University of Zurich[UZH]

Design and Implementation of a Threat Modeling Approach for AI-based Systems

Jamo Sharif
Zurich, Switzerland
Student ID: 11-453-388

Supervisor: Jan von der Assen, Chao Feng
Date of Submission: December 26, 2023

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zurich, Switzerland

ifi

BACHELOR THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Bachelor Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zurich, Switzerland
URL: http://www.csg.uzh.ch/

Zusammenfassung

Diese Arbeit konzentriert sich auf die dringende Notwendigkeit, effektive Bedrohungsmodellierungsstrategien im Bereich der Künstlichen Intelligenz (KI) und des Maschinellen Lernens (ML) zu entwickeln. Der Bedarf hierfür wird besonders deutlich, wenn die Grenzen traditioneller Modellierungsansätze im Kontext von KI- und ML-Systeme betrachtet werden. Diese Systeme sind omnipräsent und haben signifikante Auswirkungen in verschiedenen Lebensbereichen, von der Problemlösung bis hin zur potenziellen Kontrolle verschiedenster Aspekte. Die einzigartigen Schwachstellen von KI- und ML-Systemen, insbesondere ihre Anfälligkeit für Angriffe durch feindseliges ML sowie herkömmliche Cybersicherheitsherausforderungen, unterstreichen die Notwendigkeit eines speziell angepassten Bedrohungsmodellierungsansatzes. Diese Arbeit zielt darauf ab, diese Lücke zu schliessen, indem sie einen solchen spezialisierten Ansatz entwickelt und implementiert.

Diese Arbeit bietet eine eingehende Untersuchung bestehender Literatur, Methoden und Cyberangriffe in der ML-Pipeline, um eine wesentliche Lücke in den aktuellen Bedrohungsmodellierungsansätzen für KI-Systeme aufzudecken. Diese Lücke bezieht sich auf das Fehlen von systematischen Prozessen und etablierten Techniken, die speziell darauf ausgerichtet sind, die Komplexität von KI-Systemen effektiv zu bewältigen. Ein grundlegendes Referenzmodell für den Bereich der KI wird als entscheidender Schritt zur Entwicklung eines neuen Bedrohungsmodellierungsansatzes identifiziert. Zusätzlich erläutert diese Arbeit den herausfordernden Prozess der Erarbeitung dieses Bedrohungsmodellierungsansatzes für KI-basierte Systeme. Dieser Prozess basiert auf dem grundlegenden KI-Referenzmodell, das von der Europäischen Union Agentur für Cybersicherheit (ENISA) bereitgestellt wird, und modifiziert einen allgemeinen 5-Schritte-Bedrohungsmodellierungsprozess, um den spezifischen Anforderungen der KI gerecht zu werden.

Ein Prototyp namens *ThreatFinder* wird entwickelt, um die praktische Anwendung des vorgeschlagenen Bedrohungsmodellierungsansatzes zu demonstrieren. Die Funktionalität und Struktur von *ThreatFinder* werden erläutert, wobei seine zentrale Rolle innerhalb der Sicherheitslandschaft im Bereich KI hervorgehoben wird. Die Wirksamkeit von *ThreatFinder* wird durch ein gezieltes Experiment mit Teilnehmern aus verschiedenen Bildungshintergründen bewertet, um Einblicke in seine Effektivität und den potenziellen Mehrwert, den es für das Themengebiet bringt, zu geben. Folglich leistet diese Arbeit einen bedeutenden Beitrag zum Bereich der KI-Sicherheit, indem sie die Sicherheit und Zuverlässigkeit von KI-Systemen gewährleistet, eine entscheidende Weiterentwicklung angesichts der tiefgreifenden Bedeutung der KI.

Abstract

This thesis focuses on the urgent need to develop an effective threat modeling approach in the field of Artificial Intelligence (AI) and Machine Learning (ML). The necessity for this becomes particularly clear when considering the limitations of traditional threat modeling approaches in the context of AI and ML systems. These systems are ubiquitous and have significant impacts on various aspects of life, from problem-solving to the potential control of diverse aspects. The unique vulnerabilities of AI and ML systems, particularly against Adversarial Machine Learning (AML) threats, alongside traditional cybersecurity, underscore the need for a specially tailored threat modeling approach. This thesis aims to close this gap by developing and implementing such a specialized approach.

This thesis provides a comprehensive examination of existing literature, methodologies, and cyber attacks in the ML pipeline, revealing a significant gap in current threat modeling approaches for AI systems. This gap pertains to the lack of systematic processes or established techniques for effectively addressing the complexities of AI systems. A foundational reference model within the AI domain is identified as a crucial step for developing a new threat modeling approach. Additionally, this thesis outlines the challenging process of designing the threat modeling approach for AI-based systems using the foundational AI reference model provided by the European Union Agency for Cybersecurity (ENISA) and modifying a general 5-step threat modeling process.

A prototype tool named *ThreatFinder* is developed to demonstrate the practical application of the proposed threat modeling approach. The functionality and structure of *ThreatFinder* are explained, highlighting its central role within the AI security landscape. The effectiveness of *ThreatFinder* is assessed through a targeted experiment involving participants from various educational backgrounds, providing insights into its effectiveness and the potential added value it brings to the field. Consequently, this work makes a significant contribution to the field of AI security by ensuring the safety and reliability of AI systems, a crucial advancement given the profound relevance of AI.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my supervisor, Jan von der Assen, for the continuous support, expert feedback, and interesting discussions throughout this thesis. Moreover, I would like to thank my co-supervisor, Chao Feng, for his inputs.

I would also like to thank Prof. Dr. Burkhard Stiller for the possibility to complete my Bachelor thesis at the Communication Systems Group (CSG) of the University of Zurich. Finally, I thank my friends and family for proofreading this thesis, and for their constant support.

