizations relied on long-term success, the ML pipeline (*cf.,* Section 2.3.2) would need to be continuously adapted. Hence, this study, where *SecBox* is designed, focuses on this limitation (*i.e., L5)*), addressing how to efficiently and effectively collect behavioral data from malicious samples. To do so, *SecBox* explores how container-based isolation could be used to provide reproducible, scalable, collaborative, secure, and rich data collection based on dynamic analysis.

5.3.1 Architecture and Implementation

At the core, dynamic analysis requires an execution environment and a means to monitor the execution. As outlined by the main modules in the *SecBox* architecture displayed in Figure 5.12, several components surround the actual sandboxes, where malicious payloads are executed. To provide the sandboxing (*i.e.*, runtime isolation), *SecBox* explores the viability of using Linux containers – which are likely more lightweight, albeit providing less isolation, hence not being widely explored by related literature (*cf.,* Section 3.2.1). To support downstream methods, such as anomaly detection or attack classification, two sandboxes are created for every experiment. For example, an analyst requiring data on *RansomwarePoC* would execute the payload, yielding one dataset containing purely benign behavior and one with benign and malicious behavior. To create these datasets, *SecBox* monitors the sandboxes at runtime, extracting various features from the hypervisor, including resource consumption, system calls, and network traffic. In the following, the components supporting the functions (*i.e.*, providing execution, monitoring, persistence, and user interaction) are introduced [81]. Through these components, the following desired properties of the *SecBox* design are targeted:

1. **Performance**: minimizing the resources required to execute a sample.
2. **Isolation**: securing the environment against attacks involving escapes and escalation.

3. **Low Visibility**: contradicting Isolation; a sample should not be able to detect the sandboxed environment and alter its behavior.

4. **Monitorability**: extracting rich data.

5. **Automation**: minimizing manual effort.

6. **Reproducibility**: maximizing the similarity of outputs during subsequent analyses.

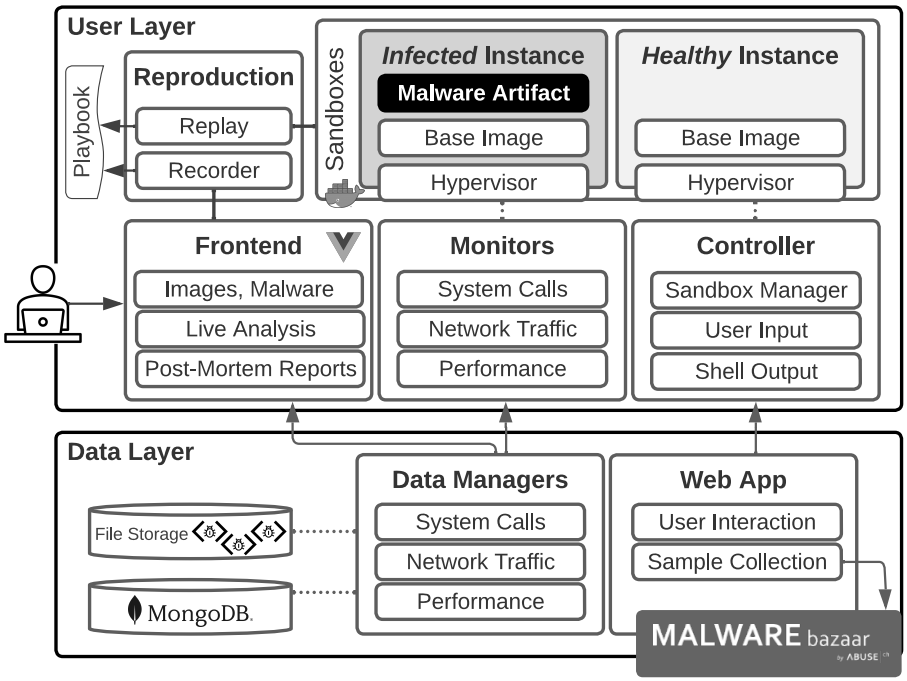

Figure 5.12: Container-based Dynamic Analysis Architecture [81]

At the core, malware must be executed in an isolated context. Since containerized isolation does not fully separate kernel-level resources, *SecBox* uses *gVisor*, an application-level kernel that reimplements (a subset of) system calls made from the container to the underlying kernel. Thus, it provides an additional layer of isolation [81]. Its developers claim that this degree of isolation allows execution of untrusted code without concerns for execution or escaping threats [250]. Related studies demonstrated that it provides stronger security guarantees than other container-based runtimes [286].

Monitoring this execution is facilitated through available interfaces. A *Monitor* operates on the same machine as the sandbox to extract that data. Metrics on resource consumption are obtained by relying on the Python-based *Docker* API, while network traffic can be sniffed on the virtual interfaces

of the host operating system. Due to the application-level hardening, system call invocations are gathered directly from *gVisor*. The persistence and processing of this data can be offloaded – a set of *data managers* regularly process and store data in a document database. Furthermore, several real-time metrics are computed and made available for the user interface. By using *WebSockets* for communication, any of those components could run distributedly. For example, the sandboxes could run on one host, persistence occurring on another, and users access the platform from their own premises through the browser. This not only decouples those systems but makes the most resource-intensive task (*i.e.*, execution) parallelizable. To synchronize the input and output of the distributed components, a *Controller* relays those streams and manages the life cycle of the components (*e.g.*, spinning up sandboxes, destroying them upon user signal) [81].

Malware samples require input data (*e.g.*, C&C orchestration, parameters, human user input). To run a sample, the UI prompts the user with locally available base images for the sandboxes. Malware samples are retrieved from MalwareBazaar [4], which the user can conveniently select. Based on these two visual components, the user parametrizes the malware and its execution environment. After starting the experiment, the front end renders a dedicated page to visualize the output during the live analysis phase, where users can interact with the sandboxes by interacting with the shells of the sandboxes. Here, user input can be conveniently redirected to one or two sandboxes, simplifying the creation of benign or malicious behavior. Figure 5.14 illustrates how the prototype provides this through the respective UI components: a full-sized input box indicates input being relayed to both shells, and a button allows toggling this feature. While doing so, feedback in the form of visualized metrics (*e.g.*, CPU usage, packet transmissions, number of active processes) is provided [81]. A subset of them is illustrated in Figure 5.13

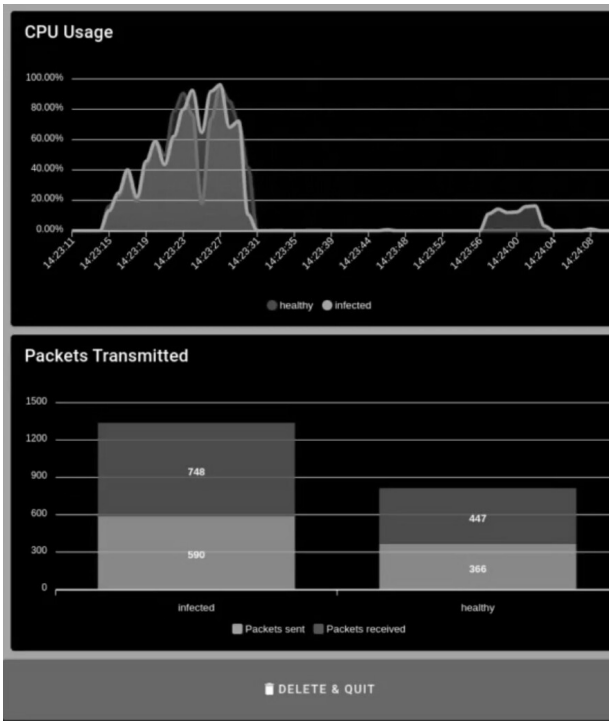

Figure 5.13: Realtime Visualizations [271]

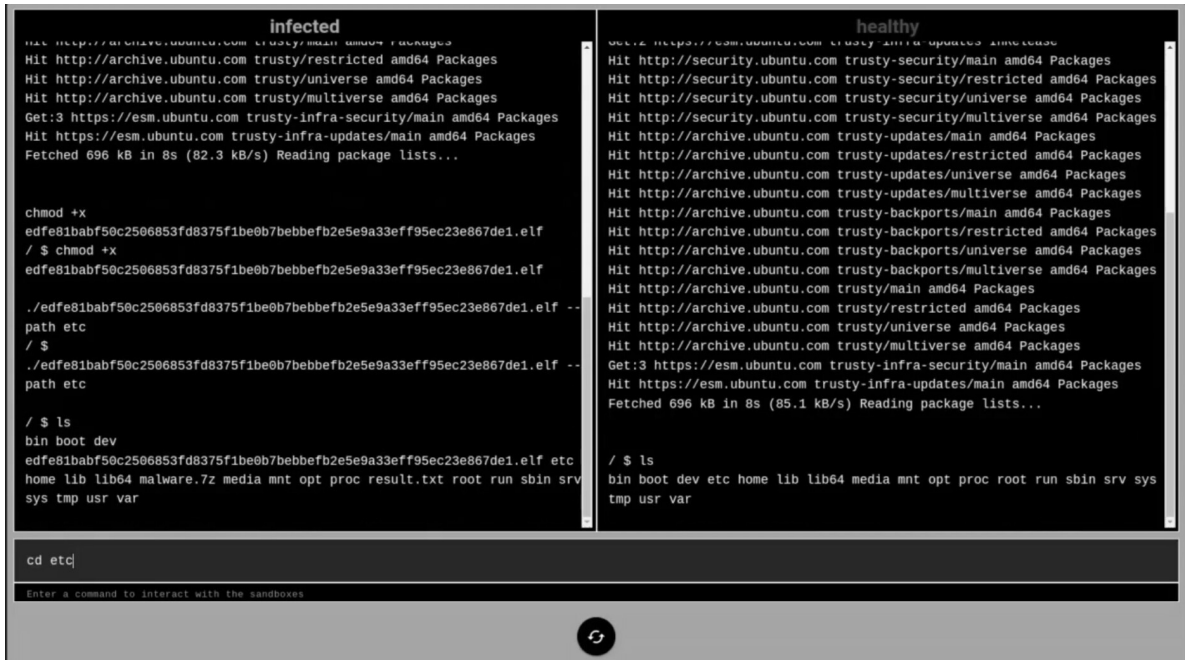

Figure 5.14: Live Interaction being Relayed to Both Sandboxes [271]

Once the user terminates the interaction with the sandboxes, their environment is destroyed, and additional data analyses and reports can be created. On this page, visualizations across the whole experiment can be performed, and annotations can be made (*e.g.*, noting when an experiment started or marking a feature that may be useful). The volumetric features that were not processed in real-time but merely collected (*e.g.*, system calls, network traffic) are then asynchronously processed and visualized (*e.g.*, displaying which file paths were discovered in system calls, comparing the relative share of network protocols used). In addition, raw data can be exported in the form of CSV and PCAP files [81].

During this process, *SecBox* ensures reproducibility and transferability of the execution and analysis by persisting all input parameters. For example, the selected malware is stored within the platform. Similarly, the usage of Docker images ensures that the underlying infrastructure can be reproduced (*e.g.*, running Ubuntu 22.04 LTS with a specific set of tools). Capturing user interaction is critical since the manual interaction with the sample may hinder reproducibility. In *SecBox*, any user input entered during an experiment is automatically recorded and transformed into a script with relative timestamps. Thus, an experiment could be re-executed, altering, for example, the underlying platform but employing the same artifact and the same series of steps [81]. A complete iteration involving several malware samples is demonstrated in [271].

In the following, selected experiments performed on *SecBox* are introduced, and their results are contrasted to the initially postulated design objectives (*e.g.*, Performance, Isolation, Reproducibility).

Resource Consumption To assess the efficiency of the platform, *SecBox* was deployed into a productive environment, enabling controlled experiments where dynamic analysis of real samples is performed. As indicated before, *SecBox* aims to achieve efficiency by *(i)* decoupling the three key areas (*i.e.*, sandboxed execution, data persistence and analysis, and user interface) so that each could be scaled individually or run in different infrastructures. In addition, *(ii)* the platform leverages lightweight execution environments for dynamic analysis, assuming to inherit the benefits of that isolation approach [81].

For this first experiment, minimal infrastructure is leveraged, and all components are deployed into a single VM that is based on the Ubuntu 22.04 operating system. In terms of hardware, 4 CPU cores (2.4 GHz), 16 GB of RAM, and a 50 GB disk were attached to the VM. Then, the load of the three modules was individually measured, collecting CPU and RAM consumption using the `ps` utility. Then, three scenarios are executed, measuring the consumption of these components. First, *SecBox* is executed in a completely passive way (*i.e.*, all components running, no sandboxes deployed). Secondly, two sandboxes are deployed, running an Ubuntu image, starting two instances of the *live analysis* phase. No active benign or malicious behavior is simulated to assess the overhead of the sandboxes themselves. In the third scenario, a resource-intensive malware is deployed: a new scenario is created, deploying an instance of the *CoinMiner* malware [81].

Table 5.12: Resource Consumption [81]

	Sandbox Host		Back end		Front end	
Scenario	***CPU***	***RAM***	***CPU***	***RAM***	***CPU***	***RAM***
Idle	0.5%	0.03 GB	0.6%	0.09 GB	0.21%	0.27 GB
Passive	0.6%	0.26 GB	3%	0.09 GB	0.3%	0.27 GB
Malware	26.0%	0.48 GB	1.0%	0.19 GB	1.5%	0.32 GB

Based on the measurements indicated in Table 5.12, it is argued that *SecBox* generally has low resource requirements, especially in an idle state. Moreover, adding sandboxes is cheap, although the containerized environments leverage additional security components. It is argued that these components can be deployed even in constrained environments. For example, a simple *Virtual Private Server* (VPS) with 1 vCPU and 1 GB of RAM could be used to deploy the whole platform. In existing VM-based isolation, this is unlikely to be possible due to the increased resource consumption

and the requirement for nested virtualization to be supported. Although executing *SecBox* is resource efficient, executing malware may not be. Depending on the sample, this load likely varies strongly, as observed in previous studies (*e.g.*, passive botnets requiring little resources, crypto-ransomware exhibiting strong resource consumption). In the experiment, *CoinMiner* fully saturates one CPU core in the sandbox. The remaining components in *SecBox* only see a minimal increase in consumption and remain fully functional – providing consistent persistence and real-time processing in the backend and continuous visualization in the frontend. As such, *SecBox* arguably achieves the main goal: minimizing the load caused when operating intensive samples [81].

Reproducibility Reproducibility and automation were assessed in one experiment – arguing that a high degree of automation and the preservation of necessary inputs of each dynamic analysis ensure that repeated executions lead to consistent and comparable datasets. Importantly, this is done using real-world samples, which are unlikely deterministic (*e.g.*, depend on particular triggers or outside services). To measure how comparable the output of multiple executions is, cosine similarity, as defined in Equation 5.1, of the performance metric datasets is computed [81].

$$\text{cos_sim}(A, B) = \frac{a_1 \cdot b_1 + a_2 \cdot b_2}{\sqrt{a_1^2 + a_2^2} \times \sqrt{b_1^2 + b_2^2}} \tag{5.1}$$

To obtain the data, a *SecBox* sandbox is deployed using an Ubuntu-based image. Then, one of four malware samples is deployed, leading to the creation of a performance metric dataset. The samples included the Mirai, Monti, and CoinMiner families and a bespoke sample that was developed to be as deterministic as possible for comparison purposes, accessing a file corpus of 1 GB and creating network traffic in a tightly scheduled manner. Each sample is executed ten times. Then, cosine similarity is computed across the runs, as indicated in Table 5.13. As indicated by the average similarity and its standard deviation, execution using *SecBox* leads to highly similar datasets, with the average similarity score closely approaching 1.0. As such, it is argued that, although samples themselves may not be deterministic, the interaction of *SecBox* is easily repeatable [81].

Table 5.13: Dataset Similarity [81]

Malware	***Reference***	***Mirai***	***Monti***	***Coinminer***
Avg. cos_sim	0.998	0.999	0.985	0.996
Std. cos_sim	0.003	0.001	0.022	0.005

Monitorability To assess whether *SecBox* is not just efficient and secure, numerous samples were executed, and the utility of its output was assessed through data analysis and by creating proto-

typical detection models based on the raw datasets. Through both, it is illustrated that *SecBox* can be used to elicit useful features and to create ML models [81, 271].

In the first scenario, *SecBox* operates in the same environment used for resource analysis. First, a sample representing the Mirai botnet family is deployed. To get the sample to run, relevant information about one sample of Mirai is obtained directly from MalwareBazaar. To create a sandbox, Ubuntu 14.04 is employed, which already satisfies several CLI dependencies (*e.g.*, utilities like busybox, cron). The operator then chooses the sample and base image. Based on this, the two sandboxes are automatically created – one containing the malware binary. During the execution of the sample, several visual indicators highlight the characterization of the sample. For example, the number of active processes and the load on the CPU increase. After this initial phase, regular spikes in CPU load and network traffic are visible in the infected environment. This pattern continues over time, indicating that the sample may be in a passive but operational state. Once the analyst concludes the interactive analysis, volumetric data is asynchronously processed upon user request and made available to the user on the reporting page. For instance, a report can be created to summarize the recorded system calls. In the sandbox that contained the sample, 2400 additional system call invocations were made using the root directory as a working directory. From the network traffic, significant ARP scanning can be observed. The spider graph, illustrated in Figure 5.15, overviews the occurrence of packets belonging to different network protocols [81, 271].

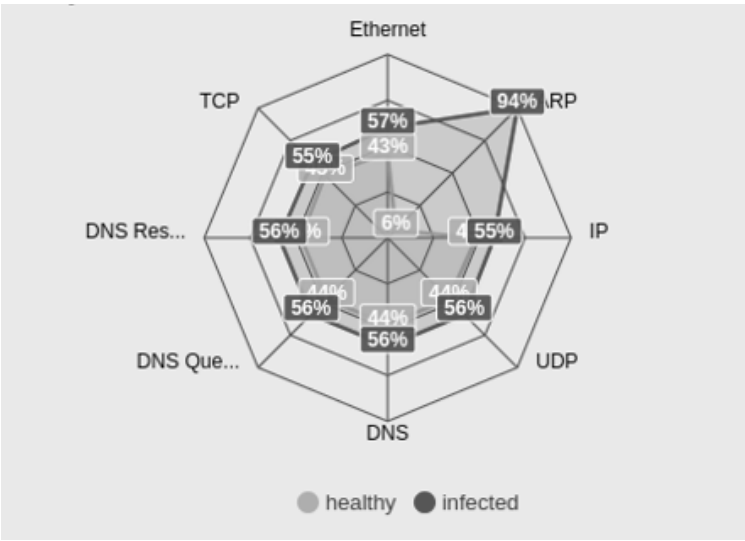

Figure 5.15: Network Usage Across Protocols [81]

From the illustration, it can be deduced that Mirai arguably uses a specific protocol, promoting downstream activities (*e.g.*, detecting malicious behavior using a network-based classifier, creating

network policies to prevent such scanning). In addition, concrete information, such as the related target IP addresses, is presented in a tabular manner. Based on these observations, the underlying datasets (*i.e.*, performance metrics, system call invocations, network traffic) could be exported for those downstream activities. Subsequent iterations can be made in a reproducible manner. For example, if the ML model trained on that data were to decrease in performance after upgrading to Ubuntu 16.04, a modeler could select a different base image – all other parameters (*e.g.*, sample, user interaction) would be reproduced by the platform [81, 271].

To illustrate the utility of data generated by *SecBox* in such downstream activities, [21] describes a design study that implements a complete ML pipeline within *SecBox*. Specifically, the platform was extended so that users could generate datasets, label them according to timestamps, and train models in a no-code approach. As such, models are trained on the whole data during an infection (*i.e.*, including all processes) and do not classify individual processes but the observed system behavior during a time frame. Reproducibility is ensured by recording each hyperparameter used during training. For example, when the user trains a model with the *Local Outlier Factor* (LOF) algorithm, *SecBox* internally persists parameters, such as the number of neighbors or the contamination factor [21].

To test this ML pipeline in *SecBox*, several datasets are generated by dynamically analyzing the *Monti* ransomware. First, the platform is deployed on a bare metal machine with an Intel i7-8750H CPU and 16 GB of memory running the Ubuntu 22.04 operating system. To collect malicious data, a ZIP file containing various structured and unstructured files (*e.g.*, plain text, PowerPoint presentations, CSV datasets, PDF files) was extracted and duplicated, leading to $\approx$5 GB of benign data being available in the `/etc` directory of the malicious sandbox. Here, an instance of the ransomware was deployed, and the starting time was noted. After 15 minutes, the experiments were halted. Then, several datasets were curated, leveraging the performance metrics and system calls extracted by *SecBox*. In all cases, data was aggregated into 1-second time slices. For the resource metrics, problematic features were removed, and only four features were used: relative CPU load, absolute RAM usage, and number of packets sent and received. For the system call data, two pre-processing methods were employed, leading to two pre-processed datasets. Both of these datasets hold the same number of features, *i.e.*, the total number of system calls analyzed by *SecBox*. For these 82 features, the first approach simply computes the frequency of each system call being invoked during the 1-second time slice. The second approach computes the sequence of system calls appearing after the system call indicated by the feature, *i.e.*, each feature is represented by a list indicating a sequence of system calls following that invocation. Out of all the trained classifiers and ADs, only the frequency-based system call classifier leveraging the Naive Bayes Model (*cf.*, Table 5.14) demonstrates strong performance

Table 5.14: Model Performance [21]

Model	F_1 **Score**
Frequency-based System Call – Naive Bayes	0.9508
Resource Usage – Decision Tree	Undefined
Sequence-based System Call – LOF	0.6170
Resource Usage – Isolation Forest	0.9142

(*i.e.*, considering $F_1 > 0.95$), with an F_1 score of 95.08 [21]. The visual features of *SecBox* can be used to investigate why other models do not perform well. For example, the visualizations show that the start of the isolated Monti ransomware only leads to a spike in CPU, while the other features remain stable after infection. Thus, the platform helps explain why certain models do not perform well and how features or data quality could contribute to it. A second factor that likely influences the performance of the AD models is the lack of simulations on benign behavior, since the experiments considered only a passive benign Ubuntu image. It is concluded that *SecBox* helps with both manual tasks (*e.g.*, interacting with a sample, exploring features, investigating model performance) and automated workloads (*e.g.*, pre-processing according to standardized pipelines, retraining using pre-defined hyperparameters). Hence, the platform simplifies several concerns but cannot remove the main challenge in behavioral data generation: simulating appropriate behavior. This appears as a key challenge when following such approaches since models ultimately learn the simulated behavior and organizations would need to closely mimic that behavior and continuously monitor and retrain if that behavior were to change in the real assets.

Isolation: Signature-based Detection Due to the size of the project, providing isolated execution, it is infeasible to *prove* the degree of isolation provided. The following two evaluations, therefore, approximate those attributes experimentally.

Since the samples that were executed during all evaluations are known payloads, an experimental approach is leveraged. The Linux-based malware detection tool `maldet` was deployed on the operating system hosting *SecBox*. `maldet` comprises over 10,822 signatures, which are updated on a daily basis [209]. Through automatic notifications, it is assumed that the authors would have been informed if any of the known samples were able to affect the infrastructure.

In the previously introduced evaluations, numerous diverse samples were obtained from real-world collections [4] and executed. This involved botnets, ransomware, rootkits, data stealers, spyware, and cryptojackers. During all experiments, no breaches were detected. Thus, it is assumed that the isolation approach provides protection against escaping and escalation. However, due to the novelty of employing containerized isolation, it may also simply be the case that none of the samples are opti-

Table 5.15: Isolated Runtime Detection Strategies for Evasion [222]

Strategy	*Environment*						
	0	*1*	*2*	*3*	*4*	*5*	*6*
dmidecode_mem	100	0	0	0	0	0	0
dmidecode_sysfs	100	0	0	0	0	0	0
intel_rapl	100	0	0	0	0	-	-
dockerenv	100	0	0	0	0	0	0
meminfo_attributes	100	0	0	0	0	0	0
meminfo_zeros	100	0	0	0	0	0	0
meminfo_typo	100	0	0	0	0	0	0
available_dmi_info	100	0	0	0	0	0	0
proc_combined_idle_time	100	0	0	0	0	0	0
syscall_unimplemented	100	0	0	0	0	0	0.34
kernel_ring_buffer_messages_set	100	0	-	0	-	0	-
kernel_ring_buffer_messages_count	100	0	-	0	-	0	-
kernel_ring_buffer_messages_fixed	100	0	-	0	-	0	-
dev_kmsg	100	0	0	0	0	0	0
syslog_clear	100	0	-	0	-	0	-
Final weighted score	**100**	**0**	**0**	**0**	**0**	**0**	**0.34**

mized for this objective. Nevertheless, this finding and the results of related studies focusing on *gVisor*, in addition to the assumption that *SecBox* is not deployed within critical infrastructure, indicate that the overall risk exposure is acceptable for the functional goal of extracting behavioral data [81].