Contents

Chapter 1

Introduction

In recent years, the progress in Artificial Intelligence (AI) techniques has been impressive, making AI nearly omnipresent. A notable illustration is the Guardian's report on AlphaZero AI [1], which won against a champion chess program after just four hours of self-training. This exemplifies the current reality where the impact of AI extends beyond imagination, from solving complex problems to potentially influencing and controlling various aspects of our lives [2].

The application of AI, particularly through various methods such as Machine Learning (ML), has demonstrated its efficacy in addressing challenges within the cybersecurity field. These advanced technologies are now tackling problems once effectively considered unsolvable. For instance, malware detection has traditionally relied on static analysis, examining the contents of malware artefacts. However, this method is susceptible to evasion through simple obfuscation techniques, such as changing the source code. In this context, leveraging ML, behavior-based approaches were able to identify generic patterns of malicious software behavior, even in cases where the source code is altered [3]. While a large share of companies have transitioned from research to adopting AI systems in their Information Security Programs, notable challenges persist [4]. According to a recent study [5], an inadequate threat model is one of the most prevalent pitfalls of applying AI to the security field. For instance, while an AI system can identify a server security breach, it might not ensure the safety of its own activities, such as collecting unfalsified data, conducting model evaluation, and generating reports.

1.1 Motivation

As AI-based systems become increasingly integrated into our daily lives, ensuring their security is paramount. Especially, ML plays a significant role in a wide range of application domains [6]. These systems, however, while vulnerable to the same threats that plague traditional software systems without AI components, exhibit additional vulnerabilities to various types of Adversarial Machine Learning (AML) threats (see Sections 3.1 and 3.2). These AML threats also pose potential severe consequences [7], [8]. Nevertheless, security concerns within AML research primarily focus on ML models. More precisely, the research in this field examines the complex methods used to compromise ML-based systems during the crucial phases of the learning process, specifically during training and inference [9], [10]. Hence, there is a notable gap in exploring a holistic paradigm that concurrently addresses both AML threats and traditional security concerns in ML systems. This gap remains widely unexplored [10].

Threat modeling (see Section 2.3) has emerged as a crucial practice in safeguarding traditional software systems. Yet, the reason why similar approaches have not been broadly applied to AI-based systems remains unclear. While threat modeling could offer a valuable overview of potential threats in AI systems, there is currently no systematic process or established technique for identifying threats and vulnerabilities across the entire AI pipeline [9].

As previously noted, threat modeling has primarily been employed in the context of traditional software systems. Nonetheless, it is crucial to acknowledge that AI systems, with their unique structure, encompass specific assets beyond the conventional elements of information and communication technology, such as data, software, hardware, and communication networks. The distinctive set of assets in AI, including models, processors, and artefacts, can be susceptible to compromise or damage, whether intentional or unintentional [11]. Consequently, the significant challenge lies in adapting threat modeling methods designed for traditional software systems to address the complexities of AI systems effectively.

This thesis wholeheartedly embraces the challenge. Its objective is to develop a threat modeling approach for AI-based systems, enabling system implementors to characterize the hostile execution environment of the AI system. The focus is on identifying and highlighting potential threats to the system. This endeavour would mark a significant milestone in AI security, ensuring the safety and reliability of AI systems, a critical advancement given the profound relevance of AI.

1.2 Description of Work

As mentioned, this thesis aims to design and develop a threat modeling approach tailored to AI-based systems, mainly focusing on AI and ML. The absence of an established threat modeling approach for AI systems requires a comprehensive review of existing literature and methodologies, followed designing and implementing a tailored threat modeling approach.

The thesis begins by introducing key terms like threat modeling and AI-based systems. A thorough literature research ensues, encompassing existing cyberattacks in the ML pipeline, methodologies, and tools for threat modeling. The exploration delves into existing threat modeling approaches, emphasizing challenges and prompting the identification of a novel approach tailored to AI systems. This comprehensive research not only establishes a robust knowledge foundation but also highlights a critical gap within this domain that warrants attention and resolution.

In the next stage, the focus shifts to identifying a foundational reference model within the AI domain, a crucial step for shaping the new threat modeling approach. This chapter further outlines the challenging process of designing the threat modeling approach for AI-based systems, using the foundational AI reference model provided by ENISA and modifying a general 5-step threat modeling process. Subsequently, the prototype is developed, highlighting the pivotal role of the diagrams.net tool. Additionally, this part explains the overall functionality and structure of the newly crafted *ThreatFinder* tool.

The conclusive phase involves the evaluation of the prototype through a targeted experiment. Seven participants with different educational backgrounds engage with the newly developed *ThreatFinder* tool, providing insights into its effectiveness and the potential added value it brings to the domain.

1.3 Thesis Outline

This report is structured as follows: Chapter 2 establishes the theoretical foundation, introducing key concepts crucial for comprehending the main topic, the ”threat modeling approach for AI-based systems”. In Chapter 3, comprehensive research explores cyberattacks in the ML pipeline, failure mode evaluation, and existing threat modeling approaches for AI systems. Additionally, the selection of the most suitable threat modeling tool for AI systems is considered, laying the groundwork for the subsequent chapter. In Chapter 4, an appropriate AI architecture is proposed, and, guided by a general 5-step threat modeling approach, the prototype for a threat modeling approach tailored to AI-based systems is conceptualized. On this basis, the development of the approach takes place in Chapter 5. Finally, the evaluation of the implemented prototype is the focus of Chapter 6, followed by a conclusion in Chapter 7.

Chapter 2

Background

In order to understand the core of this bachelor thesis, it is essential to introduce basic terms and concepts first. This background section systematically clarifies definitions and nuances related to vulnerabilities, threats, and some different methodologies used for threat modeling. In addition, it explains the methodologies applied in threat modeling and the relevance of AI-based systems, particularly ML.

2.1 Vulnerability

To understand the concept of vulnerability, it is important to define a weakness in this context. A weakness is an underlying defect that alters behavior or functionality, leading to incorrect behavior or allowing access to data that is either unauthorized or inaccurately granted [12]. Consequently, vulnerabilities can be comprehended as weaknesses in a system's architecture or its design, which permit an invader to execute commands, gain improper access to data, or carry out denial-of-service attacks [13], [14]. Therefore, exploitable weaknesses constitute a vulnerability [12].

Vulnerabilities pave the way for an antagonist with malicious intent to inflict a certain level of damage to a system [12]. In particular, vulnerabilities can be located in the system's hardware or software, in the policies and procedures that govern the system, or even amongst the users of the system themselves [15].

2.2 Threats

A threat is an action or potential action that exploits vulnerabilities in a system, which can have a negative impact on it [16]. The primary origins of these threats can be traced back to either humans or nature [17], [18]. It is important to highlight that any entity engaging with the system capable of causing threats can be termed a threat actor.

Natural threats, such as floods, hurricanes, earthquakes, and fire can seriously harm computer systems. Few mitigating measures can be implemented against these natural disasters. However, the natural disasters themselves are ultimately unavoidable. The most effective strategies to secure systems against natural threats are disaster recovery plans like data backup and contingency plans [19].

Human threats derive from human actions, such as malicious threats that can be internal. Internal threats often originate from individuals with authorized access to the system. On the other hand, external threats are individuals or organizations working outside the network intending to inflict damage or cause disruption to the system [19]–[21]. Hence, a threat, whether internal or external, regardless of its intent, is anything that can exploit a vulnerability to impact the system's assets [12] negatively.

Threats originating from human actors are categorized into the following [19]: *(i)* Unstructured threats, which largely come from inexperienced individuals, utilizing easily available hacking tools [19] or *(ii)* structured threats, where the actors know the system vulnerabilities and have the ability to understand, develop and exploit various codes and scripts [19]. The Advanced Persistent Threat (APT) is a prominent example of a structured threat [22]. APT represents a complex network attack targeted at high-value information in business and government organizations, such as financial industries, manufacturing, and national defense, with the aim to steal data [23]. Additionally, for a comprehensive overview, Figure 1 presents a visually informative illustration that effectively depicts the relationship between the terms threat and vulnerability [24].

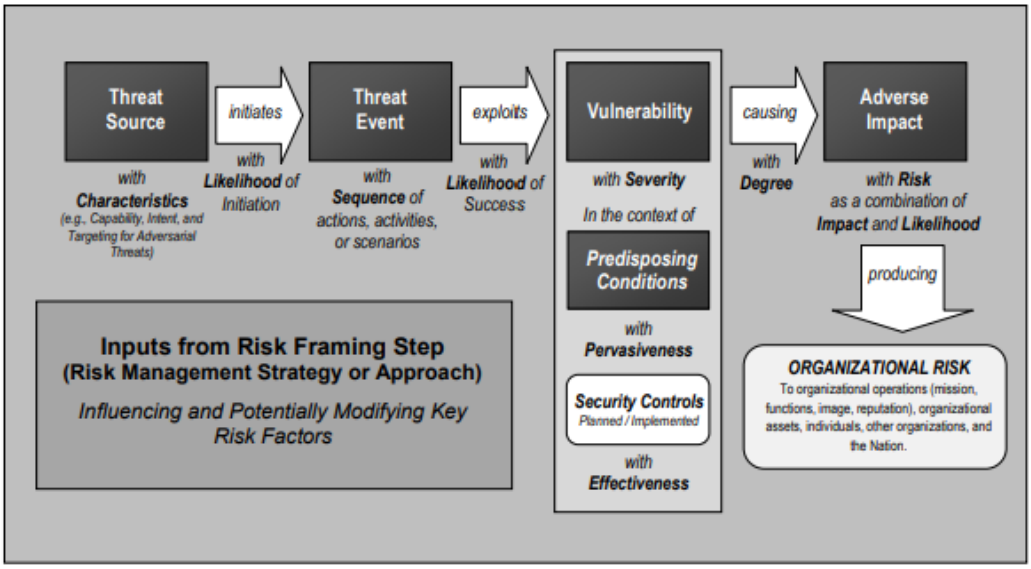

Figure 1: Generic Risk Model With Key Risk Factors [24]

2.3 Threat Modeling

Threat modeling is an analytical and cyclic procedure to look for system vulnerabilities that come from suboptimal design choices. The activity aims to detect these weaknesses before they become part of the system through implementation or deployment, thereby enabling early corrective actions [12]. Threat modeling is a distinct system design activity that must be distinguished from other methodologies, such as risk assessment. While risk assessment encompasses a broader scope, periodically addressing overall risks, threat modeling is more specific. It concentrates on identifying and mitigating vulnerabilities through targeted scenarios [25]. Furthermore, threat modeling is a well-known process for meticulously designing systems, networks, and businesses [26]. Once potential threats are identified, suitable countermeasures can be considered, communicated, and implemented [27].

The threat modeling process consists of five steps, each identifying the following elements [28]: *(i)* Assets are valuable components and crucial for operational delivery. Assets can include hardware, software, services, and data. Moreover, due to their value, they might be targets for adversaries. 2 Attack surface are the various components of the system vulnerable to unauthorized access by adversaries. *(iii)* Adversary model specifies the adversary's characteristics, such as their identity, motives, and capabilities. *(iv)* Vulnerabilities are the weaknesses within the asset that the adversary can exploit for security compromise (see Section 2.1). Threats, on the other hand, are events in which the adversary takes advantage of the asset's vulnerabilities to initiate an attack (see Section 2.2). *(v)* Mitigation measures are the security solutions designed to prevent, detect, or reduce the impact of threats [28]–[30]. However, it is important to note that these steps serve as a valuable meta-model for considering threat modeling methods. Notably, not all steps within this framework are obligatory, highlighting flexibility. These steps are adaptable and open to modification. For example, step *(iii)* is occasionally omitted.