Visibility: Offensive Testing Opposing the relevance of escapes and elevation attacks, the aspect of visibility (*i.e.*, whether a sample could detect the isolated environment) is potentially more relevant since it directly opposes the main goal: obtaining relevant data. To test this, samples that were not executing with apparent impact on key features were redeployed in a bare-metal machine. None of those non-executing samples were successfully executed in a realistic environment. Again, it must be stressed that none of the techniques may consider detecting this particular environment. This hypothesis was further investigated in [222], where analysis evasion was investigated from an offensive perspective by building a malware that detects the *SecBox* containerization and alters its own behavior. In short, the following are the key findings of the study.

Table 5.15 highlights fifteen detection strategies that were specifically designed, implemented, and tested through a white-box analysis of the isolation approaches provided by the *gVisor* kernel and the Docker runtime. For example, the character device `/dev/mem` is not present by default (except if

mapped explicitly, circumventing the security control). As such, utilities, such as `dmidecode`, cannot access BIOS information by reading the DMI table. While this is used to prevent data leakage to an untrusted container, malware can use the absence of this device to assume it is executing within an isolated environment and obfuscate its behavior. Indeed, as outlined in Table 5.15, the technique constructed out of this security control was successfully used to detect that the malware is running in an environment restricted through *gVisor* and *Docker* (*cf.*, Table 5.16). The remaining six environments differ in privilege level, operating system, kernel version, and CPU configuration. The `dmidecode_mem` strategy detected that those were non-isolated environments. As such, malware could restrict its damage function in the isolated environment, as demonstrated through the CoinMiner adaption [222].

Table 5.16: Test Environments [222]

ID	*Environment*	*OS (Host)*	*Kernel (Host)*	*CPU*
0	gVisor	Debian Bookworm	6.5.0	Intel i5-7300U
1	Native (Privileged)	Debian Bookworm	6.5.0	Intel i5-7300U
2	Native (Unprivileged)	Debian Bookworm	6.5.0	Intel i5-7300U
3	Native (Privileged)	Artix	6.9.7	Intel i5-8350U
4	Native (Unprivileged	Artix	6.9.7	Intel i5-8350U
5	Native (Privileged)	Debian Stretch	4.9.0	AMD Athlon G. 3150U
6	Native (Unprivileged)	Debian Stretch	4.9.0	AMD Athlon G. 3150U

Similar results were achieved for the remaining strategies highlighted in the table. These exploited a variety of data sources, including BIOS information, kernel and driver logs, power management information, docker environment variables, and init systems, memory, file systems, or restriction of system calls. In summary, those varied techniques present several directions on how malware could evade *SecBox* and obfuscate its behavior. Assuming that the platform was to be used to, for example, train models that should accurately detect attacks in critical systems, this would lead to devastating effects. However, as demonstrated through the prototypical improvements to the isolation mechanisms presented in [222], many of those offensive approaches could be mitigated through obfuscation on the defensive side. For example, the previously mentioned character device could be simulated. Out of the proposed techniques, the authors assume that the one assessing unimplemented (or restricted) system calls is the most challenging since it would require that each of the missing system calls is implemented in a way that resembles a native implementation. However, this would only apply to samples that assume privileged access [222].

5.3.3 Key Observations and Discussion

From the development and testing of *SecBox* through several design studies, the following observations are discussed:

1. *SecBox* positions containerization as an alternative isolation strategy to the usage of Virtual Machines. By doing so, dynamic malware execution can be done on low-cost environments (*e.g.*, cheap VPS or low-cost hardware) and in environments where this degree of isolation is not desired or possible (*e.g.*, reproducibly creating datasets in IoT devices without virtualization support or within nested virtualization).

2. The platform specifically targets behavioral features, opposing static features in datasets analyzing malware samples. As demonstrated, interactive and informative analyses can be done, and rich datasets can be extracted from doing it, which can be used for other tasks (*e.g.*, model explainability, training classifiers).

3. Understanding clearly the security guarantees provided by the isolation layer in *SecBox* would require further investigation. Due to the size of the project and the tests performed, it is suggested to use the platform in another decoupled component (*e.g.*, isolated physical machine or Virtual Machine). Furthermore, as demonstrated by offensive experiments, a sample explicitly targeting the platform could evade analysis – specific improvements to avoid invasion have been proposed in [222].

4. Dynamic malware analysis still comprises several human-in-the-loop activities, such as the replication of an environment or simulating the interaction with a sample. *SecBox* addresses these factors through increased automation and by considering reproducibility during all stages of the data and ML training pipeline.

5. Closely related to the last observation, defining malicious and benign behavior is likely the key challenge of using behavioral data for security purposes. This includes not just the interaction with a malicious sample but also the simulation of benign behavior, which is likely subject to change over time. Additional studies would be needed to understand how closely behaviors are replicated in this isolated environment. After all, *SecBox* yields a more secure way to extract data from malicious samples than gathering on-device at the cost of losing some degree of similarity.

In summary, *SecBox* contributes to behavioral detection with an efficient, reproducible, guided, and highly automated approach to extracting behavioral data. The key limitations are open security

concerns and the degradation of data when transferring from a real device to the sandbox environment. Future studies could investigate how strong this tradeoff is. Then, this gap could be bridged by deploying the sandboxing module of *SecBox* – which is completely detached from the remaining components – in different hardware configurations.

5.4 File System Defense

Compared to the two MTD studies introduced, the design study on *SecBox* questions whether the data used for reactivity is aligned with the threat model, *i.e.*, whether data collected on-device is trustworthy enough to inform a defense on it. This observation, in addition to the observations in Section 5.1 highlighting the inefficiency of proactive defenses, is congruent with the limitation *L8)* synthesized from the literature. In the following, a file-system-based approach is introduced that adopts a particular threat model and investigates whether this area of the attack surface can be virtualized. From this file system-based viewpoint, both proactive and reactive MTD approaches are evaluated to investigate whether a partially virtualized attack surface can make proactive MTD efficient enough. Furthermore, the studies assess the merits of a detection system that focuses only on features that can be collected securely. The merits of detection are assessed from a defense perspective, *i.e.*, not only by testing models in a lab environment.

For the development of these file system approaches, the assumed threat model is the following: a malware-based attack has breached an initial line of defense (or even a secondary one, such as another reactive MTD technique that is not correctly detecting the behavior). The sample is assumed to have full user-space access for a given non-root user. For example, a web app running as a regular user could be breached, and subsequently, ransomware could encrypt all files accessible by the application.

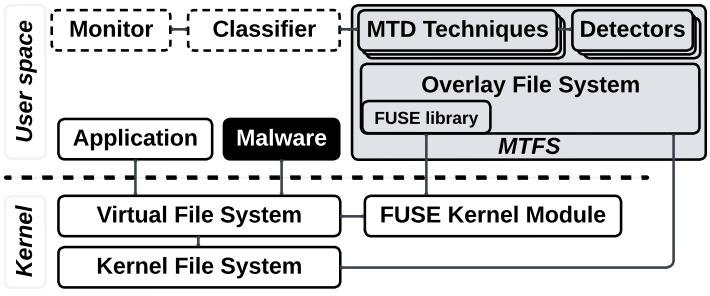

Figure 5.16: *MTFS* Architecture and Prototypical Design [268]

5.4.1 File System and Testbed Architecture

The development studies proposing the proactive *MTFS* approach [1] and the reactive *GuardFS* approach focus on malware comprising a file-based damage function and use crypto-ransomware as a key threat, represented by multiple samples. Figure 5.16 highlights that the architecture closely follows the architecture of file systems for the Linux kernel, abstracted through the Virtual File System. Essentially, the approach presents a file system that enables two aspects: monitoring interactions with the file system through the kernel and the interception of system call responses – these aspects are then used by three proactive [268] and four reactive [272] techniques.

To provide a prototypical implementation of the file system, *MTFS* leverages the FUSE library to receive and intercept system calls issued by user-space applications (*i.e.*, malware or benign software). For the prototype, *MTFS* acts as an overlay file system for data stored in another file system (*e.g.*, ext4, FAT). Thus, *MTFS* sits between the user-space process and the actual data, enabling monitoring of requests and interception of invocations. A realistic implementation could move all those components to kernel space for improved protection and performance. However, for the prototype, the FUSE-based implementation presents a minimal yet fully functional file system. The implementation in user space would allow easy integration with other applications, such as ML-based classifiers. More specifically, *MTFS* implements the overlay file system using the `go-fuse` library. The file system and the three MTD techniques are all implemented in the *Go* programming language. Each MTD technique defines a set of system call hooks it would like to intercept. For example, an MTD technique could indicate that it wants to act as a middleware for `WRITE` calls. Then, it could modify any of the parameters (*e.g.*, file path, write buffer) and dispatch a new system call to the underlay [268].

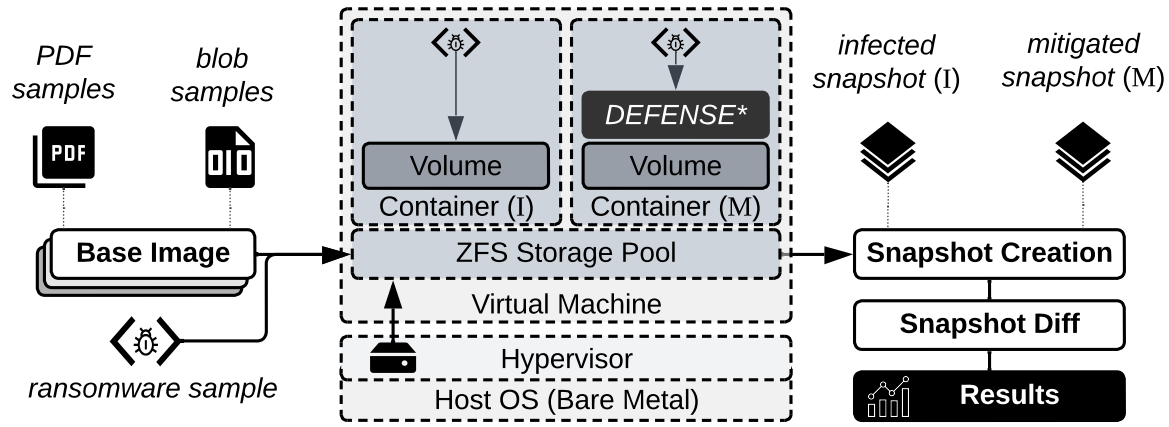

Figure 5.17: Ransomware Testbed Architecture [268]

[1] **M**oving **T**arget **F**ile **S**ystem

Both the reactive and proactive techniques are tested against real and dynamically executed (*i.e.*, not simulated) malware samples. As previously argued, dynamic execution of such samples requires consideration of their isolation, interaction, and observability. Figure 5.17 proposes a means of how ransomware samples can be executed, preserving those objectives. The input is represented by the ransomware samples (*e.g.*, Linux ELF files, Bash scripts) and the benign user data to be mounted in the underlay. Those are stored in a ZFS storage pool to enable quick and straightforward snapshot creation before and after experiments. Each sample can be tested in two ways. First, an "uninterrupted" (*i.e.*, no defense being enabled during the experiment) sample is placed into an isolated container running in a VM. Within the container, the sample can directly access the user data. In the "interrupted" scenarios, the file system and a particular technique are deployed. In both cases, the following data is extracted from an experiment: performance metrics are recorded through the Docker API, and changes made to benign data are assessed by creating a new snapshot and comparing it to the old snapshot. Snapshots are then compared based on a pessimistic approach: For each file in the original snapshot, it is assessed whether it was modified, renamed, or deleted. If any of these operations took place, the complete size of the initial file is considered a data loss, hence the consideration of this approach being pessimistic – even when merely the file suffix is modified or a single bit in the file is modified. With those two measures, the efficiency (*i.e.*, how many resources the ransomware or defense consumes) and effectiveness (*i.e.*, how much data was lost) are reported [268].

In [265], an implementation of the testbed is provided. All components of the testbed can be instantiated in a VM with a single command. The following evaluations were executed in an isolated Ubuntu 22.04 VM. In terms of hardware, an AMD Ryzen 5700G processor (4.7 GHz) is used. The storage media is likely essential to avoid distorting results, *e.g.*, a slow disk could lead to low encryption speed, distorting the number of bytes lost. The ZFS pool shown in Figure 5.17 uses two NVMe storage devices (WesternDigital SN550 NVMe SSD) in a RAIDz1 configuration. Based on this configuration, all 20,741 benign files (10,460 MB) that were obtained from [95] could be read in 9 seconds (≈10 Gbit/s) by executing a simple bash command (*i.e.*, `time cat /data/**/*`). Concerning write speed, the files could be overwritten with an equal number of random bytes in 27 seconds (≈3.2 Gbit/s). Regarding malicious payloads, the source code and binaries of 24 ransomware families were collected, and their interfaces were harmonized through a deployment script. Only 13 samples were successfully executed, which led to damage. For example, some samples targeted specific technologies, such as volumes from the *ESXi* hypervisor, which were not replicated. The sample's implementations include various languages (*e.g.*, Python, Java, C, Go) and formats (*e.g.*, sources, ELF files, scripts). Furthermore, the samples contain both prototypical samples and samples pertaining to known families (*e.g.*, Lockbit, Blackbasta) [268].

5.4.2 Proactive Techniques

MTFS integrates three MTD techniques. The first technique `MTD_DELAY` simply delays the forwarding of system calls to the underlay. Different durations can be specified to delay the attacker. However, no active modifications to the system calls (*i.e.*, *OPEN, CLOSE, READ, WRITE, STAT, RENAME*) are made. This technique, by itself, is unlikely to mitigate an attacker. However, it closely aligns with the MTD paradigm, increasing the mean time to compromise (*e.g.*, discover all files for encryption, delete a file) [268].

Secondly, `MTD_INF` delves into deceptive defense approaches. This technique exploits the previously described directory traversal employed by numerous ransomware samples. Hence, it provides an *Infinite Directory Graph*, similar to the one described in Section 5.1. However, it does so directly on the file system abstraction, avoiding the creation of real files. When a sample enumerates the items in a specific directory, the technique shuffles the output, including real paths and a self-linking trap directory. If the files in this trap directory are listed, files from the underlay and another trap directory are presented, leading to an infinite structure that traps a depth-first traversal [268].

The third and final technique `MTD_SUFFIX` not only adds deceptive information to real data (*i.e.*, the contents of a directory) but also alters the metadata of existing files. Through this, it provides an on-the-fly implementation of the file obfuscation approach. As such, it obfuscates not only the suffixes contained in the file path but also the initial bytes of a file's content. Hence, an attacker cannot use those "magic numbers" to infer the actual file type. This technique targets the commonly observed approach of focusing on (or excluding) particular file types [268].

5.4.3 Reactive Techniques

To implement and validate a set of reactive techniques under the term *GuardFS*, a detection plane is implemented using the baseline architecture of *MTFS*. The detection system developed herein aims to detect damage *before* it is done. Section 5.1 presented a detection mechanism looking at kernel events. However, this signal essentially detects *ongoing* damage. Since the *MTFS* overlay system "sits" between the application and the underlay file system, each invocation can be individually analyzed and handled. To implement this detection as granular (*i.e.*, distinguishing between malicious and benign processes) and early (*i.e.*, detecting loss before it occurs), system calls appear as an appropriate data source since related research established their usefulness for detection [272].

Monitoring Although the interface to a file system is conceptually simple, numerous types of system calls can be analyzed from the viewpoint of a file system. The first feature for detection is the type of the call (*e.g.*, `MKDIR`, `OPEN`). Depending on the system call, additional data is passed as

an argument. For example, the suffix of the file path can be extracted from the `open()` system call. Furthermore, flags, such as the mode (*e.g.*, `O_APPEND`, `O_ASYNC`), could be extracted. Other sources likely require more computational power to process. Most prominently, `WRITE` calls pass the new buffer to be written. For the detection system, the entropy of the buffers is computed, as described by Equation 5.2, and used as a feature [272].

$$H = -K \sum_{i=1}^{m} p_i log(p_i) \tag{5.2}$$

To obtain data for model training, three ransomware samples were executed, *i.e.*, *RansomwarePoC*, *DarkRadiation*, and *roar*. Hence, the stealthy sample, previously described through the *RansomAI* experiment, is included, too. Each sample had direct shell access. To represent non-trivial benign behavior, a workload comprising numerous read-and-write operations with diverse user data was created. An FTP server was set up on a Raspberry Pi to control the benign behavior. The file system operations were induced using the JMeter stress testing suite [249], configuring multiple parallel clients that browse the files obtained from [95]. Furthermore, the client uploaded structured and unstructured data (*e.g.*, ZIP files, images). Data was collected over 258 minutes. The *roar* ransomware (*cf.*, Section 5.1) did not fully encrypt benign data due to its stealthy nature in the timeframe and was terminated after two hours [272].

The datasets representing the different scenarios (*i.e.*, benign, *RansomwarePoC*, *DarkRadiation*, *roar*) store one timestamped row per system call observed. These were aggregated into time slices (*i.e.*, summarizing per-process behavior over 2, 5, and 10 seconds). Time-sensitive and dangerous features (*e.g.*, timestamps, process IDs) were removed for model training. Models were then trained on 80% of the data, using two approaches. First, a single global model is trained based on an aggregated dataset, where all malicious behaviors are labeled as malicious (*i.e.*, a binary classifier). The second variation trained a model for each time slice configuration but always left out one sample to assess the ability of the model to detect unseen instances of the same threat. AD models leveraged benign data only. Commonly used models, such as Random Forest Classifier, Logistic Regression, and Isolation Forest, were tested for training [272].

Defense Techniques Several defense techniques are developed under the design assumption that the monitors can inform the techniques of malicious operation. As such, they assume that the file path and PID of an offending process are available.

The first technique `PKILL` is not necessarily novel from the research perspective. It is implemented to compare the remaining techniques that follow the deceptive and MTD paradigm more substan-

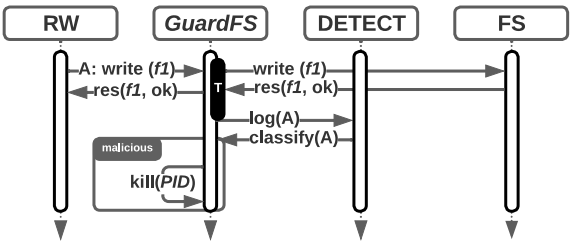

Figure 5.18: Baseline Defense Through Process Termination [272]

tially. Effectively, Figure 5.18 demonstrates that the overall flow of the technique involves three stages. The malware starts accessing the file system through *GuardFS*. Operations are forwarded to the underlay and used for detection. After the detection time (*i.e.*, depending on monitoring delay and the recall of the model), this technique would terminate the process based on its PID. Presumably, this leads to initial data loss, but it may not hinder subsequent infections. In addition, it may present an explicit signal to the attacker to alter its behavior [272].

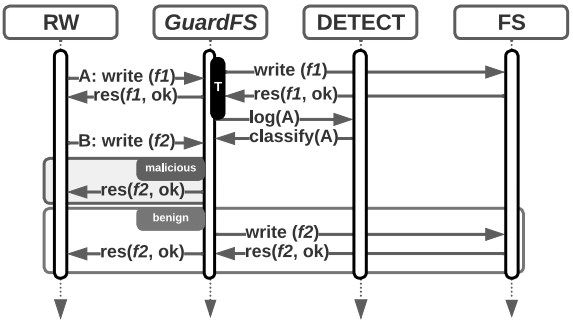

Figure 5.19: Defense Through Obfuscation [272]

The second technique explores a novel way to respond to malicious behavior more stealthily than PKILL. The basic operation of the OBF technique is visualized in Figure 5.19. File system requests from unknown processes are redirected to the underlay while monitoring this behavior. At some point, the detection system would flag the behavior as malicious, invoking the defense by passing the PID of the calling process. At that point, the PID is used to imitate interaction with the underlay. However, no actual operations are forwarded to the underlay, and responses are simulated for specific system calls (*e.g.*, `rename()`, `unlink()`). For example, to obfuscate a `write()` system call, the system waits for a certain amount of time and yields an integer indicating the number of bytes that the client wants to write, providing the illusion of a successful write operation [272].

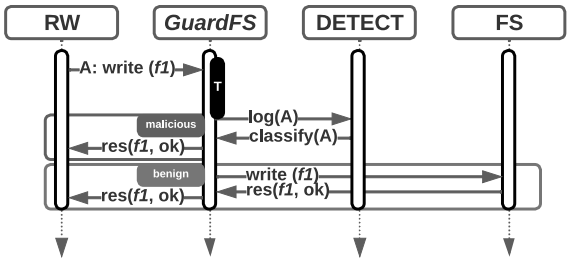

Figure 5.20: Defense Through Delaying and Obfuscation [272]

The previously discussed technique applies obfuscation only after an initial detection phase. During this stage, an adversary is uninterrupted. DEL+OBF is designed for scenarios with high sensitivity to data loss while sacrificing latency. As indicated in Figure 5.20. The technique applies the same mitigation post-detection. However, the initial loss is avoided by buffering the system calls to be forwarded to the underlay until the detection system classifies them as benign. Hence, latency and buffering requirements and constraints arise [272].

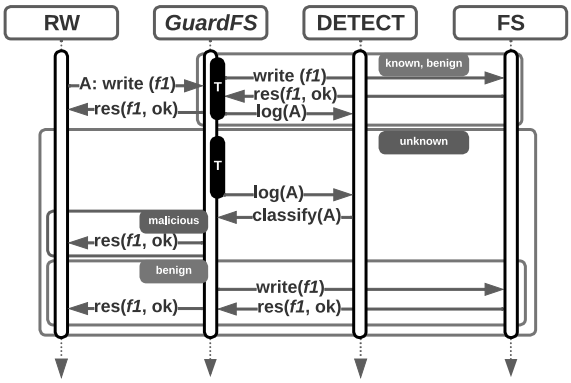

Figure 5.21: Defense Involving Tracking of Processes [272]

DEL+OBF arguably provides stronger security guarantees by sacrificing performance. The last technique is shown in Figure 5.21. TRACK discriminates known and unknown processes. Unknown processes are subjected to the same behavior as in DEL+OBF, *i.e.*, they are buffered until known to be benign. PIDs classified as benign are tracked over a longer time and treated as benign. However, although their execution is forwarded to the underlay, they are continuously analyzed. Hence, a sample pretending to be benign but turning to malicious behavior would be detected. Nevertheless, other benign and long-running processes would benefit from the improved latency since their execution is not buffered [272].

5.4.4 Evaluations

The following experiments are done on the two respective systems, leveraging *MTFS* for proactive defense and *GuardFS* for reactive defense.

Proactive Defense

The proactive techniques [268] are tested in a real-world scenario (*i.e.*, a productive *ElectroSense* sensor) and in the previously introduced testbed.

Sensor Experiment For the sensing scenario, a Raspberry Pi 4 Model B with a 64 GB class 10 SD card was used. 1000 files from [95] were deployed to `/home/pi`. Then, the effect of introducing *DarkRadiation* and *RansomwarePoC* was studied for four scenarios each: using the normal *ext4* file system and when mounting one of the three proactive techniques. As indicated by the results in Table 5.17, both samples achieved to encrypt all files when no defense is present. *DarkRadiation* achieves full encryption significantly faster. When leveraging the described proactive techniques, *RansomwarePoC* is unable to cause any data loss. Surprisingly, this is even the case when accessing files that are simply delayed through `MTD_DELAY`, indicating that the sample is unlikely highly optimized, as the sample's execution halts for `MTD_SUFFIX` and `MTD_DELAY`. *DarkRadiation* demonstrates stronger offensive performance. For `MTD_INF`, it does manage to cause data loss. However, the technique is successful at delaying this goal, introducing a delay at a factor of ≈25. Furthermore, the other techniques successfully protect against data loss [268].