2.4 CIAA Properties in Threat Modeling

The security of computer-related systems requires four aspects: *(i)* Confidentiality, *(ii)* Integrity, *(iii)* Availability, and *(iv)* Authentication, often referred to as the CIAA process for threat modeling [31]. The corresponding definitions can be found in Table 1.

Table 1: CIAA Triad Overview [32]

Property	**Definition**
Confidentiality	Property that information is not made available or disclosed to unauthorized individuals, entities, or processes.
Integrity	Property of accuracy and completeness.
Availability	Property of being accessible and usable upon demand by an authorized entity.
Authentication	Verifying the identity of a user or system entity.

The associated attacks are explained in more detail along with the various properties: *(i)* Confidentiality attacks attempt to access information without proper authorization. When attackers attain administrator-level access to a system, they often have relatively unrestricted access to the resources. However, it is important to note that attackers can also read information without illegitimate privilege escalation,for example, as insider attackers [31]. *(ii)* Integrity attacks, on the other hand, seek to modify information within a system without proper authorization. Modification can encompass a range of actions such as creating, changing, appending, writing, and deleting both data and metadata [31].

Furthermore, there exist *(iii)* Availability attacks, which aim to make data or services unavailable for some time. These services, including both data and metadata, must be available on-demand to legitimate parties when requested. Denial-of-service (DoS) attacks are designed to make data and services unavailable by exhausting resources through legitimate mechanisms, which makes this attack the hardest to prevent [31]. As a final aspect, *(iv)* Authentication attacks occur when an attacker masquerades as a legitimate user identity, often by using stolen passwords or credentials, or an attack device masquerades as a legitimate device. For instance, a perpetrator can launch insider attacks to access data/metadata (confidentiality), modify data/metadata (integrity), or block others from accessing data/metadata (availability), all based on the legitimate user identity authorization capabilities they have taken over. Another way for authentication attacks is man-in-the-middle attacks [31]. The man-in-the-middle

attack is one of the most well-known attacks in computer security, ranking among the top concerns for security professionals. Here, the attacker aims to compromise the actual data that flows between endpoints and the confidentiality and integrity of the data itself [33]. Hence, the man-in-the-middle acts as an invisible intermediary between the communicating parties.

2.5 Threat Modeling Methodologies

Threat modeling methods are used to form an abstraction of the system, create potential attacker profiles with their corresponding goals and strategies, and generate a catalog of possible threats that might emerge [34]. Therefore, threat modeling methods enable the identification of critical areas of the design which need to be protected [27]. Hence, establishing security in software systems considerably relies on threat modeling methods [27].

Numerous threat modeling methods have been developed so far and are now used to design secure web applications [27], [34]. Hereby, some threat modeling methods prioritize detailed system abstraction and granularity, while others are more people-centric, while others emphasize risk or privacy concerns [34]. In the following sections, various threat modeling methodologies will be enumerated and explained.

2.5.1 STRIDE and DREAD

The STRIDE method was invented by Loren Kohnfelder and Praerit Garg in 1999 [29]. STRIDE is an acronym for Spoofing, Tampering, Repudiation, Information Disclosure, DoS, and Elevation of Privilege (see Table 2) [12], [29], [35]. In the following paragraph, the definitions of these terms are explained, and some examples are provided as well.

(S) Spoofing is the act of pretending to be someone or something other than oneself. In contrast, (T) tampering means to modify something on a network, on disk, in memory, or elsewhere. Changing either the data a program is using or the running program would be a tampering threat. Meanwhile, (R) repudiation refers to the claim that you did not undertake a certain action, or were not responsible for it. Repudiation can be truthful or deceitful, and the primary challenge for system designers here is determining the proof they possess. Repudiation might manifest claims like, ”I never pressed that red button” or ”I did not place an order for that car”. It is important to highlight that repudiation is the somewhat odd threat among other threats. At the same time, many threats focus on technical aspects, repudiation bridges into the business layer. Another significant concern is (I) information disclosure, which is the act of providing information to those not authorized to access it. Here, the most obvious example is allowing access to files, e-mail, or databases. Furthermore, there are (D) DoS threats, focusing on consuming resources needed for service delivery. For example, a program that can be tricked into using up all its memory. Finally, (E) elevation of privilege allows someone to do something even though they are not authorized. As an illustration, allowing a regular user to execute code with admin privileges [12], [29], [34], [35].

STRIDE analyzes vulnerabilities within each system component that an attacker could manipulate to breach the whole system [35]. Hence, STRIDE examines the detailed system design [34]. Due to the absence of a standard methodology, applying STRIDE threat modeling against a system is carried out in different steps [35].

As a first step, the in-place system is modeled, which means decomposing the system into its logical or structural components [34], [35]. It is common to use data flow diagrams (DFDs),

which are still a powerful tool for system analysis and design process [36]. A DFD is a fundamental artefact in the structured approach, crucial for every system. It offers a hierarchical structure, providing different abstraction levels essential for system design. The DFD is a graphical representation of the data flow within an information system. Moreover, it illustrates the movement from one process to another and provides essential information for other system artefacts to represent dynamic aspects [37]–[39]. Hence, by creating DFDs, system entities, events, and boundaries of the system can be identified [40]. The accuracy of the DFDs will determine how successful the application of STRIDE will be [41]. Nevertheless, using DFDs as the main input to threat modeling can be restrictive, as they do not offer mechanisms for representing security-related architectural decisions [42].