Table 5.17: Defense Effectiveness in Sensor Node [268]

Ransomware	***Implementation***	***Loss***	***Duration***
RansomPoC	Default file system	946/979	7m 51s
	`MTD_INF`	0	10m 32s
	`MTD_SUFFIX`	0	4s
	`MTD_DELAY`	0	4s
DarkRadiation	Default file system	976/979	1m 14s
	`MTD_INF`	0	31m
	`MTD_SUFFIX`	0	1m 50s
	`MTD_DELAY`	976/979	15m 7s

Controlled Experiment To assess the performance with a greater number of samples than in the real sensor, the virtualized testbed is employed. Figure 5.22 presents the results when testing the *MTFS* file system with the `MTD_INF` trap directory being enabled in `/home/USER/!`. In contrast, the data loss when no defense is present is indicated, too. As highlighted, the defense effectively protects benign data, protecting the majority of data across all tested samples. Although the sample was successfully mitigated in the *ElectroSense* scenario, *RansomwarePoC* was the sample with the highest data loss. Upon deeper manual investigation, a potential explanation for this result is the layout of benign files and the exploration strategy of the sample (*i.e.*, it began encrypting files in other, randomly accessed paths without completely establishing all target files before starting encryption). Across all samples, the presence of the defense technique resulted in preventing 97.01% of data loss. It is argued that this technique can be used as a honeypot for detecting or directly as a defensive measure, preventing data loss, depending on the risk sensitivity. Furthermore, the experiments using the snapshot yielded additional insights into the implementation differences of samples. For example, samples differ in encryption speed, exploration strategy, and encryption approach (*e.g.*, some overwrite the file content, while others write to a new file and rename the file). From a threat-oriented perspective, this does indicate that subtle differences in those threats' approaches can have an effect on the effectiveness of a generically designed defense system [268].

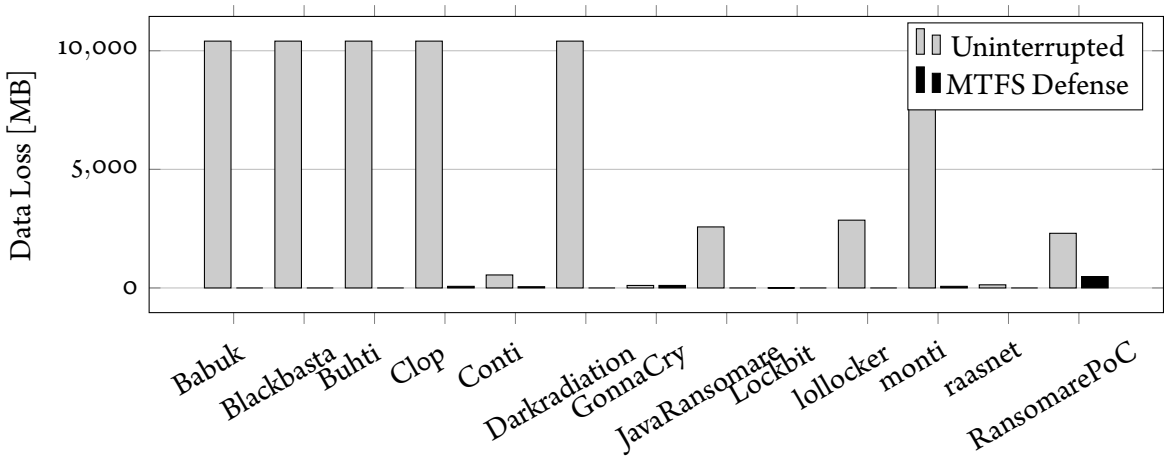

Figure 5.22: Data Loss in Testbed [268]

In a second experiment, only benign behaviors are executed to assess the overhead (*i.e.*, the efficiency of the defense during normal operation). Although cyber attacks are relevant to mitigate – efficient normal operation is highly relevant since it will constitute the large majority of the device's lifetime. Three workloads were tested: *(i)* installing a web server, *(ii)* creating a `tar` archive of user

files, *(iii)* and continuously persisting sensor data from a web service. Each workload is executed directly on the underlay and with the *MTFS* overlay. Since none of the workloads are trapped by the technique, no significant delay is measured (*i.e.*, leading to differences in the task completion time of +0.12%, +0.45%, and -4.4%). Furthermore, no consistent increase in CPU consumption is observed when running *MTFS*. Memory consumption is increased by less than 0.2% [268].

Reactive Defense

In the following, the transition to a reactive, ML-based defense (*cf.*, Section 5.4.3) is tested by deploying *GuardFS*. Then, the results obtained from proactive and reactive defense are compared. First, the performance of the detection plane is assessed by testing the developed models in two tests: using the remaining 20% of test data and by relying on deploying the models to a real sensor node and testing the detection of infections in an online manner. After establishing how capable the detection system is, the overall defense, based on those models, is tested. For this, the defense effectiveness is established by measuring the prevention of data loss in the testbed. Finally, efficiency is assessed by measuring the time and resource overhead of benign operations.

Table 5.18: RF Classifier Performance: Detecting Unseen Samples [272]

Time Window	*RansomwarePoC*	*DarkRadiation*	*roar*
2s	99.92%	94.71%	99.43%
5s	99.91%	95.55%	99.65%
10s	99.90%	96.92%	100%

Ransomare Detection Multiple models were trained – leveraging different algorithms and exposure to different behaviors. Table 5.18 indicates that the performance improves with larger time windows. However, the increase in performance between five and ten-second time windows would have to be gauged by assessing the data loss that occurred during the additional three seconds. More surprisingly, all samples can be detected for the larger time windows, even though the models were not trained on data obtained from those samples. It is argued that detection of a generic ransomware threat may be feasible [272].

Due to the performance achieved when detecting unseen samples, a generic model is trained using all three samples. Table 5.19 highlights a similar trend when evaluating them: performance increases with larger time windows. Based on the two tests, the RF-based model trained on all three samples with an interval of 5 seconds is used for deployment [272].

Table 5.19: Accuracy of LF and RF Classifiers [272]

Algorithm	*2s*	*5s*	*10s*
Logistic Regression	99.53%	99.76%	99.87%
Random Forest Classifier	99.93%	99.97%	99.98%

In the last test isolating model performance, the model that performed well in a lab setting is deployed to the sensing use case, *i.e.*, to the same scenario where the data was initially collected. Then, it was measured how long it took until the execution of the sample was successfully detected, as presented in Table 5.20. For each sample, the minimum, maximum, and average detection time is collected over multiple rounds. Based on the average duration, the data loss of each sample is illustrated based on the encryption speed. This highlights that it may be sensible to test ML-based security applications not only from a pure detection perspective (*i.e.*, reporting common metrics when testing within a collected dataset). Instead, such metrics can be put in perspective by measuring and thereby contextualizing the impacts of a defense scenario. In this test, it is apparent that stealthy samples such as *roar* take longer to detect. However, their stealthiness is achieved by adapting – among other aspects – the encryption speed. If assessed through overall data loss, *roar* is outperformed by more aggressive samples [272].

Table 5.20: Testing Detection in Sensor [272]

Sample	*Min.*	*Avg.*	*Max.*	*Data Loss*
RansomwarePoC	4 s	6 s	10 s	18 Files – 12.3 KB
DarkRadiation	4 s	8 s	10 s	35 Files – 23.9 KB
Roar	8 s	19 s	44 s	3 Files – 2.1 KB

Ransomare Mitigation Using the ransomware testbed and the pessimistic loss calculation method, each sample is tested in a dedicated experiment with two runs per defense mechanism (*i.e.*, without interruption and with the reactive defense enabled). Each experiment lasts for a maximum of five minutes. Table 5.21 describes that without any defense enabled, most samples achieve full encryption [272].

Table 5.21: Data Loss (in KB) Across Defense Techniques [272]

Defense	NO DEF.	PKILL	OBF	DEL+OBF	DEL+OBF	DEL+OBF	TRACK
Delay	0	0	0	1 s	5 s	10 s	$<$ 5 s
Babuk	10'655'744	107'752	77'172	1'286	1'289	1'286	27'025
Blackbasta	10'655'744	9'168	9'168	8'266	4'313	4'313	8'781
Clop	10'655'744	668'159	70'644	19'598	0	387	0
conti	561'152	49'255	96'371	69	0	19	42'427
GoCry	10'655'744	0	0	0	0	0	0
javaRans.	2'634'752	39'762	48'500	0	0	0	12'149
lollocker	2'926'592	620'125	137'304	1'692	19	19	29'777
Monti	10'655'744	28'889	88'259	69	0	10	19'489
Average	7'425'152	190'388	65'931	3'873	703	754	17'456

Evidently, all infections benefit from the simplistic reactive defense using the PKILL strategy, highlighting a reduction of data loss to only 3% of the uninterrupted case. Still, in absolute terms, this refers to $\approx$ 200 MB of data being lost on average since the sample encrypts uninterrupted until the detection is successful, motivating either a more timely detection or the usage of preventive measures, as envisioned by the other techniques. OBF further improves the already strongly reduced data loss observed when using PKILL. Compared to this defense, another $\approx$65% reduction of data loss is found. When investigating the logs, it can be seen that the ransomware is indeed continuously trying to attempt encryption. This method is also potentially more stealthy (*i.e.*, lacking an obvious signal to the attacker). Further improving loss, DEL+OBF variants all lead to only minimal losses, with losses decreasing and then stabilizing for the 5-second duration. Still, some data is lost since the models do not perfectly detect every system call. TRACK was designed to provide better usability than the delaying defenses. This can be observed in the experiments as more data is lost compared to DEL+OBF. For example, on average, TRACK loses $\approx$17 MB of data compared to $<$ 1 MB lost by DEL+OBF in the 5-second variation. All techniques present an improvement over not enacting any autonomous defense while stealthily mitigating damage. Furthermore, the different techniques present varying degrees of security and usability [272].

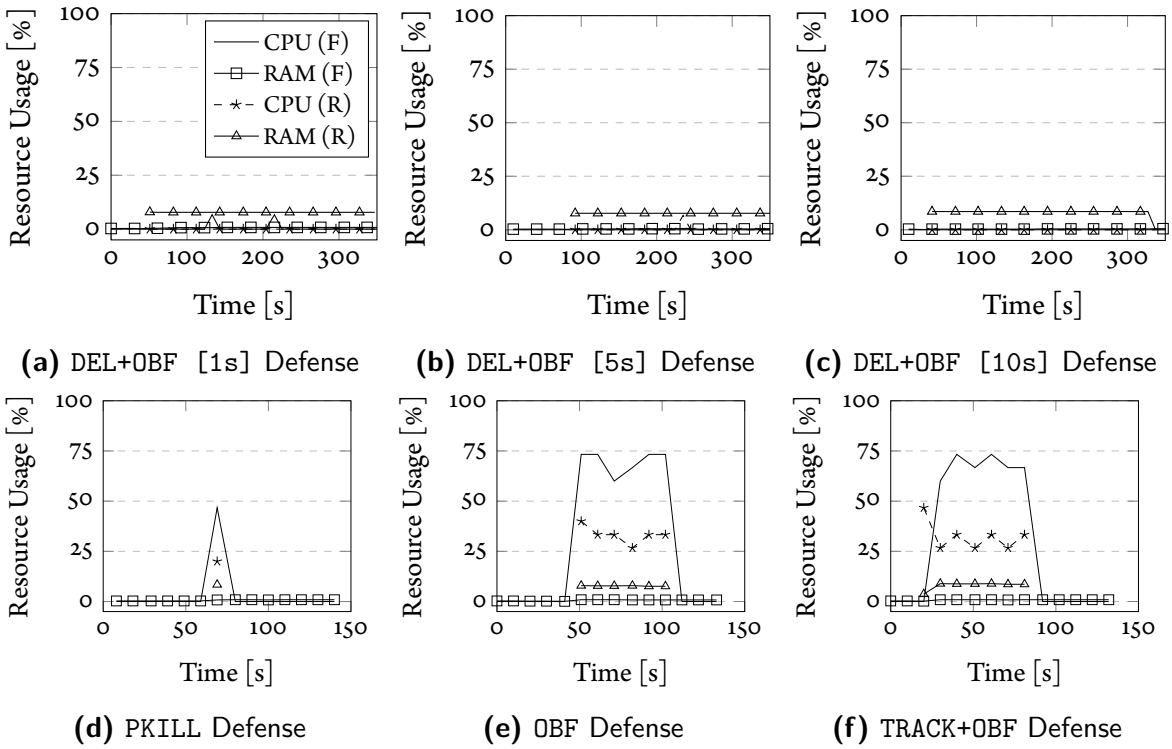

Figure 5.23: Performance Analysis During Defense [272]

The aspect of providing stealthy defense must also be analyzed from an efficiency perspective. After all, resource-sensitive scenarios would not benefit from an attacker encrypting random data over an extended period of time for the sake of the adversary remaining in the unknown. Figure 5.23 plots the relative resource consumption of the file system (denoted by F) and of the encrypting process and its subprocesses (denoted by R). Taking the PKILL defense as a baseline, a short burst of increased consumption can be observed for both defense and offense. From a resource perspective, this is the most efficient approach when considering total load and when accepting short bursty resource consumption. Due to their buffering approach, the DEL+OBF defenses smooth resource consumption over time: the encryptor remains encrypting random bytes for as long as there is input data to be encrypted. OBF does not delay the encryptor. Thus, the same amount of resources is consumed in a shorter time. Since the overall number of resources spent is comparable, a defender would need to gauge how relevant the bursty nature of this impact is. The same holds true for TRACK since it only delays unknown processes – processes classified as malicious encrypt random bytes without any rate limiting. PKILL presents, in theory, the optimal efficiency. However, assuming advances in self-adapting ransomware, subsequent encryption attempts could be made, which would waste additional resources. Regarding the remaining defense mechanisms, operators would need to discuss the

sensitivity to bursty workloads. The final observation is that, in passive and benign scenarios, the file system requires few resources, as will be explicitly measured in the next experiment [272].

Benign Execution As argued before, resource usage and usability degradation due to the defense must be assessed since devices presumably spend the majority of their lifetime in a non-infected state. This experiment tests the TRACK defense to assess the tradeoff implied by the previously presented security guarantees. Similar to the proactive scenario, four benign workloads (*i.e.*, WL1 to WL4) are executed in the testbed – once with the reactive defense system absent and once enabled. WL1 installs a series of Linux packages (*i.e.*, apache2, BIND9, MariaDB), which are downloaded over HTTPS and then installed using the dpkg-deb CLI utility. Both the raw packages and the installation occur in the target file system. WL2 fetches a reading from a web-based sensor service and stores the value on disk. WL3 creates a gzip-based backup of 977 protected files. WL4 runs for a longer duration, and a large amount of data is written: the full LibreOffice software suite is downloaded to the file system. To control for network latencies, it is fetched over a local 1 Gbit/s Ethernet connection [272].

Table 5.22: Usability Impact and Resource Consumption During Normal Operation [272]

Scenario	***Time*** [s]	***CPU*** [%]		***RAM*** [MB]	
		System	FS	System	FS
WL1	3	19.7	-	2.6	-
WL1+TRACK+OBF	8	13.4	56.5	1.8	10.8
WL2	4	1	-	4.8	-
WL2+TRACK+OBF	6	1	1	4.9	14.1
WL3	16	51.1	-	13.43	-
WL3+TRACK+OBF	20	33.33	1	1.9	12.1
WL4	23	9.8	-	10.5	-
WL4+TRACK+OBF	24	2.24	50.1	2.25	7.28

Table 5.22 indicates for each scenario the task completion time as well as the CPU load and memory consumption. Resource consumption is measured and reported for the file system and the remaining user space. Memory consumption is stable across all tasks, requiring only several megabytes of data to run the defense system in the background. However, it can be observed that the execution time is delayed in different ways. For long-running tasks, such as WL4, this additional delay of $\approx$4% appears negligible compared to short-running tasks like the sensing service in WL2, which is delayed by 50%. Thus, there is clearly a tradeoff between security and latency for short-lived tasks. However,

one would need to consider whether this latency is critical – if only the long-term persistence of regular polls (*i.e.*, repeating WL2) matters, such a delay could be accepted in favor of increased resilience. Lastly, as can be seen from the CPU consumption of the file system, heavy file operation leads to significant process requirements for the system. This is likely where the presented lightweight proactive scenarios outperform the reactive ones: due to the constant processing of the benign activity, resources are wasted in times of benign execution. Hence, increased CPU consumption is likely the key drawback for intensive, long-running processes. Fortunately, no false deployments were discovered during the experiments, and thus, no benign data was lost [272].

5.4.5 Key Observations and Discussion

The previous studies implemented and evaluated several defense techniques following the threat model of an overlay file system faced with a user-space infection. The defense experiments orchestrated in proactive and reactive modes are summarized by the following observations:

1. The implemented defense techniques prevent data loss to a strong degree. The proactive defense system does so in a more efficient manner than the baseline described in Section 5.1. The reactive, preventive instance further decreases the remaining data loss. Still, every scenario sees some data being lost, stressing that reactive scenarios with well-performing models cannot completely prevent losses.

2. Overhead and usability impairments were observed during the active mitigation of infections and during benign behavior. Realistic scenarios need to perform analyses of defense scenarios according to the threat model and the sensitivity of benign execution to those impairments.

3. The reactive system demonstrates that even when basing detection on a safe area of the system, the resulting models are useful when evaluated in a defense context. However, from a technical perspective, the approach relying on a subset of file-system-related system calls requires significant data processing in both benign and malicious cases.

4. Such reactive systems would also require careful monitoring and re-assessment. In the experiments, a static threat model was considered. Although it was demonstrated that a detection model is applicable to several instances of this threat, future instances may diverge. Hence, the concept of employing threat models to design detection and mitigation systems arises as both a challenge and solution: threat models can provide a stable formalization of what can be generically detected without focusing on individual implementation specifics. However, since threat models are, ultimately, abstractions, their applicability and stability must be tested.

Overall, based on these experiments, it is suggested to carefully examine the tradeoff between defense resilience and efficiency. To do so, the approach employed in these design studies argues that this should be done by implementing a complete pipeline, where detection systems are not just tested on lab data but used for a defense scenario to discuss whether the design assumptions are appropriate while leading to measurable defense improvements.

5.5 Conceptual Integration

The aforementioned solutions are integrated by the conceptual framework presented in Figure 5.24. In the first stage, a threat model is always required since it defines the goal and assumptions of the defense. For reactive defense, the collection of specific attack samples and accurate simulations of benign behavior is beneficial. In the second stage, techniques are designed and implemented to meet the threats in the model, resulting in a set of operational techniques and data about their defense performance and the impact they yield on normal operation. This data is critical if the defense system is to be deployed reactively, *i.e.*, when the detection stage is incorporated. Here, data is obtained, and models (*i.e.*, classifiers and anomaly detectors) are built. Importantly, the data must consider the impact of the defense system – at runtime, both the defense and normal behavior are executed. In the last stage, the execution of defense techniques is orchestrated proactively or reactively by incorporating the detection system.

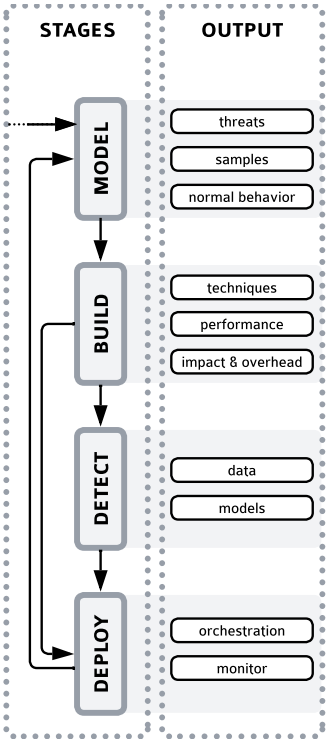

Figure 5.24: Overview

As outlined by the previous studies, several approaches can be leveraged for the respective stages. For example, detection and integration can be done using a binary classifier to analyze the system's behavior. Alternatively, a similar system could be built with pre-defined rules that invoke a specific technique. The decision to select an approach for defense must be aligned with the initial stage and key limitation of the defense paradigm: the underlying threat model. The initial threat model influences the viability of these approaches (*e.g.*, can a human operator define policies for a model incorporating many attacks?). Furthermore, the staticity of the underlying model implies that the monitoring stage must provide data on the applicability of the model, which is, ultimately, a static input parameter. Any changes to this model then require that each stage be executed again.

Some people think I am an alien. (..)
Perhaps it is because, as researchers, we exist somewhere other than the "real world." We speak of "real world" case studies, grounding our theory in the "real world," doing our fieldwork out in the "real world."

— Richard Baskerville (2008)

6

Deployments and Pilots

The previous two chapters closely followed the conceptual framework of design research laid out in Section 2.4. That is, problems were identified for a particular domain, and limitations of existing solutions were assessed using literature and dynamic analysis, motivating the design and evaluation of new artifacts with various methodologies. Although the term "real world" must be used cautiously – as exemplified by the above quote – several of these designed artifacts were evaluated close to an instance of a problem domain. This not only grounded the relevance of those artifacts or illustrated their applicability. Instead, using qualitative methods allows for discovering new strengths, weaknesses, and related problems in those problem domains. Specifically, the following artifacts were deployed into more realistic scenarios (*i.e.*, cases where little control was enacted over their usage):

1. *CoReTM* was employed in a participatory manner to create a threat model of an e-Government information system using only limited information. When doing so, the functions providing flexible diagramming while having basic automation (*e.g.*, summarizing assets, compiling threats) were perceived as most useful. Since the experts were already familiar with several methodologies, the guides were only used supplementally.

2. When performing a threat impact analysis with *QuantTM*, it was discovered that domain experts relied mainly on estimates and did not hold historical data. In contrast, there was historical data to gauge technical aspects (*e.g.*, attack occurrence, attack success). Furthermore, technical representatives were unaware of the effects and magnitude of technical impacts as perceived by domain withexperts.

3. Insider attacks were modeled for an IT service provider. Due to the resilient posture of the provider, several threats proposed by the artifact were validated. However, this scenario also outlined that threat analysis is more challenging when designers act as third parties and when a system is operated in a non-commercial manner. For example, as later illustrated using the postal voting scenario, quantifying attacks on such a system is challenging.

4. Deploying the *ThreatFinderAI* tool further highlighted the difficulty of publishing data on risk sensitivity and exposure against threats, which further challenges research on approaches such as *QuantTM*. For example, if many threat models of real systems and historical attack or impact data were available, more advanced data-driven methods could likely be used, moving away from estimates.

5. *ElectroSense* served as a test scenario for artifacts leveraging detection and mitigation. While techniques were successfully applied to the case, the experiments demonstrate the fragility of such results. For example, shifts in the normal operation or the threat model of the scenario would inevitably impact the effectiveness and efficiency of the techniques. Nevertheless, it is argued that deployment in such a realistic scenario is necessary to ensure the relevance and performance of detection and defense studies.

In addition to these observations from deploying the designed artifacts (*i.e.*, a researcher executing them in a controlled instance of a problem domain), several pilots (*i.e.*, productive solutions that are directly implemented and validated within a problem domain) were developed in cooperation with industry practitioners. These do not necessarily follow the meta-methodology mentioned above. Instead, they create an artifact for a particular problem instance, thereby capturing the challenges and viewpoints of the industry.

6.1 Adopting Secure Development Practices

The first pilot was developed within a multinational organization. The developed approach highlights the complexity of adopting, among other practices, threat modeling. From the perspective of building

and operating secure software, OWASP SAMM presents a framework (in the non-technical sense) for adopting security practices. Figure 6.1 highlights the structure of the OWASP SAMM, which organizes security practices into different pillars, activities, and streams. Different maturity levels can be reached to allow organizations with varying capabilities and resources to adopt the framework. Here, threat modeling is only a small part of a more extensive set of activities to consider in order to secure a system. For example, it could be done on an ad-hoc, best-effort basis to achieve maturity level one, or it could be continuously practiced with tooling and methods harmonized across an organization to reach the highest level [199]. The specific framework was chosen due to its open nature, modular architecture, and its adoption by the organization.

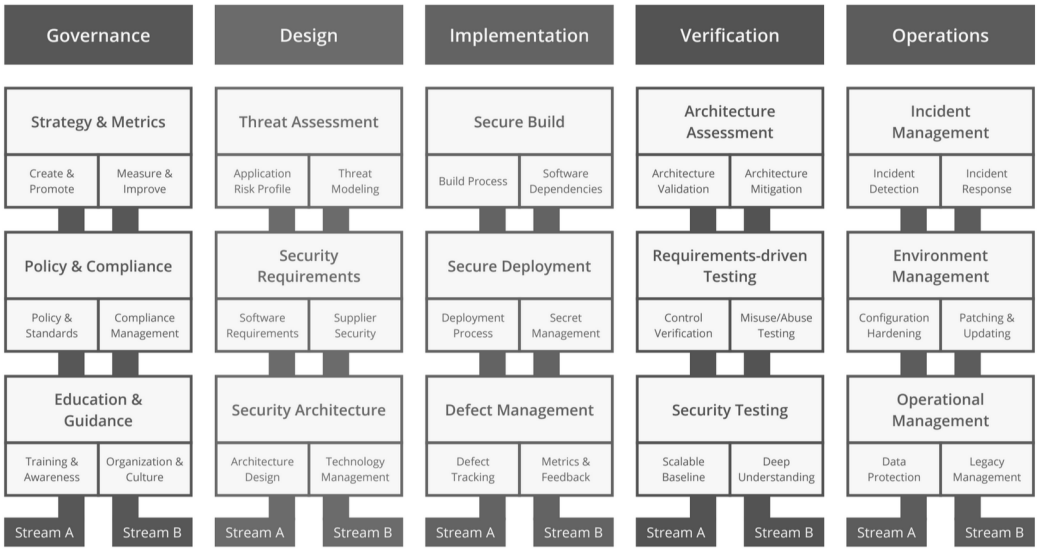

Figure 6.1: Security Practises in OWASP SAMM [199]

One of the key challenges of threat modeling and related security practices is to justify the expenses incurred in doing it [163]. Here, OWASP SAMM pragmatically tackles this by designing the framework to be both actionable and measurable [199]. Hence, the question arises of how the adoption and maturity of security practices could be measured within an organization. The following pilot depicts how this question is approached by developing a prototypical artifact that automatically gathers and aggregates metrics to gauge the maturity of security practices.

The architectural model of *CAVE* introduces and subsequently expands each character of the *CAVE* mnemonic. On the left side of Figure 6.2 are a set of *Collectors* that gather metrics from a pre-defined configuration (*i.e.*, the organization's selection of targeted security practices). At runtime, the actual

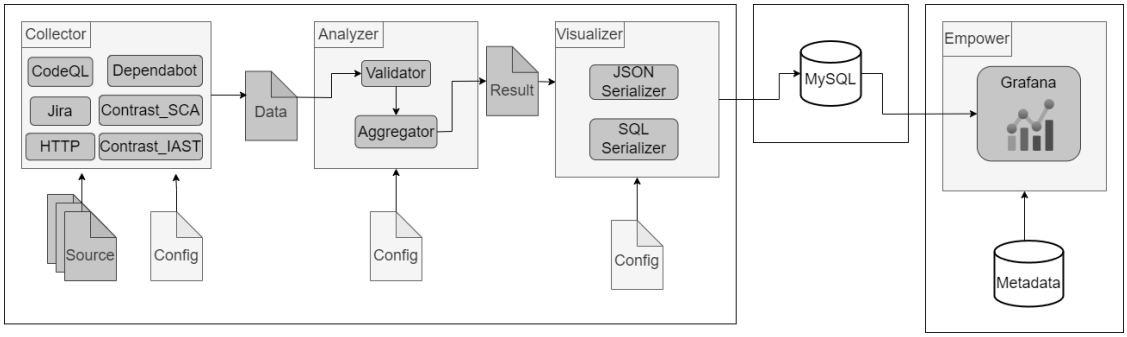

Figure 6.2: Architecture of the *CAVE* Platform [291]

values to be evaluated are automatically extracted from existing tools that one can find in a DevOps pipeline. For example, a development team may already employ *CodeQL* for static code analysis, providing continuous insight into security flaws. Secondly, *Validators* and *Aggregators* then interpret the output of those collectors. For example, a validator parses an output, such as the string "critical vulnerability" from a collector and evaluates it, yielding an aggregated result. Based on these, validation rules can be enforced. For example, the *Visualizer* collects and serializes metrics for visualization and stores them in a database. Based on these, managers can *empower* their security-related decision-making by leveraging dashboards and metrics from the database. For a more in-depth explanation of the design decisions, illustrations of its practical usage, and references to the source code of *CAVE*, refer to [291].

Concerning threat modeling as a practice, the development of *CAVE* highlights several observations. First, there exist organizations that embrace – or at least evaluate – the security-by-design principle, which underlines the threat modeling philosophy. Nevertheless, the existence of frameworks like OWASP SAMM and tools such as *CAVE* underline that positioning and defending the value of such practices does remain an ongoing challenge. From a more technical viewpoint, it stands out that the assessment of threat modeling maturity is still highly qualitative: threat modeling is "being done" or "not done"; it is done at continuous intervals or in a more ad-hoc manner. Likewise, threat modeling is conducted using best practices in an unstructured manner or maybe with highly formalized processes, formalisms, and structures. However, there appears to be still relatively little knowledge on how to measure various aspects of threat modeling, such as its effectiveness, the value gained, the efficiency of its creation, especially when considering other organizational factors (*e.g.*, technological, or human changes).

6.2 Service-based Threat Modeling

In the second pilot, a second version of the *CoReTM* prototype was developed in cooperation with an industrial partner. The partner not only practices threat modeling frequently but also champions the approach in conferences and other organizations. This makes the partner an invaluable (albeit from a scientific perspective, anecdotal) source of information since he not only practices and conveys threat modeling, but he does so as a service for other companies, having a great deal of insight into how it is being practiced across organizations.

The development of *CoReTM* 2.0 occurred as follows. First, the partner proactively incurred about the development of *CoReTM*, as described in [275]. Then, a feature elicitation of a novel threat modeling prototype was conducted in a bilateral discussion. Generally, the philosophy of the new design was to develop a threat modeling tool that can preserve much of the *CoReTM*'s modeling flexibility while automating common cross-cutting concerns. A particular threat modeling methodology (*i.e.*, STRIDE-per-interaction) was targeted to identify those areas for automation. Then, the partner defined several functional and non-functional requirements. For example, the tool was supposed to automatically create an entry for a potential threat when a data flow crosses a visually modeled trust boundary.

Then, this prototype was developed in several iterative design phases, where a set of requirements were prioritized, implemented, and tested. Finally, the industrial partner tested the resulting prototype, where feedback was collected in a think-aloud meeting and supplemented with a set of predefined questions.

Figure 6.3 presents the revised architecture of *CoReTM*, which was then implemented as a completely client-side running web application using the *React.js* framework. On the landing page, a new project can be created by providing a name, or an existing one can be imported. Then, the user is redirected to the modeling page, where the *diagrams.net* wrapper from *CoReTM* was refined and extended. Specifically, a new stencil library was created, holding only eight abstract items: processes, multi-processes, data stores, initiators, data flow, and trust boundary. To simplify the computation of data flows leaving a trust domain, the trust boundary was modeled as a simple rectangle. Hence, once the user creates an XML diagram, the data flows can be extracted. More specifically, the front end computes cross-boundary flows by assessing whether each element lies within a trust boundary. Simply put, it calculates whether the coordinates of elements lie within one or more trust boundaries by comparing the coordinates of corners in two-dimensional space. Then, the data flow triples are checked, *i.e.*, whether the source and destination, connected by a flow, do not share the same surrounding trust boundary. For those flows, the user is then prompted to enumerate which of the

STRIDE components may apply. Figure 6.4 exemplifies the creation of threat tables for a hypothetical scenario where a threat model is created of two backup strategies (*i.e.*, pull-based and push-based data transfer). Once the user selects those properties and navigates to the next page, instances of threats belonging to those properties can be modeled. For example, in Figure 6.4, the user could define several threats that perform a form of *Tampering* on the first data flow. This process is completely manual, *i.e.*, the modeler must identify threats based on experience or external knowledge bases [262].

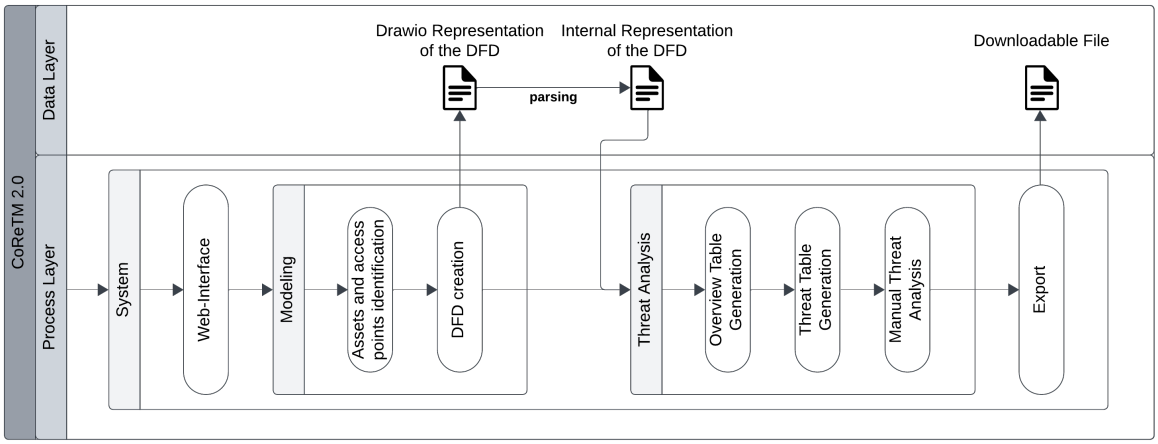

Figure 6.3: *CoReTM* 2.0 Architecture [262]

After developing a stable version of the tool, which comprises the main functionality and related utility functions (*e.g.*, exporting models, importing models), the industry partner evaluated the tool. Several UI improvements and future feature extensions were discovered. However, the industrial partner considered the initially defined requirements to be met, attesting to the completeness and usability of the prototype. [263] stands as a functional prototype that carries a subset of aspects of *CoReTM* into a practical state [262]. Based on the development of the tool and the discussions and tests conducted with the industrial partner, the following observations are described.

First, the development of *CoReTM* 2.0 does not directly address new research limitations or exploit new technologies or approaches other than the ones described by *CoReTM*. This can be interpreted in multiple ways. As argued by the partner, the key strength of *CoReTM* 2.0 lies in its simplicity and flexibility during modeling. Presumably, a practitioner with basic security knowledge and exposure to *diagrams.net* could navigate the tool, even without extensively learning about the *STRIDE* methodology. This contrasts with research and commercial tooling, which aim to automate the process fully. One could argue that these tools do not address a relevant need or that their implementation is not

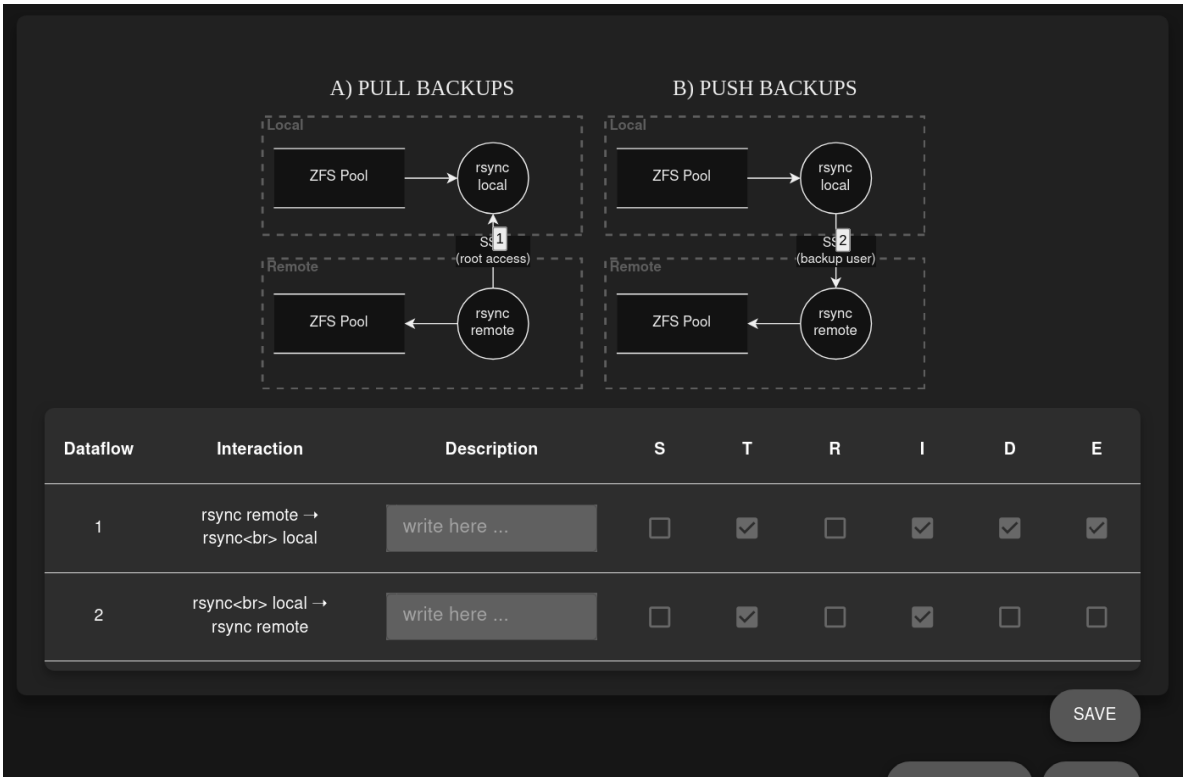

Figure 6.4: Threat Tables Created by *CoReTM* 2.0

mature. In contrast, *CoReTM* 2.0 does not aim to automate the threat modeling process completely and instead focuses on transparently guiding the methodology while simplifying tasks that would be laborious when done on a whiteboard or spreadsheet (*e.g.*, drawing tables, enumerating flows). Thus, it meets the voices of the *Threat Modeling Manifesto*, stating that "Support your approach with tools that allow you to increase your productivity, enhance your workflows, enable repeatability, and provide measurability [253]."

Secondly, it had been argued before that a lack of data (*e.g.*, having access to numerous data flows, system architectures, and threat reports) could negatively contribute to automating the threat identification stage. Assuming that practitioners could use the tool, the resulting threat reports and architectural models would remain fully machine-readable. Through this, the application of *CoReTM* 2.0 could contribute to this challenge. Still, the aspect of privacy and confidentiality would need to be addressed since it is unlikely that organizations will simply share all of their models.

Thirdly, the interaction with the partner highlighted that, for threat assessment, the usage of qualitative or "quasi-qualitative" methods is still common. Here, future research could focus on qualitatively surveying why this is still the case among threat modeling practitioners.

Clear is better than clever.

— Rob Pike (2015)

7 Summary and Future Research

Cyber attacks remain a highly relevant risk. 9.5×10^{12} USD have been reaped by cybercriminals. Aggregating the cybercrime economy, its *Gross Domestic Product* (GDP) would only be outnumbered by two behemothic economies (*cf.*, Figure 7.1). In theory, a risk-based approach provides a simple way for organizations, states, and individuals to reduce those adverse outcomes. Simply put, the events and factors leading to the most considerable negative outcomes of sensitive assets must be identified, and the risk-reducing measures with the largest net benefit on risk reduction be implemented.

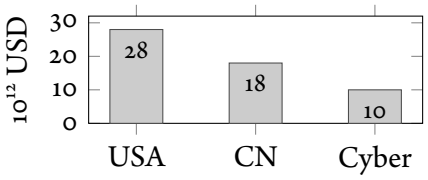

Figure 7.1: Cybercrime GDP [28]

In reality, this is non-trivial for various reasons; risk is fundamentally concerned with the uncertainty of those outcomes. Practically, this uncertainty must be approximated at multiple stages – from the conceptualization of a system to its ongoing operation. Furthermore, the influencing factors must be assessed in different hierarchies of an organization facing technical, social, and economic challenges. Thus, this thesis adopts the risk-based philosophy and reports on several design studies that tackled specific problems across an information system's life cycle – while focusing on the abstraction of threats.

7.1 Summary

After the scope of the research and its motivations were introduced, this thesis introduced the theory and practices behind the concept of information security risk. Since this thesis embraces a holistic view of cybersecurity, prevention, detection, and response were introduced as concrete risk-reducing measures and as a frame to discuss security practices. As a motivation for those activities, the threat landscape was overviewed, illustrating a variety of threats. The thesis at hand does not provide a new overarching framework or methodology; instead, it reports on several design studies. These studies elicited concrete problems asserting relevance through environmental factors framed through *people*, *organizations*, and *technology*. The space summarizing existing solutions for those problems was surveyed, presenting limitations and opportunities to be addressed. To support the design science research methodology, several problem domains, research foundations and methods were introduced.

After introducing these overarching aspects, existing solutions were reviewed. Motivating examples were provided, illustrating specific problems (*e.g.*, the need for autonomous defense or the relevance of insider attacks). Concerning threat modeling tooling and methods, these limitations included a lack of collaborative features, an absence of tooling addressing the technological change caused by the emergence of AI-based systems, the inclusion of non-technical insider threats, and the lack of quantitative approaches when assessing threats. With respect to detecting and managing such threats, the unaddressed problem of orchestrating and building a suite of dynamic defenses was described. In addition, reactive defense suffers from the conceptual limitation of detecting *damage* while proactive defense is prone to inefficiency.

In the following chapters, these limitations are transformed into opportunities, and a set of artifacts was developed that improves over the limitations while taking the specific problem domains as validation scenarios. *CoReTM* addressed the role of collaboration during threat modeling – as emphasized by scenarios where asynchronous modeling or remote stakeholder inclusion is necessary. The solution proposes and prototypically implements a flexible, real-time, annotation-based editor, knowledge bases, interactive methodological guides, and macroscopical navigation. The utility of *CoReTM* illustrates that the tool can reflect technical and non-technical security modeling of a real-world information system. The second solution proposed a process modeling editor and a knowledge base of insider threats. Together, the prototype guides users to model insider threats using process models. Two evaluations were performed: a case study in a commercial context highlighted that the prototype is applicable against existing models and that stakeholders can identify relevant threats. Another evaluation targeted non-commercial settings, illustrating that users could leverage the tool to analyze the postal voting process in Switzerland. The resulting threat models reflect relevant threats,

indicating the approach's effectiveness. The design study yielding the *QuantTM* approach proposes integrating quantitative methods into threat modeling. To gather the values for impact-related factors, *QuantTM* explored concepts from BCM, developing a BIA prototype that helps users assess the impacts of threats. Various evaluations attest to *QuantTM*'s applicability in real scenarios, the expressiveness of results, and the prototype's usability. Lastly, *ThreatFinderAI* emphasizes the previously mentioned problems while moving to a particular problem domain: AI security. A seven-stage methodology is proposed and instantiated through a prototype leveraging a custom stencil library of architectural AI assets, a modeling methodology, and threat and control identification through a knowledge base. To assess the remaining risk exposure, *ThreatFinderAI* proposes a customized BIA approach for AI security and estimate-based quantification. A demonstration of *ThreatFinderAI* illustrates how the approach can provide insightful security discussions to an LLM system integration. A user experiment demonstrates the ability of non-security users to reproduce a threat model, and a case study demonstrates the applicability of the tool against a real scenario.

The second part of the thesis presented studies to detect and mitigate threats. A lightweight MTD framework was developed, including four defense techniques. Specifically, the framework is constrained to an autonomous defense scenario, testing both reactive orchestration and proactive deployment. It was demonstrated that the framework can mitigate the threats of a statically defined threat model. Reactive techniques present responsive mitigation for non-passive workloads, demonstrating improved effectiveness and efficiency over proactive scenarios. *R-MTD* mapped techniques to a threat model while building a rule-based behavioral detection system. Experiments demonstrate that the rule-based system can effectively defend against malware-based attacks while reducing operational resource usage at the expense of increased development complexity. *SecBox* isolated the concern of gathering behavioral data in a systematic, reproducible, and cost-effective manner by proposing a container-based sandboxing framework. These objectives were evaluated in various tests, considering offensive testing, escape detection, dataset similarity, resource consumption, and data utility. Finally, proactive and reactive defense techniques were designed by proposing a file-system-based approach. The three proactive techniques demonstrate that virtualizing the attack surface can make the defense more cost-effective. In contrast, the reactive and preventive techniques reduce data loss to a more substantial degree while introducing a tradeoff to resource consumption or usability.

Through those studies and industry collaborations, the solution space is enriched with methods and concrete tools – many of which are available as an open-source prototype. On the other hand, the deployments and evaluations of those artifacts present additional insights into the problem space domains, *i.e.*, outlining novel research directions for technical and social challenges based on the newly obtained knowledge.

7.2 Research Questions Revisited

In the following, the initially defined research questions that guided this thesis's studies and practical developments are revisited. For each, the contributions are briefly reiterated, and an attempt at answering the questions is made while outlining open gaps that could be addressed by future research. Naturally, the evidence for several solutions includes qualitative and interpretative mechanisms. Thus, an attempt is made to disclose such results transparently.

RQ1) "Which limitations are faced when modeling cybersecurity threats, and how can they be overcome through novel methods and tooling?"

Research on RQ1 established the threat-oriented approach that influenced the remaining research lines. Through participation in practical threat modeling contexts, literature analysis, and dynamic execution, a variety of limitations were discovered. These include both technical (*e.g.*, which data sources can be leveraged, which system assets should be modeled) and social challenges (*e.g.*, how to include asynchronous modelers or how to model insider attacks). In this thesis, those aspects were mainly addressed through approaches that do not aim to fully automate the creation of a threat model. Instead, the approach is to provide tooling that guides and navigates the user when creating such models. Through this, users can discover attacks they may not have been educated for (*e.g.*, insider attacks, attacks on AI) while making the overall approach systematic. This philosophy aligns with practitioners outside the academic community, who seek flexible, non-intrusive tooling as backed by the Threat Modeling Manifesto and the pilot performed.

Nevertheless, automation and scalability remain key challenges in threat modeling. For example, one could imagine that a one-hour threat modeling workshop with two software developers could lead to a security flaw being *prevented* at design time instead of being *mitigated* during production. Arguably, these two hours could be considered well-spent. However, how would this net benefit look after the architecture is revised after several weeks? How much time would be spent familiarizing with the models, and how many design-time flaws can be found? Here, the inherent scalability issues of a design-time activity would need to be addressed, and practitioners would likely need strong arguments to conduct threat modeling regularly.

The aspect of automation is closely related, which is a commonly envisioned functionality. Here, a juxtaposition arises when discussing the degree of automation that could be sensibly achieved. Automatically generating threat models would require, first and foremost, data. One would need data from real threat models to build a meta-model and data about the specific scenario for which one would like to obtain a threat model, just like one would require training data and inference data in

ML. Assume that a powerful recommender system has learned from thousands of threat models. What would be the input data for that model, assuming that threat modeling is to be done *as early as possible*? At that point, only minimal data about the target system will likely be available. For this reason, the prototypes of this thesis rely on high-level artifacts that may already exist due to other activities (*e.g.*, architectural modeling or business process modeling). Rich data is likely only available in two cases – which are both suboptimal: when creating it specifically or when conducting threat modeling later in the process. The latter symbolizes the approaches investigated in this thesis, where source code is used to generate a threat model, indicating that the idea of design-time flaw prevention would not be fulfilled.

Thus, in response to ***RQ1)***, the contributions in this thesis could be considered to raise attention to certain blind spots in threat modeling, while attesting that proper tooling can improve the effectiveness and efficiency of threat modeling. For example, the modeling of insider threats was not observed in practice, such as in the illustrated e-government examples. Moreover, practical approaches to cover those problems have been proposed, implemented, and validated concerning their feasibility.

RQ2) "Is it possible to formulate quantitative risk-based representations of threats, and how sensible is it to do so in the context of threat modeling?"

When analyzing the literature and the existing prototypes, it becomes evident that quantitative approaches are almost completely absent in threat modeling. More so, approaches using organization-centric assessments are not commonly employed. Instead, quantitative approaches focus on measuring technical aspects (*e.g.*, number of interfaces or size of a system). Looking at how threat modeling is practiced, this may not be different, and existing approaches, even ones abandoned by their developers, may continue to be used. In this thesis, quantitative approaches considered the business impact of a threat. In response to ***RQ2)***, it was demonstrated that this approach is feasible to be practically executed and could arguably present more granular results through a systematic methodology.