The next step involves identifying STRIDE threats within the DFD of every system component [35]. As it can be seen in Table 2, STRIDE uses a general set of known threats resulting from its name [34]. This acronym can serve as a mnemonic for discovering threats while exploring the systems model, which was established in step one [29]. Some resources provide checklists and tables to facilitate this step, describing threats, property violations, common victims, and attackers' actions [29], [34], [43]. At this point, it is important to note that depending on the functionality of each system component, certain STRIDE threats might not be applicable to it [35].

After identifying threats for each system component, it is essential to examine the vulnerabilities that cause them [35]. The final step is formulating effective mitigation strategies according to the discovered vulnerabilities [35]. Once mitigation strategies to the corresponding threats have been developed, it is crucial to document and prioritize these findings [41], [42].

Currently, STRIDE is the most well-developed threat modeling method [34]. So far, the application of STRIDE has been successful to cyber-only and cyber-physical systems [29], [35], [41], [43]. Moreover, the STRIDE method is straightforward in traditional software systems but can be time-consuming [29], [41]. A key challenge with STRIDE is that the number of threats can grow rapidly as a system's complexity increases [34]. Furthermore, it has been demonstrated in a descriptive study that the STRIDE method yields a relatively low rate of false positives but a relatively high rate of false negatives (e.g., overlooked threats) [43].

There are two different variants for performing STRIDE, namely STRIDE per element and STRIDE per interaction [12], [29]. Due to its focus on analyzing the behavior and operations of each system component, STRIDE per element is considered more complex [35]. However, it might lead to ignoring the holistic approach to the system during the analysis – even if the system is fully represented – since certain threats are not evident from the DFD [12], [35]. In contrast, the STRIDE per interaction approach enumerates threats by considering tuples, including origin, destination, and interaction, and enumerating threats against them [29]. In comparison, STRIDE per interaction is easier to execute, and its protection strategies are generally sufficient for system protection – in particular since cyber attacks commonly involve harmful interactions between system components [35].

Microsoft developed another modified STRIDE method called DREAD [35]. The DREAD model serves as a threat evaluation model and is utilized to assign threat severity and priority level of detected threats [27]. DREAD is also a mnemonic for Damage potential, Reproducibility, Exploitability, Affected users, and Discoverability [34]. It allots either of three values (0,5,10) to the first four categories, and one of four values (0,5,9,10) to the last category, enabling the computation of an average value that represents the risk of the entire system [34], [44].

Table 2: The STRIDE Threats [29]

	Threat	Property violated	Typical Victims
S	Spoofing	Authentication	Processes, external entities, people
T	Tampering	Integrity	Processes, data stores, data flows
R	Repudiation	Non-Repudiation	Processes
I	Information Disclosure	Confidentiality	Processes, data stores, data flows
D	DoS	Availability	Processes, data stores, data flows
E	Elevation of Privilege	Authorization	Processes

2.5.2 PASTA

The Process for Attack Simulation and Threat Analysis (PASTA), developed in 2012, represents a threat modeling methodology that is risk-centric [34]. Thus, it evaluates the potential risks that influence a business or a system, starting with contextual references, emphasizing the inherent importance of an application, its components, the underlying infrastructure, and data on business operations [12]. PASTA contains seven different stages, each involving several activities depicted in Figure 2 [12], [45], [46].

Figure 2: Risk Centric Threat Modeling Case Studies [47]

The goal of PASTA is to harmonize business objectives with technical requirements [48]. To achieve this goal, it uses various design and elicitation tools in different stages [34]. For instance, to identify the technical scope, high-level architectural diagrams are used during stage two, or

DFDs are implemented in stage three [34]. In stage six, attack trees are built, and use and abuse cases are developed for analysis and attack modeling [12], [49].

This approach raises the threat modeling procedure to a strategic level, requiring the participation of decision-makers and security inputs from operations, governance, architecture, and development [50]. Broadly recognized as a risk-centric framework, Pasta maintains an attacker-centric perspective – the attacker's point of view is adopted [12], [34]. Ultimately, the process yields an asset-centric output manifested as enumeration and scoring of threats [49], [50].

2.5.3 Attack Trees

Utilizing attack trees to model threats dates back to one of the earliest and most widely chosen techniques, applicable to cyber-only, cyber-physical, and physical systems [44]. The initial idea behind this method – developed by Bruce Schneider in 1999 – was to be applicable as its own. However, it has been combined with various other methods and frameworks since [34], [44].

Attack Trees offer a structured, methodical way of describing the security of systems by considering various potential attacks [51]. The attack against a system is represented as a tree structure, where the root node symbolizes the goal of the attack, while the leaf nodes illustrate the different ways for achieving that goal [27]. Every node represents a subgoal, and the children of that node are ways to accomplish that subgoal [51]. Hence, there are multiple ways to reach the goal. To integrate these various options within the tree, the logical operators AND and OR are used [34]. OR nodes symbolize alternative options and AND nodes indicate different steps necessary to achieve the same goal [27].

Once the tree is constructed, usually through a few iterations of decomposing the goal, distinct values can be assigned to the leaf nodes. These values are subsequently employed to evaluate the security of the goal [34], [51]. The values are assigned manually and totally dependent on the security expert and system engineer [27]. Other attributes, such as the time needed to complete a step, operational expense, and the level of expertise required to initiate an attack can also be incorporated to attack trees [27]. An Attack Tree aids in design and requirement decisions [51]. If the costs of an attack exceed the benefit of the perpetrator, it is unlikely that the attack will occur; however, easy attacks that may yield a benefit need defenses [51].