However, the key limitation here is a lack of data. Due to a lack of ground-truth data, proving the correctness of these approaches is not provided. Nevertheless, the illustrated properties (*e.g.*, being systematic, explicit, non-interpretative) still present an advantage over others, even when modelers can only rely on estimates. This is illustrated by the case of AI security threats, where historical data is scarce due to the novelty of the paradigm. Quantitative methods could be advantageous to formulate the uncertainty of modelers more precisely for technical and strategic optimization of the specific organization. For example, a modeler expressing their uncertainty about the negative impact of an attacker exfiltrating sensitive training data from a model is less ambiguous when formulated as "presumably between 0 and 1000 customer records could be recovered" instead of introducing "a

medium impact". Even when data is complex to attain (*e.g.*, due to unwillingness or unavailability of external data), it is argued that impacts should be translated into business terms, which would be a more approachable dimension than attack probability.

Finally, when contrasting threat modeling with other risk-based approaches, the aspect of it being a design-stage activity may be relevant. Arguably, not every threat in threat modeling benefits from an in-depth quantification, indicating that decision-making in security is and may continue to be fuzzy. For example, implementing the control measures "requiring strong passwords" and "two authentication factors" may not need an impact assessment to be justified over using only one authentication method. This also emphasizes that qualification and quantification resemble merely a language to reason about threats, and threats are not inherently qualitative or quantitative. Nevertheless, quantifying using financial terms may be unnatural depending on the discussion. For example, one experiment considered voting security. Certainly, attacks could be quantified by expressing their magnitude in terms of votes being impacted; however, translating this attack into a financial statement would be challenging.

RQ3) "What novel ways exist to extract cyberattack data, and how useful is that data to represent threat abstractions?"

The detective approaches used in this thesis focused on behavioral data as a rich and powerful avenue for building decision-making systems. Here, the usage of threat models arises as a key direction. If data is not gathered in a secure area, the detection system may be prone to adversarial attacks. An approach satisfying a particular threat model is to move the creation of behavioral threat data outside of the device into a secure sandbox. Here, the studies demonstrate that a container-based framework could provide the necessary functions to obtain such data securely and scalably. This showed that it is possible to obtain data that can be used for a family of threats; for example, crypto-ransomware attacks were successfully mitigated with data collected across samples. Thus, research on ***RQ3)*** argues that data can be captured both on-device and in isolated environments, aligning detection with a particular threat model.

However, the key limitation in using such data is closely aligned with its strength: even simple changes in behavior can be detected, implying that small changes in both benign and malicious behavior also affect the system's effectiveness. One would need to *(i)* closely monitor the performance of such systems, *(ii)* screen the threat landscape and the *(iii)* benign workload, since both likely require *(iv)* re-execution of samples and simulation of benign behavior and retraining of models.

RQ4) "How effective and efficient is it to mitigate cyberattacks by developing proactive and reactive mitigation strategies that tackle families of threats?"

Initially, the results obtained from the hybrid defense framework indicated that autonomous defense can orchestrate multiple defense techniques, reducing but not wholly preventing the impact. Tangible aspects of the attack surface were used to deceive and mitigate an attacker who breached a previous line of defense. This was successful for both reactive and proactive deployments for several instances of the same threat (*e.g.*, multiple ransomware samples). However, reactive defense was more effective, causing less impact. Several proactive techniques were explored for further improvement, which leveraged a partially virtualized attack surface. It was demonstrated that, under the assumption of the threat model being applicable and static, the techniques further reduced losses while improving efficiency. The reactive scenario tested how effective such orchestration is when reducing the input to the detection system to securely obtainable features. It was shown that this, in addition to techniques designed to be preventive, can further reduce the losses incurred by the malware. However, a tradeoff between defense security and efficiency, as well as usability, had been discovered. Furthermore, the overall system could not completely mitigate and prevent losses.

This highlights that even employing a powerful detection system leads to losses. It is argued that such detection systems benefit from measuring their utility by incorporating them into an actual end-to-end defense scenario. However, this requires access to a relevant defense system and the formulation of a threat model with specific instances for testing. While this highlights concrete results, it is also a vulnerable assumption. For example, changes in the attacker's tactics would lead to a change in the threat model, questioning the system's long-term performance. Hence, for ***RQ4)***, it is argued that both reactive and proactive defense can be beneficial, while benefitting from an in-depth discussion of the underlying threat model and its assumptions. Moreover, offensive testing is suggested as a research method for the field. For example, *RansomAI* was developed to test how the defense systems hold up if an attacker uses signals emitted by the defense system. In that sense, it explores how a dynamic attack behavior within a static threat model (*e.g.*, assuming ransomware exhibits encryption behavior). As such, the specific numbers reported within these studies (*i.e.*, quantifying the "how much" in this research question) must consider the particular scenario and be carefully generalized.

7.3 Limitations and Avenues for Future Research

The studies performed in this thesis are subject to two types of limitations: those pertaining to the solutions' assumptions and their actual results, and epistemological limitations stemming from the research methods employed. For example, the MTD solution described in Section 5.1 was able to reduce–but not completely eliminate–malware execution time. The limitations of this result must further consider the assumptions (*e.g.*, leveraging a particular scenario and attack samples) and the

research method (*e.g.*, limited reproducibility from experimenting in a real, non-simulated device). In the following, key limitations are summarized. For further details, the reader is referred to Section 2.4.3 where the research methods and their limitations are introduced. For in-depth limitations of each solution, refer to the discussions of each solution presented within Chapters 4 and 5. With respect to research on threat modeling, the practical key limitation is the cost of applying them: each solution depends on existing input (*e.g.*, architectural diagrams, process models) that must be developed and the methodological execution. These modeling solutions considered their applications in specific settings, which assumes access to particular scenarios. From a research perspective, it must be re-emphasized that evaluations considered practical settings. Thus, the results are biased due to user involvement and cannot be easily generalized due to their approach and small sample size. Furthermore, the experiments targeted expressiveness, viability, and usability but did not necessarily prove the correctness of outcomes. The studies on attack detection and mitigation all reported residual damage and thus do not provide a lossless defense. In addition, the solutions and their results are based on the underlying assumptions, which may change or be inaccurate. For example, the ransomware threat model considered adversarial directory traversal. In practice, attackers may adapt this strategy, questioning the obtained results. Furthermore, the research methods leveraged both practical and partially experimental settings (*i.e.*, testing in *ElectroSense*) and cannot be generalized to any other setting. Nevertheless, the studies argue that both are needed depending on the research objective, yielding results that are not just generalizable but also relevant.

Due to these limitations, multiple avenues for future research exist with respect to the previous research questions. Concerning *RQ1*, a reasonable step would be to test the approaches with more stakeholders, especially ones sharing different backgrounds. It could be investigated whether the *ThreatFinderAI* tooling can be successfully used by additional groups of users, *e.g.*, software architects and security experts. Furthermore, a promising approach lies in the aforementioned lack of data. For example, [300] investigated from an empirical perspective how users employ Generative AI, such as ChatGPT, analyzing and descriptively reporting the security practices during their usage. Their results indicate specific numbers that, although only descriptively presented, could be integrated into a practical method. For example, *RQ2* could be further investigated with a prototype integrating this empirical data to replace the estimates used during threat quantification. By leveraging such data, other statistical methods were applicable, too. For example, Bayesian statistics could be employed to update the beliefs expressed from previously assessed estimates. Thus, since the philosophy of modeling methods embraces a "best data approach", simple subjective estimates can be improved by future research yielding complementary empirical data. Furthermore, the discipline as a whole would benefit from a collection of publicly available threat models that could be used for research. For less

technical research, assessing the benefits of conducting these practices would not only be an unanswered research direction but, in fact, one that could support decision-making in practice. However, due to the design time application of these approaches, researchers would need to investigate how such factors affect the final security of a service, accounting for various aspects.

With respect to the sandboxing system developed under *RQ3*, a key research direction could investigate the similarity of data created across different environments. For example, *SecBox* enables the deployment of the execution component into a decoupled system. Here, it could be assessed how closely a simulated environment (*e.g.*, a container running on a simulated ARM processor) compares to one deployed on a Raspberry Pi device. Similarly, the defensive techniques of *RQ4* could be questioned by revisiting and challenging the assumptions of the threat models upon which they are built. Here, the approach of iterative building, testing, and breaking could be leveraged as a research methodology, highlighting novel limitations of those defense systems, which, in turn, can be formulated into opportunities.

Bibliography

[1] A. Abbasi, J. Wetzels, T. Holz, and S. Etalle, "Challenges in Designing Exploit Mitigations for Deeply Embedded Systems," 2019 IEEE European Symposium on Security and Privacy (EuroS P), 2019, pp. 31–46.

[2] S. Abdelnabi and M. Fritz, "What's in the box: Deflecting Adversarial Attacks by Randomly Deploying Adversarially-Disjoint Models," Proceedings of the 8th ACM Workshop on Moving Target Defense. New York, NY, USA: Association for Computing Machinery, Nov. 2021.

[3] abuse.ch, "MalwareBazaar | mirai," September 2024, https://bazaar.abuse.ch/browse/tag/mirai/.

[4] abuse.ch, "MalwareBazaar," https://bazaar.abuse.ch/, 2025.

[5] P. Achimugu, A. Selamat, R. Ibrahim, and M. N. Mahrin, "A systematic literature review of software requirements prioritization research," Information and software technology, Vol. 56, No. 6, pp. 568–585, 2014.

[6] A. O. Almashhadani, M. Kaiiali, S. Sezer, and P. O'Kane, "A Multi-Classifier Network-Based Crypto Ransomware Detection System: A Case Study of Locky Ransomware," IEEE Access, Vol. 7, pp. 47 053–47 067, 2019.

[7] A. A. M. A. Alwashali, N. A. Abd Rahman, and N. Ismail, "A survey of ransomware as a service (RaaS) and methods to mitigate the attack," 2021 14th International Conference on Developments in eSystems Engineering (DeSE), 2021, pp. 92–96.

[8] K. Amin and K. Engin, "Redemption: Real-time Protection Against Ransomware at End-Hosts," Research in Attacks, Intrusions, and Defenses: 20th International Symposium, pp. 98–119, 2017.

[9] C. S. Anand and R. Shanker, "Advancing Crypto Ransomware with Multi Level Extortion: A Peril to Critical Infrastructure," 2023 2nd International Conference for Innovation in Technology (INOCON), 2023, pp. 1–5.

[10] D. R. Anderson, T. A. Williams, and J. J. Cochran, Statistics for business & economics. Cengage Learning, 2020.

[11] K. J. Andreasson, Cybersecurity: Public Sector Threats and Responses. Taylor & Francis, 2011.

[12] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and don'ts of machine learning in computer security," 31st USENIX Security Symposium (USENIX Security 22), 2022, pp. 3971–3988.

[13] R. S. Arslan, "Identify Type of Android Malware with Machine Learning Based Ensemble Model," 2021 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), 2021, pp. 628–632.

[14] O. A. Aslan and R. Samet, "A comprehensive review on malware detection approaches," IEEE Access, Vol. 8, pp. 6249–6271, 2020.

[15] T. Baluta, L. Ramapantulu, Y. M. Teo, and E.-C. Chang, "Modeling the effects of insider threats on cybersecurity of complex systems," 2017 Winter Simulation Conference (WSC). IEEE, 2017, pp. 4360–4371.

[16] R. L. Baskerville, J. Kim, and C. Stucke, "The cybersecurity risk estimation engine: A tool for possibility based risk analysis," Computers & Security, Vol. 120, 2022.

[17] K. Beck, Extreme Programming Explained: Embrace Change. Addison-Wesley, 2000.

[18] D. T. Bee and D. Murdoch-Eaton, "Questionnaire design: the good, the bad and the pitfalls," Archives of Disease in Childhood-Education and Practice, Vol. 101, No. 4, pp. 210–212, 2016.

[19] C. Bello, E. Albertin, J. Iliarte, and T. Torrijos, "Platform Architecture, Requirements, and Feedback Loops (Version 1.0)," 2023.

[20] P. Berander, "Requirements prioritization," Engineering and Managing Software Requirements/Springer Verlag, 2005.

[21] V. Berasneu, "Design and implementation of a system for reproducible machine and deep learning models," Communication Systems Group, Department of Informatics, Zürich, Switzerland, sep 2023, https://files.ifi.uzh.ch/CSG/staff/vonderassen/extern/theses/BA-Berasneu.pdf.

[22] J. Bernard, "Overcoming challenges to cyber insurance growth," Deloitte Insights, 2020.

[23] K. Bernsmed, D. Cruzes, M. Jaatun, and M. Iovan, "Adopting Threat Modelling in Agile Software Development Projects," Journal of Systems and Software, Vol. 183, p. 111090, 09 2021.

[24] P. L. Bernstein and P. L. Bernstein, Against the gods: The remarkable story of risk. Wiley New York, 1996.

[25] M. binti Mohamad Noor and W. H. Hassan, "Current research on Internet of Things (IoT) security: A survey," Computer Networks, Vol. 148, pp. 283–294, 2019.

[26] M. Bishop, H. M. Conboy, H. Phan, B. I. Simidchieva, G. S. Avrunin, L. A. Clarke, L. J. Osterweil, and S. Peisert, "Insider Threat Identification by Process Analysis," 2014 IEEE Security and Privacy Workshops. San Jose, CA, USA: IEEE, 2014, pp. 251–264.

[27] F. Björck, M. Henkel, J. Stirna, and J. Zdravkovic, "Cyber resilience – fundamentals for a definition," New Contributions in Information Systems and Technologies. Cham: Springer International Publishing, 2015, pp. 311–316.

[28] Bloomberg L.P, "The world's third-largest economy has bad intentions — and it's only getting bigger," 2024, https://sponsored.bloomberg.com/quicksight/check-point/the-worlds-third-largest-economy-has-bad-intentions-and-its-only-getting-bigger.

[29] J. Boehm, N. Curcio, P. Merrath, L. Shenton, and T. Stähle, "The risk-based approach to cybersecurity," McKinsey, New York, 2019.

[30] R. Böhme, "Security Metrics and Security Investment Models," Advances in Information and Computer Security. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 10–24.

[31] G. Booch, I. Jacobson, and J. Rumbaugh, "Unified Modeling Language (UML)," Rational Software Corporation, Santa Clara, CA, version, Vol. 1, 1997.

[32] K. Brancik, Insider Computer Fraud: An In-depth Framework for Detecting and Defending against Insider IT Attacks. CRC Press, 2007.

[33] E. S. Brenda Larcom and S. Smith, "Docs | octotrike.org," http://www.octotrike.org/docs.

[34] R. Brown, A. Marti, C. Jenkins, and S. Shannigrahi, "Dynamic Address Validation Array (DAVA): A Moving Target Defense Protocol for CAN bus," Proceedings of the 7th ACM Workshop on Moving Target Defense, ser. MTD'20. New York, NY, USA: Association for Computing Machinery, Nov. 2020, pp. 11–19.

[35] BSI, "BSI-Standard 200-3," Bonn, 2017, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/BSI_Standards/standard_200_3.pdf?__blob=publicationFile&v=2.

[36] BSI, "IT-Grundschutz-Kompendium," Bonn, 2023, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/Kompendium/IT_Grundschutz_Kompendium_Edition2023.pdf?__blob=publicationFile&v=4#download=1.

[37] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," IEEE Communications Surveys & Tutorials, Vol. 18, No. 2, pp. 1153–1176, 2016.

[38] C. Ardagna, E. Damiani, M. Anisetti, and M. C. (Editors),, "1st Year Report on Cybersecurity Threats," December 2019, https://www.zora.uzh.ch/id/eprint/232147/1/232147.pdf.

[39] CAIRIS, "Threat Modelling, Documentation and More," 2022, https://cairis.org/cairis/tmdocsmore/.

[40] Camunda Services GmbH, "Web-based tooling for BPMN, DMN and Forms," https://bpmn.io/.

[41] CAPEC, "About CAPEC," April 2019, https://capec.mitre.org/about/index.html.

[42] CAPEC, "CAPEC - ATT&CK Comparison," 2019 October, https://capec.mitre.org/about/attack_comparison.html.

[43] R. Caralli, J. Stevens, L. Young, and W. Wilson, "Introducing OCTAVE Allegro: Improving the Information Security Risk Assessment Process," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU/SEI-2007-TR-012, 2007, http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8419.

[44] V. Casola, A. De Benedictis, and M. Albanese, "A Moving Target Defense Approach for Protecting Resource-constrained Distributed Devices," 2013 IEEE 14th International Conference on Information Reuse & Integration (IRI), San Francisco, CA, USA, 2013, pp. 22–29.

[45] V. Casola, A. De Benedictis, M. Rak, and U. Villano, "Toward the automation of threat modeling and risk assessment in IoT systems," Internet of Things, Vol. 7, p. 100056, 2019.

[46] A. Cater-Steel and L. Al-Hakim, Information Systems Research Methods, Epistemology, and Applications. IGI Global, 2008.

[47] A. H. Celdrán, P. M. S. Sánchez, M. A. Castillo, G. Bovet, G. M. Pérez, and B. Stiller, "Intelligent and behavioral-based detection of malware in IoT spectrum sensors," International Journal of Information Security, Vol. 22, No. 3, pp. 541–561, 2023.

[48] A. H. Celdrán, J. v. d. Assen, C. Feng, S. Padovan, G. Bovet, and B. Stiller, "Next Generation of AI-based Ransomware," 2024 IEEE Global Communications Conference: Communication & Information Systems Security. Cape Town, South Africa: IEEE, dec 2024, pp. 1–6.

[49] Center for Internet Security, "CIS Controls Version 8," May 2021, https://www.cisecurity.org/controls/v8.

[50] V. Cerullo and M. J. Cerullo, "Business Continuity Planning: A Comprehensive Approach," Information Systems Management, Vol. 21, No. 3, pp. 70–78, 2004.

[51] I. A. Chesti, M. Humayun, N. U. Sama, and N. Jhanjhi, "Evolution, Mitigation, and Prevention of Ransomware," 2020 2nd International Conference on Computer and Information Sciences (ICCIS), 2020, pp. 1–6.

[52] R. Chinchani, A. Iyer, H. Ngo, and S. Upadhyaya, "Towards A Theory Of Insider Threat Assessment," International Conference on Dependable Systems and Networks, Yokohama, Japan, 2005, pp. 108–117.

[53] M. Chinosi and A. Trombetta, "BPMN: An introduction to the standard," Computer Standards & Interfaces, Vol. 34, No. 1, pp. 124–134, 2012.

[54] C. Chio and D. Freeman, Machine learning and security: Protecting systems with data and algorithms. " O'Reilly Media, Inc.", 2018.

[55] J.-H. Cho, D. P. Sharma, H. Alavizadeh, S. Yoon, N. Ben-Asher, T. J. Moore, D. S. Kim, H. Lim, and F. F. Nelson, "Toward proactive, adaptive defense: A survey on moving target defense," IEEE Communications Surveys & Tutorials, Vol. 22, No. 1, pp. 709–745, 2020.

[56] CloudFlare, Inc, "Application layer DDoS attack," https://www.cloudflare.com/learning/ddos/application-layer-ddos-attack/.

[57] CloudFlare, Inc, "DNS amplification DDoS attack," https://www.cloudflare.com/learning/ddos/dns-amplification-ddos-attack/.

[58] CloudFlare, Inc, "How do layer 3 DDoS attacks work?" https://www.cloudflare.com/learning/ddos/layer-3-ddos-attacks/.

[59] CloudFlare, Inc, "NTP amplification DDoS attack," https://www.cloudflare.com/learning/ddos/ntp-amplification-ddos-attack/.

[60] F. Cohen, "Computer viruses: theory and experiments," Computers & security, Vol. 6, No. 1, pp. 22–35, 1987.

[61] Committee of Sponsoring Organizations of the Treadway Commission (COSO), Enterprise Risk Management – Integrating with Strategy and Performance. AICPA, 2017.

[62] A. Continella, A. Guagnelli, G. Zingaro, G. De Pasquale, A. Barenghi, S. Zanero, and F. Maggi, "ShieldFS: A Self-healing, Ransomware-aware Filesystem," Proceedings of the 32nd annual conference on computer security applications, 2016, pp. 336–347.

[63] A. Corallo, M. Lazoi, and M. Lezzi, "Cybersecurity in the context of industry 4.0: A structured classification of critical assets and business impacts," Computers in industry, Vol. 114, p. 103165, 2020.

[64] CrowdStrike Holdings, Inc, "A Brief History of Ransomware," https://www.crowdstrike.com/cybersecurity-101/ransomware/history-of-ransomware/.

[65] Cyber-Tracer, "MTDStrategySelectionAgent," https://github.com/Cyber-Tracer/MTDStrategySelectionAgent, 2022.

[66] Cybersecurity and Infrastructure Security Agency, "2021 Top Malware Strains," August 2022, https://www.cisa.gov/news-events/cybersecurity-advisories/aa22-216a.

[67] K. Das and R. N. Behera, "A survey on machine learning: Concept, algorithms and applications," International Journal of Innovative Research in Computer and Communication Engineering, Vol. 5, No. 2, pp. 1301–1309, 2017.

[68] devsecops, "Raindance," June 2016, https://github.com/devsecops/raindance/blob/master/GET-STARTED.md.

[69] K. Dooley, "Simulation research methods," The Blackwell companion to organizations, pp. 829–848, 2017.

[70] Q. Duan, E. Al-Shaer, and J. Xie, "Range and Topology Mutation Based Wireless Agility," Proceedings of the 7th ACM Workshop on Moving Target Defense, ser. MTD'20. New York, NY, USA: Association for Computing Machinery, Nov. 2020, pp. 59–67.

[71] EC-Councl, "DREAD Threat Modeling: An Introduction to Qualitative Risk Analysis," 2022, https://www.eccouncil.org/cybersecurity-exchange/threat-intelligence/dread-threat-modeling-intro/.

[72] E. Engström, M.-A. Storey, P. Runeson, M. Höst, and M. T. Baldassarre, "How software engineering research aligns with design science: a review," Empirical Software Engineering, Vol. 25, pp. 2630–2660, 2020.

[73] A. Erola, I. Agrafiotis, J. R. Nurse, L. Axon, M. Goldsmith, and S. Creese, "A system to calculate Cyber Value-at-Risk," Computers & Security, Vol. 113, 2022.

[74] European Parliament and of the Council, "DIRECTIVE (EU) 2022/2555 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL," 2022, https://eur-lex.europa.eu/eli/dir/2022/2555.

[75] European Union Agency for Cybersecurity (ENISA), "Artificial Intelligence Cybersecurity Challenges, Threat Landscape for Artificial Intelligence," 2020.

[76] S. Faily and C. Iacob, "Design as Code: Facilitating Collaboration Between Usability and Security Engineers Using CAIRIS," 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW), Lisbon, Portugal, 2017, pp. 76–82.

[77] B. Fatemeh, "Customer Care Support at Swisscom: From Conventional to Generative Bots," Jun. 2024, https://www.youtube.com/watch?v=bcWZkEkRNbc.

[78] Federal Office of Public Health of the Swiss Confederation, "Information on the COVID-19 certificate FOPH," February 2022, https://foph-coronavirus.ch/certificate/.

[79] T. Fei, M. Boyang, L. Jinku, Z. Fengwei, S. Jipeng, and M. Jianfeng, "Ransomspector: An introspection-based approach to detect crypto ransomware," Computers & Security, Vol. 97, p. 101997, 2020.

[80] T. Felber, "Fitnesstrainer verkauft über 400 gefälschte Covid-Impfzertifikate," March 2023, https://www.nzz.ch/zuerich/fitnesstrainer-verkauft-ueber-400-gefaelschte-covid-impfzertifikate-ld.1730580.

[81] C. Feng, J. Von Der Assen, A. H. Celdran, R. Mogicato, A. Zermin, V. Ok, G. Bovet, and B. Stiller, "SecBox: a Lightweight Data Mining Platform for Dynamic and Reproducible Malware Analysis," 2024 11th IEEE Swiss Conference on Data Science (SDS), 2024, pp. 62–67.

[82] C. Feng, J. von der Assen, A. Huertas Celdrán, S. Näf, G. Bovet, and B. Stiller, "FeDef: A Federated Defense Framework Using Cooperative Moving Target Defense," 2023 8th International Conference on Smart and Sustainable Technologies (SpliTech), 2023, pp. 1–6.

[83] D. W. Fernando and N. Komninos, "FeSAD ransomware detection framework with machine learning using adaption to concept drift," Computers & Security, Vol. 137, p. 103629, 2024.