2.6 AI-based systems

Over the past ten years, increased computer processing power, expansive data sets, and refined algorithms have enabled significant progress in AI [52]. AI is a leading technology in the era of the Fourth Industrial Revolution (Industry 4.0 or 4IR), possessing the capacity to incorporate human-like behavior and intelligence into machines or systems. Consequently, AI-based modeling has become essential in constructing automated, intelligent, and smart systems in order to meet the contemporary demands of today's world [53].

The progress of AI has led to an emerging trend of Deep Learning (DL). DL is a subset of ML, which itself is a subset of AI [53], [54]. The relationship among AI, ML, and DL is illustrated in Figure 3. The feasibility of DL has enabled ML to become an integral part of many widely used software services and applications [54]. Such advancements in AI technology have resulted in many essential applications, like image and speech recognition, as well as autonomous vehicle navigation, to near-human levels of performance [52].

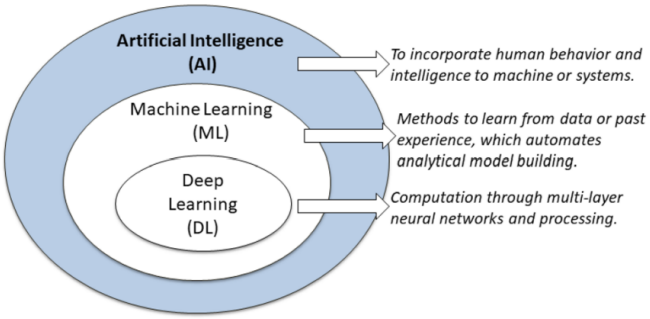

Figure 3: Illustration AI Subfields [53].

The recent developments of AI have profoundly impacted the software industry with the increase of AI-based systems now incorporating AI functionalities, driven by the progress in ML and DL [55], [56]. Software systems with functionalities enabled by at least one AI component are considered AI-based. These systems learn while analyzing their environment and make decisions to showcase intelligent behavior [57]. As stated by the expert group on AI of the European Commission, AI-based systems can be exclusively software-driven, operating in digital space like voice assistants, search engines, image analysis software, and speech and face recognition systems. Alternatively, AI can be integrated into hardware devices, evident in advanced robots, autonomous cars, drones, or Internet of Things applications [58]. Furthermore, developing, managing, and maintaining AI-based systems differ from developing and maintaining traditional software systems. The system's rules and behavior in AI-based systems are deduced from training data instead of being explicitly coded [59]. The development and maintenance of AI-based systems demand interdisciplinary cooperation of data scientists and software engineers [55]. Different quality attributes in these systems must be considered for design and analysis [60]. The evolution of AI-based systems demands attention to large and dynamic datasets, resilient and adaptable infrastructure, as well as ethics and equity requirements engineering [61]. Not considering these differences could lead to deficient AI-based systems with technical debt [62].

2.7 ML: An Overview

ML is recognized as the most favorable AI technology, which is commonly the study of computer algorithms that enable automated analytical model building [63]. ML models generally consist of a set of rules, processes, or advanced "transfer functions" that aid in identifying interesting data patterns or predicting behaviors [64].

ML, also referred to as predictive analytics, utilizes data to predict certain unknowns in the future and addresses many real-world business challenges, such as business risk prediction [53]. Figure 4 illustrates a general framework of an ML-based predictive model, where the model undergoes training using historical data in the initial phase, and the outcome is generated for new test data in the second phase [53].

ML modeling has been applied in almost all aspects of our lives, from healthcare, cybersecurity, and business to education, virtual assistants, recommendation systems, and smart cities, among others [53]. For instance, in [65], an ML strategy for getting COVID-19 assistance to the most vulnerable is provided. In the cybersecurity domain, namely in [66] and [67], many cyber anomalies and attacks detectable via ML techniques are emphasized. As a concluding example,

in [68], an ML-based approach is described to build an optimal parking pricing system for smart city environments.

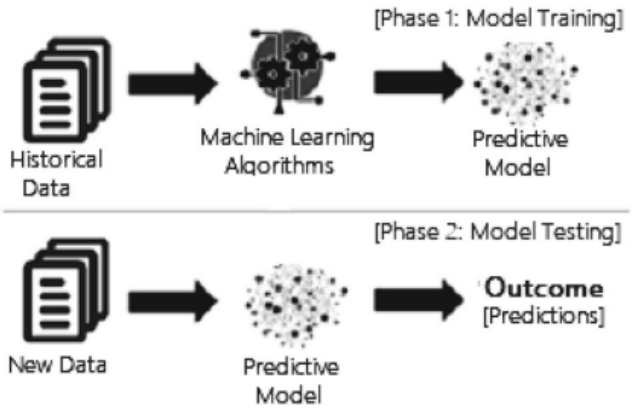

Figure 4: ML Based Predictive Model [53]

Chapter 3

Related Work

As AI-based systems become increasingly integrated into our daily lives, ensuring their security is of utmost importance. As already known, threat modeling (see Section 2.3) has emerged as a crucial practice in safeguarding traditional software systems. The question now is to what extent threat modeling has been applied to AI-based systems and what the current state of research is in this area. Therefore, this chapter explores the existing knowledge and research in the field of threat modeling for AI-based systems. This chapter gives a comprehensive overview of the current landscape by delving into related topics such as AML, failure modes in ML, categories of attacks, frameworks, approaches, and practical tools. It provides readers with a foundational understanding of the challenges and solutions in securing AI-based systems, laying the groundwork for the subsequent exploration of a novel threat modeling approach in this thesis.

3.1 AML

Supervised ML models can be compromised by attackers who manipulate or craft training samples. Such adversarial exploitation has been well documented across a range of applications, including antivirus engines, autonomous bots, visual recognition, and social networks [69]–[72]. These attacks have directed research into ML security, leading to the new research field of AML, a discipline that bridges the gap between ML and computer security [73]–[75]. A comprehensive review of the progress in active research in this area over the past decade is illustrated in [76]. Here, the authors offer a historical overview of ML security through a technical lens. In essence, current research on ML security focuses on *(i)* pinpointing potential weaknesses of ML-based systems, *(ii)* formulating related adversarial strategies, evaluating their impact on the target system, and *(iii)* suggesting protective measures against the observed attacks [9], [76].

The issue of adversarial attacks has recently attracted considerable interest, leading to the release of several studies that introduce innovative attack and defense strategies for specific ML algorithms [77]–[80]. Research in ML security primarily focuses on identifying data-, model-, and system-oriented attacks and defenses [81]–[83]. Thereby, researchers often focus on specific security issues, disregarding an in-depth investigation of secure development practices that cover both ML-specific and traditional system threats [84], [85].

Furthermore, the US National Institute of Standards and Technology (NIST) has recently published a taxonomy for AML, enriched with a glossary of essential terms [81]. The goal of the NIST document is to establish a unified terminology for upcoming standardizations. Meanwhile,

the ISO/IEC JTC 1/SC 42 international standard committee has been exploring various subjects within AI security. For example, the report ISO/IEC TR 24028:2020 [6] addresses threats that might weaken trust in AI systems and offers a concise discussion on how to cope with them.

Alongside the aforementioned initiatives, research has dug into techniques to prevent undesired behavior of ML models by proactively managing their training sets and parameters [6]. For instance, [86] introduced an automated framework for checking the safety of feedforward deep neural networks focusing on exploring regions around relevant data points to look for particular adversarial alterations. In [87] the focus of the research is on a technique for producing certificates of robustness for two-layer neural networks. This approach allows for simultaneous optimization of the certificates alongside the network during the training process. The research also proposed a framework designed to practically certify distributed ML powered applications. The certification process involves statistical observation of the actions performed by ML models during inference.

3.2 Cyberattacks in the ML Pipeline

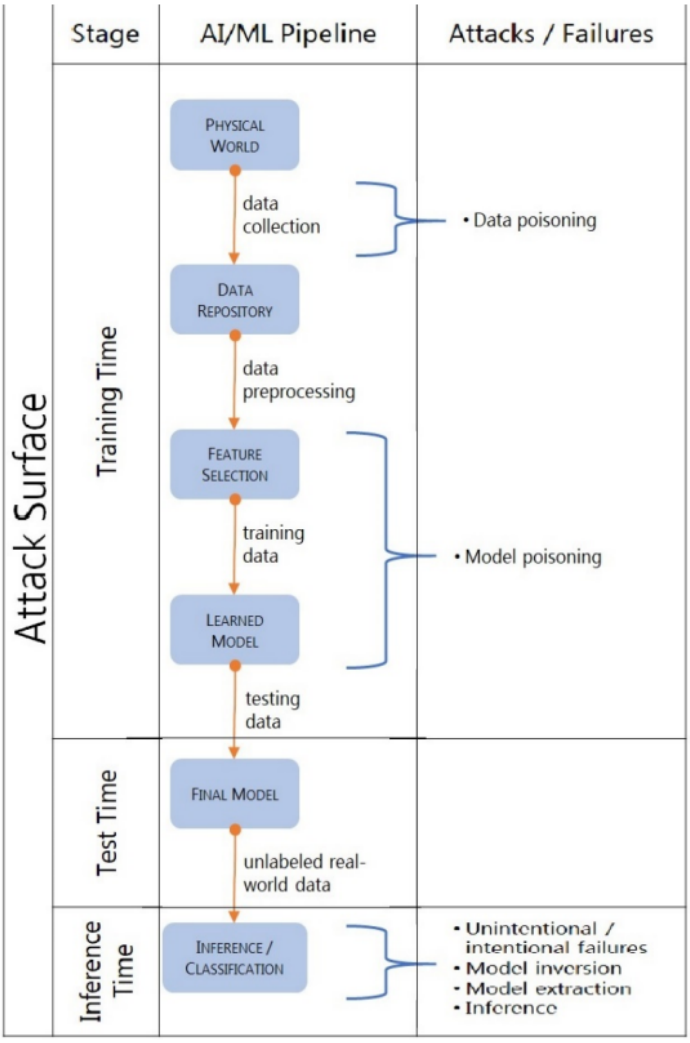

Figure 5: ML Pipeline with Cyberattacks Layout [88]

To build any ML model, data must be gathered, processed, trained, and then tested before it can be used to classify new data. This sequence of data collection, processing, training, and testing can be viewed as a generalized AI/ML pipeline, also referred to as the attack surface. An attack surface, when exposed to adversarial intrusion, might encounter a poisoning attack, an evasion attack, and an exploratory attack [89]. These cyberattacks target the three pillars of information security: Confidentiality, Integrity, and Availability, known as the CIA triad [90]. The integrity of the system is undermined by poisoning and evasion attacks. Its confidentiality can be compromised by extraction, while its availability is vulnerable to poisoning attacks. A comprehensive visualization of the entire AI pipeline and the possible attacks at each step is shown in Figure 5 [88].

3.2.1 Poisoning Attacks

Poisoning attacks arise when the adversary introduces tainted data into the training set. ML algorithms, like intrusion detection systems, often undergo retraining using the training dataset. In this type of attack, the adversaries cannot access the training dataset. However, malicious data samples [74] are introduced during the model training time. Generally, the goal of the attacker is to manipulate the AI system such that it leads to the misclassification of objects. Poisoning attacks may arise from poisoning either the training dataset or the already trained model [88], [91].