[84] M. Figueredo Franco, "CyberTEA: a technical and economic approach for cybersecurity planning and investment," Ph.D. dissertation, University of Zurich, 2023.

[85] C. L. Flores, "Facebook Downtime: What Really Happened," 10 2021, https://www.akcp.com/articles/facebook-downtime-what-really-happened/.

[86] forseeti, "Introduction," https://docs.foreseeti.com/docs.

[87] forseeti, "MAL - Meta Attack Language," https://mal-lang.org/.

[88] M. F. Franco, E. Sula, A. Huertas, E. J. Scheid, L. Z. Granville, and B. Stiller, "SecRiskAI: a Machine Learning-Based Approach for Cybersecurity Risk Prediction in Businesses," 24th IEEE International Conference on Business Informatics (CBI 2022). Amsterdan, Netherlands: IEEE, 2022, pp. 1–10.

[89] M. Franco, C. Omlin, O. Kamer, E. Scheid, L. Granville, and B. Stiller, "SECAdvisor: A Tool for Cybersecurity Planning using Economic Models," 24º Brazilian Symposium on Information and Computational Systems Security (SBSeg 2024). Porto Alegre, RS, Brasil: SBC, 2024, pp. 554–569.

[90] M. F. Franco, F. Künzler, J. von der Assen, C. Feng, and B. Stiller, "RCVaR: an Economic Approach to Estimate Cyberattacks Costs using Data from Industry Reports," Computers & Security, p. 103737, 2024.

[91] M. F. Franco, A. R. Mullick, and S. Jha, "Qber: Quantifying cyber risks for strategic decisions," arXiv preprint arXiv:2405.03513, 2024.

[92] N. Fraser, "Differential Synchronization," ACM Symposium on Document Engineering (DocEng'09), New York, USA, 2009, pp. 13–20.

[93] J. Freund and J. Jones, Measuring and managing information risk: a FAIR approach. Newton, MA: Butterworth-Heinemann, August 2014.

[94] Gaius Suetonius Tranquillus, "Divus Julius - Chapter 56, Section 6," http://www.perseus.tufts.edu/.

[95] S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt, "Bringing Science to Digital Forensics with Standardized Forensic Corpora," Digital Investigation, Vol. 6, pp. 2–11, 2009.

[96] Gartner, Inc, "Gartner Forecasts Global Security and Risk Management Spending to Grow 14% in 2024," 2023, https://www.gartner.com/en/newsroom/press-releases/.

[97] L. A. Gordon, M. P. Loeb, L. Zhou et al., "Investing in cybersecurity: Insights from the gordon-loeb model," Journal of Information Security, Vol. 7, No. 02, p. 49, 2016.

[98] D. Granata and M. Rak, "Systematic analysis of automated threat modelling techniques: Comparison of open-source tools," Software Quality Journal, Vol. 32, No. 1, pp. 125–161, 2024.

[99] D. Granata, M. Rak, G. Salzillo, G. Di Guida, and S. Petrillo, "Automated threat modelling and risk analysis in e-Government using BPMN," Connection Science, Vol. 35, No. 1, p. 2284645, 2023.

[100] T. Grance, K. Kent, and B. Kim, "NIST Computer Security Incident Handling Guide," Special Publication (NIST SP): Gaithersburg, MD, USA, 2004.

[101] Q. Gu and P. Liu, "Denial of service attacks," Handbook of Computer Networks: Distributed Networks, Network Planning, Control, Management, and New Trends and Applications, Vol. 3, pp. 454–468, 2007.

[102] S. Gulmez, A. Gorgulu Kakisim, and I. Sogukpinar, "XRan: Explainable deep learning-based ransomware detection using dynamic analysis," Computers & Security, Vol. 139, p. 103703, 2024.

[103] J. Gumbley, "A Guide to Threat Modelling for Developers — martinfowler.com," https://martinfowler.com/articles/agile-threat-modelling.html.

[104] A. Gómez-Boix, D. Frey, Y.-D. Bromberg, and B. Baudry, "A Collaborative Strategy for Mitigating Tracking through Browser Fingerprinting," Proceedings of the 6th ACM Workshop on Moving Target Defense. New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 67–78.

[105] J. Habibi, A. Gupta, S. Carlsony, A. Panicker, and E. Bertino, "MAVR: Code Reuse Stealthy Attacks and Mitigation on Unmanned Aerial Vehicles," 2015 IEEE 35th International Conference on Distributed Computing Systems, 2015, pp. 642–652.

[106] E. Habler, R. Bitton, D. Avraham, D. Mimran, E. Klevansky, O. Brodt, H. Lehmann, Y. Elovici, and A. Shabtai, "Adversarial Machine Learning Threat Analysis and Remediation in Open Radio Access Network (O-RAN)," arXiv preprint 2201.06093, March 2023.

[107] S. Hacks, R. Lagerström, and D. Ritter, "Towards automated attack simulations of BPMN-based processes," 2021 IEEE 25th International Enterprise Distributed Object Computing Conference (EDOC), Gold Coast, Australia, 2021, pp. 182–191.

[108] A. Hammami, "The Art of Threat Modeling," Journal of Computer Sciences and Informatics, pp. 0–0, 2024.

[109] R. L. Harrison, "Introduction to Monte Carlo Simulation," AIP conference proceedings, Vol. 1204. NIH Public Access, 2010, p. 17.

[110] N. A. Hashim, Z. Z. Abidin, A. Puvanasvaran, N. A. Zakaria, and R. Ahmad, "Risk Assessment Method for Insider Threats in Cyber Security: A Review," International Journal of Advanced Computer Science and Applications, Vol. 9, No. 11, 2018.

[111] A. H. Hedaoo and A. Khandelwal, "Study of Dynamic Testing Techniques," International Journal, Vol. 7, No. 4, 2017.

[112] M. Henriksen, "Draw.io for threat modeling," October 2018, https://michenriksen.com/blog/drawio-for-threat-modeling/.

[113] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design Science in Information Systems Research," MIS quarterly, pp. 75–105, 2004.

[114] G. Hill, "diagrams.net-tutamen-entry-popup," January 2021, https://github.com/geoffrey-hill-tutamantic/diagrams.net-tutamen-entry-popup.

[115] J. Hochuli, "Design and Implementation of an Insider Threat Modeling System Using Business Process Models," Communication Systems Group, Department of Informatics, Zürich, Switzerland, jan 2024.

[116] J. Hochuli and J. von der Assen, "Insider Threat Modeling using BPMN," April 2024, https://github.com/jvdassen/InsiderThreatModeling.

[117] D. W. Hubbard and R. Seiersen, How to measure anything in cybersecurity risk. John Wiley & Sons, 2023.

[118] C. Hudon, M.-C. Chouinard, M. Bisson, A. Danish, M. Karam, A. Girard, P.-L. Bosse, and M. Lambert, "Case Study With a Participatory Approach," The Annals of Family Medicine, Vol. 19, No. 6, 2021.

[119] R. D. Hughbank and R. J. Hughbank, "Assessing Threats, Vulnerabilities, and Risks within the Community," Principles of Emergency Management and Emergency Operations Centers (EOC). CRC Press, 2021, pp. 117–141.

[120] S. Hunziker, Enterprise Risk Management: Modern Approaches to Balancing Risk and Reward. Springer, 2021.

[121] S. Hunziker and M. Blankenagel, "Research Design in Business and Management," Wiesbaden: SpringerGabler, Vol. 1, 2021.

[122] S. Hunziker, A. Portmann, V. Trachsel, and F. Dubler, "Cyber Risk Management in grösseren Schweizer Unternehmen," 2022.

[123] S. Hussain, A. Kamal, S. Ahmad, G. Rasool, and S. Iqbal, "Threat Modelling Methodologies: A Survey," Sci. Int.(Lahore), Vol. 26, pp. 1607–1609, 01 2014.

[124] E. M. Hutchins, M. J. Cloppert, R. M. Amin et al., "Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains," Leading Issues in Information Warfare & Security Research, Vol. 1, No. 1, p. 80, 2011.

[125] IBM, "Cost of a Data Breach Report 2024," 2024, https://www.ibm.com/reports/data-breach.

[126] IriusRisk, "Threat Modeling Platform," https://www.iriusrisk.com/threat-modeling-platform.

[127] H. Isabella, "10 Common Types of Malware Attacks and How to Prevent Them," 2023, https://www.techtarget.com/searchsecurity/tip/10-common-types-of-malware-attacks-and-how-to-prevent-them.

[128] ISACA, Risk IT Framework. ISACA, 2009.

[129] ISO, "ISO/IEC 27005:2018," ISO, Geneva, Tech. Rep., 2018, https://www.iso.org/standard/75281.html.

[130] ISO, "ISO/IEC 31000:2018," ISO, Geneva, Tech. Rep., 2018, https://www.iso.org/standard/65694.html.

[131] ISO, "ISO/IEC 27001:2022," Geneva, 2022, https://www.iso.org/standard/27001.

[132] Ł. Jalowski, M. Zmuda, and M. Rawski, "A Survey on Moving Target Defense for Networks: A Practical View," Electronics, Vol. 11, No. 18, p. 2886, 2022.

[133] M. Javaid, A. Haleem, R. P. Singh, and R. Suman, "Artificial intelligence applications for industry 4.0: A literature-based study," Journal of Industrial Integration and Management, Vol. 07, No. 01, pp. 83–111, 2022.

[134] JGraph, "GitHub - jgraph/drawio: Source to app.diagrams.net," January 2022, https://github.com/jgraph/drawio.

[135] JGraph Ltd, "Embed mode," November 2020, https://drawio.freshdesk.com/support/solutions/articles/16000042544-embed-mode.

[136] Joint Task Force, "Security and privacy controls for information systems and organizations," National Institute of Standards and Technology, Tech. Rep., 2017.

[137] Joint Task Force, "NIST SP 800-37, Revision 2," National Institute of Standards and Technology, Tech. Rep., 2018.

[138] Joint Task Force Transformation Initiative, Guide for Conducting Risk Assessments. National Institute of Standards & Technology, 2012.

[139] A. Jones, J. Kuehnert, P. Fraccaro, O. Meuriot, T. Ishikawa, B. Edwards, N. Stoyanov, S. L. Remy, K. Weldemariam, and S. Assefa, "AI for climate impacts: applications in flood risk," npj Climate and Atmospheric Science, Vol. 6, No. 1, p. 63, 2023.

[140] Jon Oberheide, "Brief Analysis of the Gawker Password Dump," 2010, https://duo.com/blog/brief-analysis-of-the-gawker-password-dump.

[141] Josh Fruhlinger, "The Heartbleed bug: How a flaw in OpenSSL caused a security crisis," 2022, https://www.csoonline.com/article/562859/the-heartbleed-bug-how-a-flaw-in-openssl-caused-a-security-crisis.html.

[142] C. Joshi, J. R. Aliaga, and D. R. Insua, "Insider threat modeling: An adversarial risk analysis approach," IEEE transactions on information forensics and security, Vol. 16, pp. 1131–1142, 2021.

[143] K. Hu, Reuters, "ChatGPT sets record for fastest-growing user base - analyst note," February 2023, https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/.

[144] F. Kammüller and C. W. Probst, "Modeling and verification of insider threats using logical analysis," IEEE systems journal, Vol. 11, No. 2, pp. 534–545, 2015.

[145] G. Karopoulos, J. L. Hernandez-Ramos, V. Kouliaridis, and G. Kambourakis, "A Survey on Digital Certificates Approaches for the COVID-19 Pandemic," IEEE Access, Vol. 9, 2021.

[146] C. Kern, "Preventing Security Bugs through Software Design," 24th USENIX Security Symposium, Washington, D.C., Aug. 2015.

[147] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," Cybersecurity, Vol. 2, No. 1, pp. 1–22, 2019.

[148] C. Killer, "Privacy and verifiability in decentralized electronic voting systems," Ph.D. dissertation, University of Zurich, 2023.

[149] C. Killer and B. Stiller, "The Swiss postal voting process and its system and security analysis," Electronic Voting: 4th International Joint Conference, E-Vote-ID 2019, Bregenz, Austria, 2019, pp. 134–149.

[150] G. Kim, K. Phelps, and D. Milne, "Prioritizing IT Controls for Effective, Measurable Security," IT Process Institute Research Study, 2006.

[151] J. King and A. I. Awad, "A distributed security mechanism for resource-constrained IoT devices," Informatica, Vol. 40, No. 1, 2016.

[152] A. Kucharavy, O. Plancherel, V. Mulder, A. Mermoud, and V. Lenders, "Large language models in cybersecurity: Threats, exposure and mitigation," 2024.

[153] L. Kyungroul, L. Sun-Young, and Y. Kangbin, "Machine Learning based File Entropy Analysis for Ransomware Detection in Backup Systems," IEEE Access, Vol. 7, pp. 110 205–110 215, 2019.

[154] C. Lei, D. H. Ma, and H. Q. Zhang, "Optimal strategy selection for moving target defense based on markov game," IEEE Access, Vol. 5, pp. 156–169, 2017.

[155] M. S. Lund, B. Solhaug, and K. Stølen, Model-driven risk analysis: The CORAS approach. Springer, 2010.

[156] M. Maksimović, V. Vujović, N. Davidović, V. Milošević, and B. Perišić, "Raspberry Pi as Internet of things hardware: performances and constraints," design issues, Vol. 3, No. 8, pp. 1–6, 2014.

[157] G. R. Marczyk, D. DeMatteo, and D. Festinger, Essentials of research design and methodology. John Wiley & Sons, 2010, Vol. 2.

[158] A. Marttinen, A. M. Wyglinski, and R. Jäntti, "Moving-target defense mechanisms against source-selective jamming attacks in tactical cognitive radio MANETs," 2014 IEEE Conference on Communications and Network Security, 2014, pp. 14–20.

[159] L. Mauri and E. Damiani, "Estimating Degradation of Machine Learning Data Assets," ACM Journal of Data and Information Quality (JDIQ), Vol. 14, No. 2, pp. 1–15, 2021.

[160] L. Mauri and E. Damiani, "Modeling threats to AI-ML systems using STRIDE," Sensors, Vol. 22, No. 17, 2022.

[161] G. McKinney, "TCP/IP State Transition Diagram (RFC793)," 1995.

[162] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," Artificial intelligence and statistics, 2017, pp. 1273–1282.

[163] T. McPeak, "Hard Truths Your CISO Won't Tell You," 2024, https://www.youtube.com/watch?v=uXFy91gs2Gk.

[164] N. Mead, F. Shull, J. Spears, S. Heibl, S. Weber, and J. Cleland-Huang, "Crowd Sourcing the Creation of Personae Non Gratae for Requirements-Phase Threat Modeling," 2017 IEEE 25th International Requirements Engineering Conference (RE), Lisbon, Portugal, 2017, pp. 412–417.

[165] Merriam-Webster.com, "Framework Definition & Meaning," October 2024, https://www.merriam-webster.com/dictionary/framework.

[166] Merriam-Webster.com, "Method Definition & Meaning," October 2024, https://www.merriam-webster.com/dictionary/method.

[167] Merriam-Webster.com, "Methodology Definition & Meaning," October 2024, https://www.merriam-webster.com/dictionary/methodology.

[168] N. Messe, V. Chiprianov, N. Belloir, J. El-Hachem, R. Fleurquin, and S. Sadou, "Asset-Oriented Threat Modeling," 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2020, pp. 491–501.

[169] Microsoft. Threat Modeling for AI/ML Systems and Dependencies. https://learn.microsoft.com/security/engineering/threat-modeling-aiml.

[170] Microsoft, "Microsoft Threat Modeling Tool," November 2020, https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool.

[171] Microsoft, "Microsoft Threat Modeling Tool threats," March 2022, https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats.

[172] M. Miora, "Quantifying the Business Impact Analysis," Carolina Computer News, 1996.

[173] Moody's Ratings, Inc., "Cyber budgets increase, executive overview improves, but challenges lurk under the surface," 2023, https://www.moodys.com/web/en/us/about/insights/data-stories/2023-cyber-survey-highlights.html.

[174] A. P. Moore, K. A. Kennedy, and T. J. Dover, "Introduction to the special issue on insider threat modeling and simulation," Computational and Mathematical Organization Theory, Vol. 22, pp. 261–272, 2016.

[175] T. Moore, "The economics of cybersecurity: Principles and policy options," International Journal of Critical Infrastructure Protection, Vol. 3, No. 3, pp. 103–117, 2010.

[176] E. Mossburg, J. Gelinne, and H. Calzada, "Beneath the surface of a cyberattack: A deeper look at business impacts," 2016.

[177] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization," 10th ACM workshop on artificial intelligence and security, 2017, pp. 27–38.

[178] Muyao Dong, "BIA Prototype," https://github.com/Dmmmmy/BIA-prototype.

[179] S. Myagmar, A. J. Lee, and W. Yurcik, "Threat Modeling as a Basis for Security Requirements," Symposium on Requirements Engineering for Information Security (SREIS), August 2005.

[180] National Institute of Standards and Technology (NIST), "Guide to Data-Centric System Threat Modeling," March 2016, https://csrc.nist.gov/publications/detail/sp/800-154/draft.

[181] R. E. Navas, F. Cuppens, N. Boulahia Cuppens, L. Toutain, and G. Z. Papadopoulos, "MTD, Where Art Thou? A Systematic Review of Moving Target Defense Techniques for IoT," IEEE Internet of Things Journal, Vol. 8, No. 10, pp. 7818–7832, 2021.

[182] NCSC, "Results of the COVID-19 certificate security tests," 2021, https://www.ncsc.admin.ch/ncsc/en/home/aktuell/im-fokus/covid-zertifikat-pst.html.

[183] J. A. Nichols, H. W. Herbert Chan, and M. A. Baker, "Machine learning: applications of artificial intelligence to imaging and diagnosis," Biophysical reviews, Vol. 11, pp. 111–118, 2019.

[184] NIST, "A profile for us federal cryptographic key management systems," 2015.

[185] NIST, "NIST Cybersecurity Framework," 2018, https://www.nist.gov/news-events/news/2018/04/nist-releases-version-11-its-popular-cybersecurity-framework.

[186] NIST, "Security and Privacy Controls for Information Systems and Organizations," September 2020, https://csrc.nist.gov/pubs/sp/800/53/r5/upd1/final.

[187] F. Nizzi, T. Pecorella, F. Esposito, L. Pierucci, and R. Fantacci, "IoT Security via Address Shuffling: The Easy Way," IEEE Internet of Things Journal, Vol. 6, No. 2, 2019.

[188] S. Nolen, C. Henry, T. Patrick, and B. K. evin R.B., "CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data," IEEE 36th international conference on distributed computing systems, pp. 303–312, 2016.

[189] N. Nostro, A. Ceccarelli, A. Bondavalli, and F. Brancati, "Insider Threat Assessment: A Model-Based Methodology," SIGOPS Oper. Syst. Rev., Vol. 48, No. 2, p. 3–12, 2014.

[190] H. S. Obaid and E. H. Abeed, "DoS and DDoS attacks at OSI layers," International Journal of Multidisciplinary Research and Publications, Vol. 2, No. 8, pp. 1–9, 2020.

[191] F. M. of Natural History, "Risk of Death – International Shark Attack File," 2021, https://www.floridamuseum.ufl.edu/shark-attacks/odds/compare-risk/death/.

[192] H. Okhravi, A. Comella, E. Robinson, S. Yannalfo, P. Michaleas, and J. Haines, "Creating a Cyber Moving Target for Critical Infrastructure Applications," Critical Infrastructure Protection V. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, Vol. 367, pp. 107–123.

[193] R. Oppliger and A. Grünert, "How to Manage Cyber Risks: Lessons Learned From Medical Science," Computer, Vol. 56, No. 1, pp. 117–119, 2023.

[194] OWASP, "OWASP Top 10: LLM & Generative AI Security Risks," 2023, https://genai.owasp.org/llmrisk/llm01-prompt-injection/.

[195] OWASP, "Threat Modeling - OASP Cheat Sheet Series," 2024, https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html.

[196] OWASP Foundation, "OWASP Internet of Things Project - OWASP," May 2020, https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project.

[197] OWASP Foundation, "OWASP Threat Dragon," November 2021, https://owasp.org/www-project-threat-dragon.

[198] OWASP Foundation, "AI Exchange," May 2024, https://owaspai.org/.

[199] OWASP Foundation, "OWASP SAMM – owaspsamm.org," 2024, https://owaspsamm.org/.

[200] OWASP® Foundation, "OWASP Top Ten," October 2021, https://owasp.org/www-project-top-ten/.

[201] C. Park, C. Kontovas, Z. Yang, and C.-H. Chang, "A BN driven FMEA approach to assess maritime cybersecurity risks," Ocean & Coastal Management, Vol. 235, 2023.

[202] L. Pavlík, M. Ficek, and J. Rak, "Dynamic Assessment of Cyber Threats in the Field of Insurance," Risks, Vol. 10, No. 12, 2022.

[203] T. R. Peltier, Information Security Risk Analysis, 3rd ed. Auerbach Publications, 2010.

[204] H. Per, P. H. Meland, D. Spampinato, E. Hagen, E. Baadshaug, K.-M. Krister, and K. Velle, "SeaMonster: Providing tool support for security modeling Per Hakon Meland," 09 2010.

[205] Y. Prajwal, J. V. Parekh, and R. Shettar, "A brief review of micro-frontends," United International Journal for Research and Technology, Vol. 2, No. 8, 2021.

[206] S. Pramanik, "Threat motivation," 2013 10th International Conference and Expo on Emerging Technologies for a Smarter World (CEWIT), 2013, pp. 1–5.

[207] N. Prasad and P. Lynggaard, "LoRaWan sensitivity analysis and prevention strategies against wireless DoS attacks," Wireless Personal Communications, Vol. 126, No. 4, pp. 3663–3675, 2022.

[208] D. Puthal, S. Nepal, R. Ranjan, and J. Chen, "DLSeF: A Dynamic Key-Length-Based Efficient Real-Time Security Verification Model for Big Data Stream," ACM Transactions on Embedded Computing Systems, Vol. 16, No. 2, 2016.

[209] R-fx Networks Projects, "Linux Malware Detect," https://www.rfxn.com/projects/linux-malware-detect/.

[210] G. Rajakumaran and N. Venkataraman, "Performance Assessment of Hybrid MTD for DoS Mitigation in Public Cloud," International Journal of Intelligent Networks, Vol. 2, pp. 140–147, 2021.

[211] S. Rajendran, R. Calvo-Palomino, M. Fuchs, B. Van den Bergh, H. Cordobes, D. Giustiniano, S. Pollin, and V. Lenders, "Electrosense: Open and big spectrum data," IEEE Communications Magazine, Vol. 56, No. 1, pp. 210–217, 2018.

[212] M. Rauer, "Draw.io Diagramming in Confluence Is Currently the Most Successful App in the Atlassian Marketplace," 2019 April, https://blog.seibert-media.com/2019/04/18/draw-io-diagramming-in-confluence-is-currently-the-most-successful-app/.

[213] R. M. Rechie, N. N. Mohamed, Y. M. Yussoff, L. Mazalan, S. M. Jawi, and M. S. Rohmad, "Integration of Security Hardware Module Zymkey 4i With Raspberry Pi," Journal of Positive School Psychology, Vol. 6, No. 3, pp. 8461–8469, 2022.

[214] Z. Rehman, I. Gondal, M. Ge, H. Dong, M. Gregory, and Z. Tari, "Proactive defense mechanism: Enhancing IoT security through diversity-based moving target defense and cyber deception," Computers & Security, Vol. 139, p. 103685, 2024.

[215] A. Roth, "Laboratory vs. field experiments: what can we learn?" 2006, comments on Levitt-List paper. Roundtable discussion,. Presented at the ASSA meetings, Boston MA. Retrieved from http://kuznets. fas. harvard. edu/~ aroth/papers/LabVersusField. Roundtable. A SSA.

[216] A. Roy, A. Chhabra, C. A. Kamhoua, and P. Mohapatra, "A Moving Target Defense against Adversarial Machine Learning," Proceedings of the 4th ACM/IEEE Symposium on Edge Computing. Arlington Virginia: ACM, 2019, pp. 383–388.

[217] M. Salnitri, F. Dalpiaz, and P. Giorgini, "Designing secure business processes with SecBPMN," Software and systems modeling, Vol. 16, No. 3, pp. 737–757, 2017.

[218] P. M. S. Sanchez, A. H. Celdran, G. Bovet, G. M. Perez, and B. Stiller, "SpecForce: A Framework to Secure IoT Spectrum Sensors in the Internet of Battlefield Things," IEEE Communications Magazine, pp. 1–7, 2022.

[219] P. M. S. Sánchez, J. M. J. Valero, A. H. Celdrán, G. Bovet, M. G. Pérez, and G. M. Pérez, "A survey on device behavior fingerprinting: Data sources, techniques, application scenarios, and datasets," IEEE Communications Surveys & Tutorials, Vol. 23, No. 2, pp. 1048–1077, 2021.

[220] R. S. Sangwan, Y. Badr, and S. M. Srinivasan, "Cybersecurity for AI Systems: A Survey," Journal of Cybersecurity and Privacy, Vol. 3, No. 2, pp. 166–190, 2023.

[221] E. Schiller, A. Aidoo, J. Fuhrer, J. Stahl, M. Ziörjen, and B. Stiller, "Landscape of IoT security," Computer Science Review, Vol. 44, p. 100467, 2022.

[222] N. Schmatloch, "Design and Implementation of a Malware to Detect Containerized Sandboxing," Communication Systems Group, Department of Informatics, Zürich, Switzerland, aug 2024, https://files.ifi.uzh.ch/CSG/staff/vonderassen/extern/theses/ba-schmatloch.pdf.

[223] C. Schneider, "Threagile - Agile Threat Modeling," 2020, https://threagile.io/.

[224] C. Schneider, "Threagile/threagile: Agile Threat Modeling," 2020, https://github.com/Threagile/threagile.

[225] B. Schneier, "The Future of Incident Response," IEEE Security & Privacy, Vol. 12, No. 5, pp. 96–96, 2014.

[226] N. Schwarz, R. M. Groves, and H. Schuman, "Survey methods," The handbook of social psychology, Vol. 1, pp. 143–179, 1998.

[227] SecurityCompass, "SD Elements: Your essential secure development solution - Security Compass," https://www.securitycompass.com/sdelements/.

[228] SecurityCompass, "SD Elements Datasheet v5.17," month year, https://docs.sdelements.com/release/latest/guide/docs/datasheet.html/.

[229] semperis, "2024 Ransomware Risk Report," https://www.semperis.com/wp-content/uploads/resources-pdfs/ransomware-report-2024.pdf.

[230] Z. Shi, K. Graffi, D. Starobinski, and N. Matyunin, "Threat modeling tools: A taxonomy," IEEE Security & Privacy, Vol. 20, No. 4, pp. 29–39, 2022.

[231] B. Shojaie, H. Federrath, and I. Saberi, "Evaluating the Effectiveness of ISO 27001: 2013 Based on Annex A," 2014 Ninth International Conference on Availability, Reliability and Security, 2014.

[232] A. Shostack, Threat Modeling: Designing for Security. John Wiley & Sons, 2014.

[233] C. Simmons, C. Ellis, S. Shiva, D. Dasgupta, and Q. Wu, "AVOIDIT: A cyber attack taxonomy," University of Memphis, Technical Report CS-09-003, 2009.

[234] SINTEF, "CORAS - A risk modelling approach," https://coras.tools/#/.

[235] H. Snyder, "Literature review as a research methodology: An overview and guidelines," Journal of Business Research, Vol. 104, pp. 333–339, 2019.

[236] D. Srinivasan, A. Muthuvel, M. Prakash Kumar, R. Dinesh, and V. G. Prasannakumar, "Advanced Malware Analysis and Prevention," 2023 Third International Conference on Smart Technologies, Communication and Robotics (STCR), Vol. 1, 2023, pp. 1–6.

[237] A. Srivastava, B. B. Gupta, A. Tyagi, A. Sharma, and A. Mishra, "A Recent Survey on DDoS Attacks and Defense Mechanisms," Advances in Parallel Distributed Computing, 2011, pp. 570–580.

[238] State Secretariat for Economic Affairs (SECO), " Figures on SMEs: Companies and jobs ," 2021, https://www.kmu.admin.ch/kmu/en/home/concrete-know-how/facts-and-figures/figures-smes/companies-and-jobs.html/.

[239] Y. Sun, W. Ji, J. Weng, B. Zhao, Y. Li, and X. Wu, "Selection of Optimal Strategy for Moving Target Defense Based on Signal Game," ACM International Conference Proceeding Series, pp. 28–32, 2020.

[240] M. Swanson, P. Bowen, A. W. Phillips, D. Gallup, and D. Lynes, "Contingency planning guide for federal information systems," NIST special publication, pp. 800–34, 2010i.

[241] swissinfo. ch/keystone Sda/urs, "Local election official found guilty of fraud," SWI swissinfo, Jan. 2024, https://www.swissinfo.ch/eng/politics/local-election-official-found-guilty-of-fraud/46767148.

[242] swissinfo.ch, "Thousands of fake Covid-19 certificates uncovered in Switzerland," December 2023, https://www.swissinfo.ch/eng/identities/thousands-of-fake-covid-19-certificates-uncovered-in-switzerland/47214700.

[243] P. M. Sánchez Sánchez, E. Tomás Martínez Beltrán, A. H. Celdrán, R. Wassink, G. Bovet, G. M. Pérez, and B. Stiller, "Stealth Spectrum Sensing Data Falsification Attacks Affecting IoT Spectrum Monitors on the Battlefield," MILCOM 2023 - 2023 IEEE Military Communications Conference (MILCOM), 2023, pp. 673–678.

[244] J. l. Tan, C. Lei, H. q. Zhang, and Y. q. Cheng, "Optimal Strategy Selection Approach to Moving Target Defense based on Markov Robust Game," Computers & Security, Vol. 85, pp. 63–76, August 2019.

[245] J. Tan, H. Zhang, H. Zhang, H. Hu, C. Lei, and Z. Qin, "Optimal Temporospatial Strategy Selection Approach to Moving Target Defense: A FlipIt Differential Game Model," Computers & Security, Vol. 108, pp. 283–294, 2021.

[246] K. Tan and V. Garg, "An analysis of open-source automated threat modeling tools and their extensibility from security into privacy," 2022, https://www.usenix.org/publications/loginonline/analysis-open-source-automated-threat-modeling-tools-and-their.

[247] I. Tarandach, "pytm: A Pythonic framework for threat modeling," October 2021, https://github.com/izar/pytm.

[248] I. Tarandach and M. J. Coles, Threat Modeling. " O'Reilly Media, Inc.", 2020.

[249] The Apache Software Foundation, "Apache JMeter," https://jmeter.apache.org/.

[250] The gVisor Authors, "The Container Security Platform - gVisor — gvisor.dev," https://gvisor.dev/.

[251] The MITRE Corporation, "MITRE ATLAS (Adversarial Threat Landscape for Artificial-Intelligence Systems)," https://atlas.mitre.org/.

[252] M. Thompson, N. Evans, and V. Kisekka, "Multiple OS rotational environment an implemented Moving Target Defense," 2014 7th International Symposium on Resilient Control Systems (ISRCS), 2014, pp. 1–6.

[253] Threat Modeling Manifesto Working Group, "Threat Modeling Manifesto," November 2020, https://www.threatmodelingmanifesto.org.

[254] ThreatModeler Software, Inc., "Intelligent Threat Engine," https://threatmodeler.com/automated-threat-modeling-tool/.

[255] Threatspec, "Threatspec," June 2019, https://threatspec.org/.

[256] Tms, "Word template - threats manager studio," 2024, https://threatsmanager.com/training/advanced/reporting/word-template/.

[257] S. Torabi, H. Rezaei Soufi, and N. Sahebjamnia, "A new framework for business impact analysis in business continuity management (with a case study)," Safety Science, Vol. 68, pp. 309–323, 2014.

[258] S. Traer and P. Bednar, "Motives Behind DDoS Attacks," Digital Transformation and Human Behavior, Cham, 2021, pp. 135–147.

[259] Tutamantic, "Feauture | Tutamantic," January 2021, https://www.tutamantic.com/page/features.

[260] T. Ucedavélez and M. M. Morana, Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis. John Wiley & Sons, 2015.

[261] United States Securities and Exchange Commission, "SEC Adopts Rules on Cybersecurity Risk Management, Strategy, Governance, and Incident Disclosure by Public Companies," 2023, https://eur-lex.europa.eu/eli/dir/2022/2555.

[262] M. Vannini, "CoReTM 2.0: a Design Study on Semi-automated Threat Modeling with STRIDE-per-Interaction," Communication Systems Group, Department of Informatics, Zürich, Switzerland, sep 2024, https://files.ifi.uzh.ch/CSG/staff/vonderassen/extern/theses/ba-vannini.pdf.

[263] M. Vannini, J. Messerer, and J. von der Assen, "CoReTM-2.0: Semi-automated threat modeling using the STRIDE-per-Interaction methodology — github.com," 2024, https://github.com/messi1/CoReTM-2.0.

[264] I. Voce and A. Morgan, Cybercrime in Australia 2023. Australian Institute of Criminology, 2023.

[265] J. von der Assen, "Ransomare-Testbed," Dec. 2024, https://github.com/Cyber-Tracer/Ransomare-Testbed.

[266] J. von der Assen, A. H. Celdrán, J. Luechinger, P. M. S. Sánchez, G. Bovet, G. M. Pérez, and B. Stiller, "Ransomai: Ai-powered ransomware for stealthy encryption," GLOBECOM 2023-2023 IEEE Global Communications Conference. IEEE, 2023, pp. 2578–2583.

[267] J. von der Assen, A. H. Celdrán, P. M. S. Sánchez, J. Cedeño, G. Bovet, G. M. Pérez, and B. Stiller, "A lightweight moving target defense framework for multi-purpose malware affecting iot devices," ICC 2023-IEEE International Conference on Communications. IEEE, 2023, pp. 6010–6015.

[268] J. von der Assen, A. H. Celdrán, R. Sefa, B. Stiller, and G. Bovet, "MTFS: a Moving Target Defense-Enabled File System for Malware Mitigation," 2024 IEEE 49th Conference on Local Computer Networks (LCN). IEEE, 2024, pp. 1–6.

[269] J. von der Assen, A. H. Celdran, J. Sharif, C. Feng, G. Bovet, and B. Stiller, "ThreatFinderAI: Automated Threat Modeling Applied to LLM System Integration," 2024 20th International Conference on Network and Service Management (CNSM), Prague, Czech Republic, oct 2024, pp. 1–3.

[270] J. Von der Assen, A. H. Celdrán, N. Huber, G. Bovet, G. M. Pérez, and B. Stiller, "Moving Target Defense Strategy Selection against Malware in Resource-Constrained Devices," 2023 IEEE International Conference on Cyber Security and Resilience (CSR), 2023, pp. 123–129.

[271] J. von der Assen, A. H. Celdrán, A. Zermin, R. Mogicato, G. Bovet, and B. Stiller, "SecBox: A Lightweight Container-based Sandbox for Dynamic Malware Analysis," NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium, 2023, pp. 1–3.

[272] J. von der Assen, C. Feng, A. H. Celdran, R. Oles, G. Bovet, and B. Stiller, "GuardFS: a File System for Integrated Detection and Mitigation of Linux-based Ransomware," 2024, https://arxiv.org/abs/2401.17917.

[273] J. von der Assen, M. F. Franco, M. Dong, and B. Stiller, "Quanttm: Business-centric threat quantification for risk management and cyber resilience," 2024, https://arxiv.org/abs/2402.14140.

[274] J. Von Der Assen, M. F. Franco, C. Killer, E. J. Scheid, and B. Stiller, "CoReTM: An Approach Enabling Cross-Functional Collaborative Threat Modeling," 2022 IEEE International Conference on Cyber Security and Resilience (CSR), Virtual, 2022, pp. 189–196.

[275] J. von der Assen, M. F. Franco, C. Killer, E. J. Scheid, and B. Stiller, "On Collaborative Threat Modeling," IFI-TecReport No. 2022.04, Department of Informatics, Zürich, Switzerland, Tech. Rep., apr 2022, https://files.ifi.uzh.ch/CSG/staff/vonderassen/extern/publications/IfI-TR-2022_04.pdf.

[276] J. von der Assen, M. F. Franco, C. Killer, E. J. Scheid, and B. Stiller, "On collaborative threat modeling," IFI-TecReport No. 2022.04, Zürich, Switzerland, Tech. Rep., 2022.

[277] J. von der Assen, J. Hochuli, T. Grübl, and B. Stiller, "The danger within: Insider threat modeling using business process models," 2024 IEEE International Conference on Cyber Security and Resilience (CSR), 2024, pp. 186–192.

[278] J. von der Assen and J. Sharif, "Demonstration: Threat Modeling using ThreatFinderAI ," 2024, https://youtu.be/cPt8cyAjtmQ.

[279] J. von der Assen and J. Sharif, "ThreatFinder," 2024, https://threatfinder-ai.pages.dev/.

[280] J. von der Assen, J. Sharif, C. Feng, C. Killer, G. Bovet, and B. Stiller, "Asset-Centric Threat Modeling for AI-Based Systems," 2024 IEEE International Conference on Cyber Security and Resilience (CSR). IEEE, 2024, pp. 437–444.

[281] A. Voulimeneas, D. Song, F. Parzefall, Y. Na, P. Larsen, M. Franz, and S. Volckaert, "Distributed Heterogeneous N-Variant Execution," Detection of Intrusions and Malware, and Vulnerability Assessment, ser. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 217–237.

[282] S. Vuppala, A. E.-D. Mady, and A. Kuenzi, "Moving Target Defense Mechanism for Side-Channel Attacks," IEEE Systems Journal, Vol. 14, pp. 1810–1819, 2020.

[283] A. Walker and S. Sengupta, "Insights into Malware Detection via Behavioral Frequency Analysis Using Machine Learning," MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM), Norfolk, VA, USA, 2019, pp. 1–6.

[284] C. Wang and Z. Lu, "Cyber deception: Overview and the road ahead," IEEE Security & Privacy, Vol. 16, No. 2, pp. 80–85, 2018.

[285] J. Wang, M. Neil, and N. Fenton, "A Bayesian network approach for cybersecurity risk assessment implementing and extending the FAIR model," Computers & Security, Vol. 89, 2020.

[286] X. Wang, J. Du, and H. Liu, "Performance and isolation analysis of RunC, gVisor and Kata Containers runtimes," Cluster Computing, Vol. 25, No. 2, pp. 1497–1513, 2022.

[287] E. Wheeler, Security Risk Management: Building an Information Security Risk Management Program from the Ground Up. Elsevier, 2011.

[288] C. Wilhjelm and A. A. Younis, "A Threat Analysis Methodology for Security Requirements Elicitation in Machine Learning Based Systems," 2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C), 2020, pp. 426–433.

[289] C. Wilkinson, "Potential National Security Threat Created by Cyber Attackers Infiltrating Unsecured Smarthome Devices," 2019, https://www.usu.edu/cai/files/studentpaper-wilkinson.pdf.

[290] World Economic Forum, "Global Cybersecurity Outlook 2024," 2024, https://www.weforum.org/publications/global-cybersecurity-outlook-2024/.

[291] R. Wäspi, "Quantitative Maturity Assessment of DevSecOps Practices Using Metrics," Communication Systems Group, Department of Informatics, Zürich, Switzerland, aug 2024, https://files.ifi.uzh.ch/CSG/staff/vonderassen/extern/theses/is-waespi.pdf.

[292] W. Xiong and R. Lagerström, "Threat modeling–A systematic literature review," Computers & security, Vol. 84, pp. 53–69, 2019.

[293] ZABALA group. (2023) Aisym4med project. https://aisym4med.eu/about-the-project/.

[294] S. Zareen, A. Akram, and S. Ahmad Khan, "Security Requirements Engineering Framework with BPMN 2.0.2 Extension Model for Development of Information Systems," Applied Sciences, Vol. 10, No. 14, 2020.

[295] K. Zeitz, M. Cantrell, R. Marchany, and J. Tront, "Changing the Game: A Micro Moving Target IPv6 Defense for the Internet of Things," IEEE Wireless Communications Letters, Vol. 7, No. 4, pp. 578–581, 2018.

[296] M. Zeng, C. Dian, and Y. Wei, "Risk Assessment of Insider Threats Based on IHFACS-BN," Sustainability (Basel, Switzerland), Vol. 15, No. 1, p. 491, 2023.

[297] Z. Zeng and E. Zio, "An Integrated Modeling Framework for Quantitative Business Continuity Assessment," Process Safety and Environmental Protection, Vol. 106, pp. 76–88, 2017.

[298] H. Zhang, J. Tan, X. Liu, and J. Wang, "Moving Target Defense Decision-Making Method: A Dynamic Markov Differential Game Model," MTD 2020 - Proceedings of the 7th ACM Workshop on Moving Target Defense, pp. 21–29, 2020.

[299] H. Zhang, K. Zheng, X. Wang, S. Luo, and B. Wu, "Efficient Strategy Selection for Moving Target Defense Under Multiple Attacks," IEEE Access, Vol. 7, pp. 65 982–65 995, 2019.

[300] ZHAW, "Prof. Dr. Nico Ebert | ZHAW Zürcher Hochschule für Angewandte Wissenschaften — zhaw.ch," https://www.zhaw.ch/de/ueber-uns/person/ebet.

All links provided above were last accessed on February 4, 2025.

A Publications

Research conducted during the affiliation of this PhD thesis is based on several collaborations, projects, and interactions, as disseminated through the following publications. A set of publications directly related to the previously presented studies are enumerated in the following. Secondly, the complete list of publications contributing to the information security field is reported. For these publications and this Ph.D. thesis, no artificial intelligence was used in the creation of figures, code, or other substantive content. AI assistance was limited to linguistic support, such as discovering new vocabulary or identifying errors, similar to traditional proofreading.

A.1 Contributing Manuscripts

The following publications and theses were directly established, conceptualized, or coordinated by the thesis' author in the scope of this thesis report.

Publications

1. (Journal) **Jan von der Assen**, Chao Feng, Alberto Huertas Celdrán, Róbert Oleš, Gérôme Bovet, Burkhard Stiller: *GuardFS: a File System for Integrated Detection and Mitigation of Linux-based Ransomware*, Elsevier, Journal of Information Security and Applications, Vol. 93, No. 1, September 2025.

2. (Demonstration) **Jan von der Assen**, Alberto Huertas Celdrán, Jamo Sharif, Chao Feng, Gérôme Bovet, Burkhard Stiller: *ThreatFinderAI: Automated Threat Modeling Applied to LLM System Integration*, 20th International Conference on Network and Service Management (CNSM), Prague, Czech Republic, October, 2024

3. (Short Paper) **Jan von der Assen**, Alberto Huertas Celdrán, Rinor Sefa, Burkhard Stiller, Gérôme Bovet: *MTFS: A Moving Target Defense-Enabled File System for Malware Mitigation*; 2024 IEEE 49th Conference on Local Computer Networks (LCN), Caen, France, October 2024, pp 1-6

4. (Full Paper) **Jan von der Assen**, Jamo Sharif, Chao Feng, Christian Killer, Gérôme Bovet, Burkhard Stiller: *Asset-centric Threat Modeling for AI-based Systems*; 2024 IEEE International Conference on Cyber Security and Resilience (CSR), London, England, September 2024, pp 1-8

5. (Full Paper) **Jan von der Assen**, Jasmin Hochuli, Thomas Grübl, Burkhard Stiller: *The Danger Within: Insider Threat Modeling Using Business Process Models*, 2024 IEEE International Conference on Cyber Security and Resilience (CSR), London, England, September 2024, pp 1–6 (To appear)

6. (Full Paper) **Jan von der Assen**, Alberto Huertas Celdrán, Janik Luechinger, Pedro Miguel Sánchez Sánchez, Gérôme Bovet, Gregorio Martínez Pérez, Burkhard Stiller: *RansomAI: AI-powered Ransomware for Stealthy Encryption*; 2024 Cybersecurity Conference (JNIC), Sevilla, Spain, May 2024, pp. 1-2

7. (Full Paper) Chao Feng, **Jan von der Assen**, Alberto Huertas Celdran, Raffael Mogicato, Adrian Zermin, Vichhay Ok, Gérôme Bovet, Burkhard Stiller: *A Lightweight Data Mining Platform for Dynamic and Reproducible Malware Analysis*, 2024 11th Swiss Conference on Data Science (SDS), Zürich, Switzerland, May 2024, pp. 1-6

8. (Full Paper) **Jan von der Assen**, Alberto Huertas Celdrán, Janik Luechinger, Pedro Miguel Sánchez Sánchez, Gérôme Bovet, Gregorio Martínez Pérez, Burkhard Stiller: *RansomAI: AI-powered Ransomware for Stealthy Encryption*; 2023 IEEE Global Communications Conference, Kuala Lumpur, Malaysia, December 2023, pp. 1-6.

9. (Full Paper) **Jan von der Assen**, Alberto Huertas Celdrán, Nicolas Huber, Gérôme Bovet, Gregorio Martínez Pérez, Burkhard Stiller: *Moving Target Defense Strategy Selection against Malware in Resource-Constrained Devices*; IEEE International Conference on Cyber Security and Resilience, Venice, Italy, July 2023, pp. 1–7

10. (Demo Paper) **Jan von der Assen**, Alberto Huertas Celdrán, Adrian Zermin, Raffael Mogicato, Gérôme Bovet, Burkhard Stiller: *SecBox: A Lightweight Container-based Sandbox for Dynamic Malware Analysis*; IEEE/IFIP Network Operations and Management Symposium (NOMS 2023), Miami, USA, May 2023, pp. 1-3

11. (Full Paper) **Jan von der Assen**, Alberto Huertas Celdrán, Pedro Miguel Sánchez Sánchez, Jordan Cedeño, Gérôme Bovet, Gregorio Martínez Pérez, Burkhard Stiller: *A Lightweight Moving Target Defense Framework for Multi-purpose Malware Affecting IoT Devices*; IEEE International Conference on Communications, Rome, Italy, May 2023, pp. 1-6

12. (Full Paper) **Jan von der Assen**, Muriel F. Franco, Christian Killer, Eder J. Scheid, Burkhard Stiller: *CoReTM: An Approach Enabling Cross-Functional Collaborative Threat Modeling*; IEEE International Conference on Cyber Security and Resilience, Virtually, Europe, July 2022, pp. 1–8.

13. (Technical Report) **Jan von der Assen**, Muriel F. Franco, Christian Killer, Eder J. Scheid, Burkhard Stiller: *On Collaborative Threat Modeling*; UZH, IFI-TecReport No. 2022.04, Zürich, Switzerland, April 2022.

Under Review

14. (Preprint) **Jan von der Assen**, Muriel Franco, Muyao Dong, Burkhard Stiller: *QuantTM: Business-Centric Threat Quantification for Risk Management and Cyber Resilience*, arXiv e-prints (2024): arXiv:2402, February 2024

Supervised Theses

15. (Master Project) Jasmin Hochuli, Lukas Sager: *Threat Identification and Mitigation using Process Models*; Communication Systems Group, Department of Informatics, Zürich, Switzerland, August 2025.

16. (Master Thesis) Raphael Wäspi: *Design and Evaluation of a Large Language Model-powered Threat Modeling Approach with Applications to AI Security*; Communication Systems Group, Department of Informatics, Zürich, Switzerland, March 2025.

17. (Bachelor Thesis) Miro Vannini: *CoReTM 2.0: a Design Study on Semi-automated Threat Modeling with STRIDE-per-Interaction*; Communication Systems Group, Department of Informatics, Zürich, Switzerland, September 2024.

18. (Independent Study) Raphael Wäspi: *Quantitative Maturity Assessment of DevSecOps Practices Using Metrics*; Communication Systems Group, Department of Informatics, Zürich, Switzerland, August 2024.

19. (Bachelor Thesis) Niklas Schmatloch: *Design and Implementation of a Malware to Detect Containerized Sandboxing*; Communication Systems Group, Department of Informatics, Zürich, Switzerland, August 2024.

20. (Bachelor Thesis) Jasmin Hochuli: *Design and Implementation of an Insider Threat Modeling System Using Business Process Models*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, January 2024.

21. (Bachelor Thesis) Jamo Sharif: *Design and Implementation of a Threat Modeling Approach for AI-based Systems*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, December 2023.

22. (Bachelor Thesis) Viachaslau Berasneu: *Design and Implementation of a System for Reproducible Machine and Deep Learning Models*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, September 2023.

23. (Master Thesis) Muyao Dong: *Design and Implementation of a Business-driven Threat Quantification Framework*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, September 2023.

24. (Bachelor Thesis) Vichhay Ok: *Design and Implementation of a Reproducible and Realistic Data Collection System for Dynamic Malware Analysis*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, August 2023.

25. (Master Thesis) Róbert Oles: *Detection and Classification of Malware using File System Dimensions for MTD on IoT*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, May 2023.

26. (Bachelor Thesis) Maximilian Huwyler: *Design and Implementation of a Comparison Tool for Selecting an Information Security Risk Assessment Method*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, March 2023.

27. (Master Thesis) Janik Lüchinger: *AI-powered Ransomware to Optimize its Impact on IoT Spectrum Sensors*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, March 2023.

28. (Master Project) Raffael Mogicato, Adrian Zermin: *Design and Implementation of a Collaborative, Lightweight Malware Analysis Sandbox using Container Virtualization*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, February 2023.

29. (Master Thesis) Rinor Sefa: *Design and Implementation of a Virtual File System for Hostbased Moving Target Defence in IoT Devices*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, October 2022.

30. (Master Thesis) Timo Schenk: *Optimizing MTD Deployment on IoT Devices using Reinforcement Learning*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, October 2022.

31. (Bachelor Thesis) Nicolas Huber: *Design and Implementation of an MTD Strategy Selection Agent using IoT Platform Metrics*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, September 2022.

32. (Bachelor Thesis) Jordan Cedeño: *Mitigating Cyberattacks Affecting Resource-constrained Devices Through Moving Target Defense (MTD) Mechanisms*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, April 2022.

33. (Bachelor Thesis) Konstantin Moser: *Intelligent and Behavioral-based Detection of Cryptominers in Resource-constrained Spectrum Sensors*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, April 2022.

A.2 Related Manuscripts

Publications

34. (Full Paper) Chao Feng, Alberto Huertas Celdran, Zien Zeng, Zi Ye, **Jan von der Assen**, Gerome Bovet, Burkhard Stiller: *Leveraging MTD to Mitigate Poisoning Attacks in Decentralized FL with Non-IID Data*; 2024 IEEE International Conference on Big Data (IEEE BigData 2024), Washington DC, USA, December 2024, pp 7745–7754.

35. (Full Paper) Alberto Huertas Celdrán, **Jan von der Assen**, Chao Feng, Sandro Padovan, Gérôme Bovet, Burkhard Stiller: *Next Generation of AI-based Ransomware*; 2024 IEEE Global Communications Conference, Cape Town, South Africa, December 2024, pp 1–6.

36. (Preprint) Chao Feng, Hongjie Guan, Alberto Huertas Celdrán, **Jan von der Assen**, Gérôme Bovet, Burkhard Stiller: *FedEP: Tailoring Attention to Heterogeneous Data Distribution with Entropy Pooling for Decentralized Federated Learning*; arXiv, Preprint, Zürich, Switzerland, October 2024

37. (Preprint) Chao Feng, Alberto Huertas Celdran, Pedro Miguel Sanchez Sanchez, Jan Kreischer, **Jan von der Assen**, Gerome Bovet, Gregorio Martinez Perez, Burkhard Stiller: *CyberForce: A Federated Reinforcement Learning Framework for Malware Mitigation*; arxiv preprint, Zürich, Switzerland, October 2024.

38. (Journal) Chao Feng, Alberto Huertas Celdran, **Jan von der Assen**, Enrique Tomas Martinez Beltran, Gérôme Bovet, Burkhard Stiller: *DART: A Solution for Decentralized Federated Learning Model Robustness Analysis*; Elsevier, Array, Vol. 1, No. 1, August 2024.

39. (Book Chapter) **Jan von der Assen**, Alberto Huertas Celdrán, Josip Harambasic, Gérôme Bovet, Burkhard Stiller: *Enhancing Cyber Resilience in IoT: Moving Target Defense for Dynamic Malware Mitigation*, in A Complete Handbook on Defending Internet of Things Ecosystems: Challenges and Solution, Elsevier (To appear)

40. (Preprint) **Jan von der Assen**, Sandrin Raphael Hunkeler, Alberto Huertas Celdrán, Enrique Tomás Martínez Beltrán, Gérôme Bovet, Burkhard Stiller: *Reputation System based on Distributed Ledge to Secure Decentralized Federated Learning*, Research Square, October 2024.

41. (Full Paper) Alberto Huertas Celdrán, **Jan von der Assen**, Chao Feng, Sandro Padovan, Gérôme Bovet, Burkhard Stiller: *Next Generation of AI-based Ransomware*, 2024 IEEE Global Communications Conference: Communication & Information Systems Security, Cape Town, South Africa, December, 2024 (To appear)

42. (Preprint) Thomas Grübl, **Jan von der Assen**, Markus Knecht, Burkhard Stiller: *PACCOR4ESP: Embedded Device Security Attestation using Platform Attribute Certificates*, arXiv e-prints (2024): arXiv:2407, July 2024

43. (Journal) Chao Feng, Alberto Huertas Celdrán, **Jan von der Assen**, Enrique Tomás Martínez Beltrán, Gérôme Bovet, Burkhard Stiller, *DART: A Solution for decentralized federated learning model robustness analysis*, Elsevier Array, July 2024

44. (Journal) Alberto Huertas Celdrán, Pedro Miguel Sánchez Sánchez, **Jan von der Assen**, Timo Schenk, Gérôme Bovet, Gregorio Martínez Pérez, Burkhard Stiller: *RL and Fingerprinting to Select Moving Target Defense Mechanisms for Zero-day Attacks in IoT*, IEEE Transactions on Information Forensics and Security, May 2024

45. (Full Paper) **Jan von der Assen**, Christian Killer, Alessandro De Carli, Burkhard Stiller: *Performance Analysis of Decentralized Physical Infrastructure Networks and Centralized Clouds*, 2024 IEEE International Workshop on Decentralized Physical Infrastructure Networks, Dublin, Ireland, May 2024, pp. 1-6.

46. (Journal) Muriel Franco, Fabian Künzler, **Jan von der Assen**, Chao Feng, Burkhard Stiller: *RCVaR: an Economic Approach to Estimate Cyberattacks Costs using Data from Industry Reports*, Computers & Security, April 2024

47. (Journal) Alberto Huertas Celdrán, Pedro Miguel Sánchez Sánchez, **Jan von der Assen**, Dennis Shushack, Ángel Luis Perales Gómez, Gérôme Bovet, Gregorio Martínez Pérez, Burkhard Stiller: *Behavioral fingerprinting to detect ransomware in resource-constrained devices*, Computers & Security, Vol. 135, 2023

48. (Full Paper) Chao Feng, **Jan von der Assen**, Alberto Huertas Celdrán, Steven Näf, Gérôme Bovet, Burkhard Stiller: *FeDef: A Federated Defense Framework Using Cooperative Moving Target Defense*; 2023 8th International Conference on Smart and Sustainable Technologies (SpliTech), Bol/Split, Croatia, June 2023, pp. 1-6

49. (Full Paper) Alberto Huertas Celdrán, **Jan von der Assen**, Konstantin Moser, Pedro Miguel Sánchez Sánchez, Gérôme Bovet, Gregorio Martínez Pérez, Burkhard Stiller: *Early Detection of Cryptojacker*

Malicious Behaviors on IoT Crowdsensing Devices; IEEE/IFIP Network Operations and Management Symposium (NOMS 2023), Miami, USA, May 2023, pp. 1-8

50. (Full Paper) Katharina O. E. Müller, **Jan von der Assen**, Chao Feng, Burkhard Stiller: *An Overview and Ontology of Privacy to Preserve Privacy in Ultra-Wideband Networks*, IEEE International Conference on Privacy Computing, Haikou, China, December 2022, pp. 1-9.

51. (Dataset) Ramon Solo de Zaldivar, Alberto Huertas Celdrán, **Jan von der Assen**, Pedro Miguel Sánchez Sánchez, Gérôme Bovet, Gregorio Martínez Pérez, Burkhard Stiller: *MalwSpecSys: A Dataset Containing Syscalls of an IoT Spectrum Sensor Affected by Heterogeneous Malware*, IEEE Dataport, May 2022 .

52. (Full Paper) Christian Killer, Moritz Eck, Bruno Rodrigues, **Jan von der Assen**, Roger Staubli and Burkhard Stiller: *ProvotuMN: Decentralized, Mix-Net-based, and Receipt-free Voting System*; 2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), 2022, pp. 1-9.

53. (Technical Report) Ile Cepilov, Muriel Franco, Eder J. Scheid, Christian Killer, **Jan von der Assen**, Burkhard Stiller: *CCount: Correlating RFID and Camera Data for High Precision Indoor Tracking*; University of Zurich, Department of Informatics, Tech. Rep, 1. pp. 1 - 12.

54. (Demo Paper) **Jan von der Assen**, Muriel Franco, Bruno Rodrigues, Burkhard Stiller: *Analysis and Classification of Cyberattack Traffic Using the SecGrid Platform*; IEEE 46th Conference on Local Computer Networks (LCN) - Demo Session, Edmond, Canada, October 2021, pp 1–3.

55. (Full Paper) Muriel F. Franco, **Jan von der Assen**, Luc Boillat, Christian Killer, Bruno Rodrigues, Eder J. Scheid, Lisandro Granville, Burkhard Stiller: *SecGrid: A Visual System for the Analysis and ML-Based Classification of Cyberattack Traffic*; IEEE 46th Conference on Local Computer Networks (LCN 2021), Virtually, Edmonton, Canada, October 2021, pp. 1–8.

56. (Short Paper) Muriel F. Franco, **Jan von der Assen**, Luc Boillat, Christian Killer, Bruno Rodrigues, Eder J. Scheid, Lisandro Granville, Burkhard Stiller, Bruno Rodrigues, Burkhard Stiller: *DDoSGrid: a Platform for the Post-mortem Analysis and Visualization of DDoS Attacks*; IFIP Networking 2021, Virtual, June, 2021, pp. 1–3.

Supervised Theses

57. (Bachelor Thesis) Valentin Meyer: *Integration of a Blockchain-based Supply Chain Tracing Application*; Communication Systems Group, Department of Informatics, Zürich, Switzerland, August 2024.

58. (Bachelor Thesis) Jan Marc Lüthi: *HyperDtct: Hypervisor-Based Ransomware Detection*; Communication Systems Group, Department of Informatics, Zürich, Switzerland, July 2024.

59. (Bachelor Thesis) Sandrin Hunkeler: *Distributed Ledger-based Gossiping for Decentralized Federated Learning*; Communication Systems Group, Department of Informatics, Zürich, Switzerland, June 2024.

60. (Bachelor Thesis) Jano Vukadinovic: *Automatic Testing of Medical Equipment using Sensor-based Spectral Analysis*; Unviersität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, May 2024.

61. (Master Thesis) Sandro Padovan: *AI-powered Ransomware to Stay Hidden*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, January 2024.

62. (Bachelor Thesis) Jonathan Contreras: *Location-based Open Source Intelligence to Infer Information in LoRa Networks*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, July 2023.

63. (Master Project) Dario Gagulic, Lynn Zumtaugwald, Siddhant Sahu: *Ransomware Detection with Machine Learning in Storage Systems*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, February 2023.

64. (Bachelor Thesis) Josip Harambašić: *Design and Implementation of Moving Target Defense Techniques to Break the Cyber Kill Chain in IoT Devices*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, February 2023.

65. (Master Thesis) Steven Näf: *Design and Implementation of a Cooperative MTD Framework for IoT Devices*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, January 2023.

66. (Master Thesis) Timo Schenk: *Optimizing MTD Deployment on IoT Devices using Reinforcement Learning*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, October 2022.

67. (Master Thesis) Flavia Fulea: *Toward Collaborative and Cooperative Cybersecurity: A Survey on Approaches, Challenges, and Opportunities*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, August 2022.

68. (Bachelor Thesis) David Diener: *Design and Implementation of a Database-to-Blockchain Data Gathering Solution for Cheese Tracking*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, August 2022.

69. (Bachelor Thesis) Ramon Solo de Zaldivar: *Creation of a Dataset Modeling the System Calls of Spectrum Sensors Affected by Malware*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, April 2022.

70. (Bachelor Thesis) David Stalder: *Machine-learning based Detection of Malicious DNS-over-HTTPS (DoH) Traffic Based on Packet Captures*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, April 2022.

71. (Bachelor Thesis) Kyrill Hux: *Design and Implementation of a Traffic Sinkhole for Cyberattack Analysis*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, March 2022.

72. (Bachelor Thesis) Marion Dübendorfer: *Distributed Analysis of Cyberattacks*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, March 2022.

73. (Master Project) Sandro Padovan, Michael Nadig, Christian Birchler: *DDoSGrid 3.0: Enabling the Real-time Processing and Analysis of Cyber Attacks Traffic*; Universität Zürich, Communication Systems Group, Department of Informatics, Zürich, Switzerland, February 2022.

These publications and theses support the chapters of this thesis, as detailed in Table A.1, where the indices correspond to the list above.

Chapter	*Related Publication*	*Related Thesis*
Introduction	1, 3, 11-14	–
Fundamentals and Definitions	12-14, 39	15, 21, 23, 26, 28, 31
Related Work	2-5, 7, 9, 10-14, 46	20, 21, 23-25, 28, 29, 31-33
Modeling and Prioritizing Threats	2, 4, 5, 12-14	18, 20, 21, 23
Detecting and Mitigating Threats	1, 3, 6-11	19, 22, 24, 25, 28-32
Deployments and Pilots	–	17, 18
Summary and Future Research		

Table A.1: Supporting Theses and Publications

B

Keyword Search in Google Scholar

Listing B.1: Retrieving Documents from Google Scholar

```
async function get (term, year) {
  var res = await fetch(`https://scholar.google.com/scholar?q=%22${term
    }%22&hl=de&as_sdt=0%2C5&as_ylo=${year}&as_yhi=${year}`, {
    "referrer": "https://scholar.google.com/scholar",
    "mode": "cors"
  })
  var text = await res.text()
  try { console.log(year + ',' + text.match(/Ungefähr(.*?)Ergebnisse/)
    [1]) } catch (e) {}
}

async function main () {
  var start = 1960
  var years = new Array(65).fill(0).map((e,i) => start + i)
  console.log(years)
  for (i in years) {
    var year = years[i]
    await get('Threat Modeling', year)
    var timeout = 10 + (Math.random()*20)
    await sleep(timeout)
  }
}

function sleep(s) { return new Promise(res => setTimeout(res, s*1000)) }

main()
```

List of Figures

1.1 Cybersecurity Publications . . . 3
1.2 Academic Publications . . . 4
1.3 Macro-level Risk Management Steps Defined by NIST [185] . . . 5
1.4 Solutions Contributed per Problem Area . . . 10

2.1 Relation of Key Terms . . . 14
2.2 Risk Factors in the Risk Model defined by NIST [138] . . . 15
2.3 Seven Step Risk Management Steps in NIST SP 800-37 [137] . . . 18
2.4 Organizational Risk Management [137] . . . 18
2.5 Cost of Change [17] . . . 20
2.6 Exemplary Threat Model of a Fictitious System . . . 21
2.7 Abstract Threat Modeling Methodology [276] . . . 22
2.8 Levels of Measurement and Related Prioritization Approaches . . . 23
2.9 Tree-based Decomposition of a Threat . . . 25
2.10 Threat Modeling on a Whiteboard [103] . . . 26
2.11 Usage of Control Mechanisms Across Organizations [150] . . . 29
2.12 Incident Response Life Cycle in NIST SP 600-61 [100] . . . 30
2.13 Attack Taxonomy: Categorizing Attacks by Operational Impact [233] . . . 33
2.14 TCP 3-way Handshake [161] . . . 35
2.15 Cyber Kill Chain Model [124] . . . 37
2.16 Relation between Theory, Practice, Problem Space, and Solution Space [72] . . . 38
2.17 Internet-of-Things Three-Layer Architecture [221] . . . 39
2.18 Architecture of the Swiss Covid Certificate System [182] . . . 42
2.19 Technologies Subsumed under the Umbrella Term "Artificial Intelligence" . . . 44
2.20 Generic AI Life Cycle [75] . . . 45
2.21 Binary Classification . . . 46
2.22 Design Science Research [113] . . . 50
2.23 Hypothetical log-normal distribution with $\mu = 0$ and $\sigma = 0.5$ . . . 55

4.1 Threat Modeling Research: Environmental Challenges, Solutions, and Contributions . . . 98
4.2 Overview of *CoReTM* Components and Interfaces [274] . . . 99
4.3 Visual Stencil Library Provided by *CoReTM* [274] . . . 100
4.4 Threat Modeling Methodology Selection Questionnaire [274] . . . 102
4.5 Key Prototype Components At Modeling Stage [274] . . . 104
4.6 Workspace After Conducting the Misuse Case Modeling [274] . . . 106

4.7 Deducing Threats and Controls from OWASP ASVS [274] . . . 108
4.8 Approach and Key Components [277] . . . 110
4.9 Knowledge Base Sources [277] . . . 111
4.10 Reviewing Threats in Context [115] . . . 112
4.11 Reported Insider Threat Model [277] . . . 114
4.12 Procedural Elements (White) and Artifacts (Black) in *QuantTM* [273] . . . 122
4.13 Steps and Artefacts of the BIA Approach [273] . . . 129
4.14 Defining Exemplary Impact Factors [273] . . . 131
4.15 Visualizing Two Hypothetical Persistent Loss Scenarios [273] . . . 133
4.16 Qualitative Analysis [273] . . . 144
4.17 *ThreatFinderAI* Architecture and Prototype [269, 280] . . . 147
4.18 Quantifying the Business Impact on an AI System [269] . . . 152
4.19 Architectural Modeling and Asset Annotation in *ThreatFinderAI* [269, 280] . . . 153
4.20 Demonstration: Scenario Definition and User Interface Overview [269] . . . 154
4.21 Demonstration: Modeling the Architecture [269] . . . 155
4.22 Demonstration: Identifying Threats [269] . . . 156
4.23 Demonstration: Identifying Controls [269] . . . 157
4.24 Demonstration: Identifying Impacts [269] . . . 158

5.1 Solutions' Context for Studies on Detection and Mitigation . . . 170
5.2 Framework Architecture for Lightweight, Host-based MTD [267] . . . 171
5.3 File Encryption Defense [267] . . . 173
5.4 Confusion Matrix of the RF Model [267] . . . 177
5.5 Defense Example [267] . . . 179
5.6 *RansomAI* Architecture [48] . . . 179
5.7 Methodology to Develop Rule-based MTD [270] . . . 184
5.8 Rule-based MTD Framework Architecture [270] . . . 184
5.9 Three-stage Dynamic Experiments [270] . . . 186
5.10 RansomwarePoC and MTD1: CPU Idle Time [270] . . . 187
5.11 Framework Efficiency [270] . . . 191
5.12 Container-based Dynamic Analysis Architecture [81] . . . 194
5.13 Realtime Visualizations [271] . . . 195
5.14 Live Interaction being Relayed to Both Sandboxes [271] . . . 196
5.15 Network Usage Across Protocols [81] . . . 199
5.16 *MTFS* Architecture and Prototypical Design [268] . . . 205
5.17 Ransomware Testbed Architecture [268] . . . 206
5.18 Baseline Defense Through Process Termination [272] . . . 210
5.19 Defense Through Obfuscation [272] . . . 210
5.20 Defense Through Delaying and Obfuscation [272] . . . 211
5.21 Defense Involving Tracking of Processes [272] . . . 211
5.22 Data Loss in Testbed [268] . . . 213
5.23 Performance Analysis During Defense [272] . . . 217
5.24 Overview . . . 220

6.1 Security Practises in OWASP SAMM [199] . . . 223

6.2 Architecture of the *CAVE* Platform [291] 224
6.3 *CoReTM* 2.0 Architecture [262] 226
6.4 Threat Tables Created by *CoReTM* 2.0 227

7.1 Cybercrime GDP [28] 229

List of Listings

2.1 Text-based System Description for Threat Modeling [230] 26
2.2 Approximating π by Randomly Generating Points in a Square 56
4.1 Internal Representation of a Threat Elements [274] 101
4.2 Modeling the distribution of P_i [273] 127
B.1 Retrieving Documents from Google Scholar 269

List of Tables

2.1 Non-exhaustive Set of Malware-based Attack Vectors . . . 34
2.2 Illustrative Overview of Denial-of-Service Attack Vectors . . . 36
2.3 Threats Related to ML [152, 194, 198] . . . 47
2.4 Overview of BPMN 1.2 Elements [53] . . . 52
2.5 Research Methodologies [113, 157, 235] . . . 61

3.1 Overview of Surveyed Threat Modeling Tools [274] . . . 71
3.2 Comparison of Toolkit Support for Collaboration [275] . . . 77
3.3 Studies Applying Threat Modeling to AI Systems [280] . . . 79
3.4 Studies Investigating Insider Threat Modeling or BPMN [277] . . . 82
3.5 Threat Assessment Tools and Methods [273] . . . 83
3.6 Dynamic Malware Analysis Solutions [81] . . . 87
3.7 Constrained MTD Techniques Targeting Malware [267] . . . 89
3.8 Comparing MTD Policy Selection Methods [270] . . . 90
3.9 Categorization of Surveyed Related Work [272] . . . 93
3.10 OS-level Moving Target Defense [268] . . . 94

4.1 Insider Threat Report [277] . . . 115
4.2 *QuantTM* Terminology [273] . . . 123
4.3 Case Context [273] . . . 134
4.4 Information System Threat Model [273] . . . 135
4.5 Threat Model Interpretation [273] . . . 136
4.6 Modeled Business Impacts [273] . . . 137
4.7 Quantified Threat Model [273] . . . 138
4.8 Focus Group Participants [273] . . . 141
4.9 Usability Scores (SUS) [273] . . . 143
4.10 DREAD-based Analysis [273] . . . 145
4.11 Generic 5-Step Threat Modeling Process [160, 179] . . . 148
4.12 Participant Overview [280] . . . 160
4.13 System Usability Score [280] . . . 161
4.14 Identified Countermeasures [280] . . . 164
4.15 Estimating Impact Factors [280] . . . 165
4.16 Simulated Losses [280] . . . 165

5.1 Monitoring Efficiency [267] . . . 176
5.2 Model Performance [267] . . . 176

5.3 Enforcement Efficiency and Effectiveness [267] . . . 178
5.4 Ransomware Configurations [266] . . . 180
5.5 Accuracy of Offensive Reactivity Agent [266] . . . 181
5.6 Anomaly Detector Performance [266] . . . 182
5.7 Initial Feature Hypothesis [270] . . . 186
5.8 Defense Policies [270] . . . 187
5.9 Detection and Mitigation Effectiveness [270] . . . 189
5.10 Detection Performance [270] . . . 190
5.11 Defending Multiple Malware Infections [270] . . . 191
5.12 Resource Consumption [81] . . . 197
5.13 Dataset Similarity [81] . . . 198
5.14 Model Performance [21] . . . 201
5.15 Isolated Runtime Detection Strategies for Evasion [222] . . . 202
5.16 Test Environments [222] . . . 203
5.17 Defense Effectiveness in Sensor Node [268] . . . 212
5.18 RF Classifier Performance: Detecting Unseen Samples [272] . . . 214
5.19 Accuracy of LF and RF Classifiers [272] . . . 215
5.20 Testing Detection in Sensor [272] . . . 215
5.21 Data Loss (in KB) Across Defense Techniques [272] . . . 216
5.22 Usability Impact and Resource Consumption During Normal Operation [272] . . . 218

A.1 Supporting Theses and Publications . . . 268

Curriculum Vitae

Personal Details

Name	Jan von der Assen
Date of Birth	December 7, 1993
Place of Birth	Winterthur, Zürich, Switzerland

Education

July 2021 – July 2025	Doctoral program at the University of Zurich, UZH Department of Informatics, IfI, Communication Systems Group, CSG *University of Zurich, UZH*
Sep 2018 – Jan 2021	Master of Science in Informatics (M. Sc.) Major: Software Systems, Minor: Informatics *University of Zurich, UZH*
Sep 2014 – Jan 2018	Bachelor of Science in Informatics (B.Sc.) Major: Information Systems *University of Zurich, UZH*

Professional Experience

2021 – 2025	Junior Researcher, *Communication Systems Group (CSG), University of Zurich, UZH*
2015 – 2021	Software Engineer, *dizmo AG*
2013 – 2014	Technical Consultant, *rbc Solutions AG*