Adversaries can target the data source, a platform from which a defender pulls its data, or they might attack the database of the defender directly. They could substitute an authentic model with a compromised model. Moreover, poisoning attacks can leverage the limitations of the underlying learning algorithms [88]. This attack occurs within federated learning scenarios where the data privacy of individual users is preserved [92]. In these contexts, the adversary exploits the weakness of federated learning and may take control of both the data and learning algorithm on a specific user's device, thereby worsening the model's performance on that device [93].

Dataset Poisoning Attacks

Dataset poisoning attacks mainly appear as error-agnostic and error-specific attacks. In error-agnostic attacks, adversaries aim for a widespread impact, similar to a DoS approach, where the system generates errors randomly. This involves the manipulation of both data point features and their labels, resulting in common misclassifications and maximizing the loss function of the learning algorithm. On the other hand, error-specific attacks are more targeted, as attackers prompt the system to produce specific misclassification errors. Here, the hacker focuses on misclassifying selected data points without compromising the normal system operations, ensuring that the attack is undetected [88], [94], [95]. The essence of these attacks involves manipulating the training dataset, which results in a compromised model. When adversaries tamper with a dataset, they can induce the model to identify and adopt these manipulated patterns, serving their harmful intentions [88].

There are two main types of dataset poisoning attacks, namely data modification and data injection [96]. In data modification, the adversaries manipulate existing training data, particularly labels. For instance, attackers might randomly assign or optimise new labels to cause maximum disruption. On the other hand, in the case of data injection, the adversaries can inject new, incorrect data into the dataset, increasing the impact of the attack beyond mere label

manipulation. This is possible even without having direct access to training data or the learning algorithm [96].

Model Poisoning Attacks

Poisoning of models resembles traditional cyberattacks. When an AI system is compromised by attackers, the attackers may replace the original AI model with a poisoned one. Alternatively, they can perform "A man in the middle" attack [97] to have the wrong model downloaded while transferring learning [88]. Model poisoning generally happens through Backdoored Neural Network (BadNet) attacks [98]. BadNets are modified neural networks where the neural model is trained on a mix of clean and poisoned data. The training mechanism is thereby fully or partly outsourced. This opens up new security risks, whereby adversaries often observe the training, embedding covert triggers that lead to intentional misclassifications known only to them. There exist two main BadNet categories: (i) Outsource training attack, where training is delegated externally, and (ii) the transfer learning attack, where a pre-trained model is outsourced and utilized [88], [98].

3.2.2 Model Inversion Attacks

Given the model parameters, the model inversion attack aims to reconstruct the training data. Such attacks raise privacy concerns, particularly due to the growing number of online model repositories [88]. To illustrate, in the study [99], the model inversion attack within a blackbox context has been explored. The objective was to reconstruct an input sample considering the confidence score vector from the target model. Moreover, it has been successfully shown that it is feasible to recreate specific input samples from a given model. To accomplish this, an inversion model has been trained on a supplementary dataset designed to act as the inverse of the given target model. This inversion model used the confidence scores of the target model as input and attempted to recreate the original input data. Furthermore, it has been shown that the inversion model surpassed the performance of earlier models.

In contrast, within a whitebox context, in a study [100], a model inversion attack was introduced that only yields a representative sample from training data, rather than reconstructing a distinct input sample, guided by the confidence scores of the target model. Several related studies [100]–[102] were introduced to infer sensitive features or statistical details about the training data via inversion models. Furthermore, another study [103] focused on inversion attacks within federated learning, where the attacker had whitebox access to the model.

3.2.3 Model Extraction Attacks

An attack during which an ML model is extracted arises when an attacker, with only blackbox access to the target model, successfully duplicates it or reconstructs a model very similar to it [88]. In [104], the authors investigated the vulnerabilities in specific ML models, especially Support Vector Machines and Support Vector Regression Machines used in ML as a service. The research shows that adversaries can make use of these vulnerabilities efficiently, making it an attractive target for potential malicious attackers. In [105], attacks on neural networks were examined, demonstrating an approach where adversaries generate queries for simple deep neural network architectures.

Moreover, in [106], model extraction attacks were introduced, targeting hyperparameters of a simple architecture similar to a neural network with three layers. A particularly noteworthy extraction was shown in [107], where the extracted model was more accurate than the original model. Furthermore, the authors in [108], [109] used a model compression method called distillation, carrying out model extraction attacks on deep neural networks and convolutional neural networks, focusing on image classification.

3.2.4 Inference Attacks

ML models can accidentally expose information about the specific data entries on which they were trained [88]. In the study [110], the concept of the membership inference attack was considered. In this type of attack, an adversary can conclude whether a particular data record was included in the model's training dataset, given the data record and blackbox access to the model. According to this study, such a revelation can be regarded as a privacy breach. If the adversary can deduce whether the record was incorporated into the training through the model, then such a model is viewed as disclosing information. The significance of this privacy breach does not only impact a singular data point but resonates throughout the entire dataset, given the high correlation between the covered and the uncovered dataset [111]. This is particularly true when the model is based on statistical facts about the population [88].

Other research studies [112]–[114] focused on attribute inference attacks. In these scenarios, an attacker gets access to mostly public data about a target user to infer their private details. The attacker initially gathers data from users who are voluntarily sharing data in public. This data is then used to train a ML classifier, which uses a user's public information as input and thus estimates their private attribute values.

3.3 Failure Mode and Effects Analysis

Failure Mode and Effects Analysis (FMEA) represents a systematic and well-established approach used to discover potential failures in the design of a product or process [115], [116]. Failure modes refer to the various ways an entity, whether a process, system, or component, might fail. Effects, on the other hand, are the consequences these failures could have, leading to waste or harmful outcomes for the end users. The primary goal of FMEA is to identify, rank, and mitigate the failure modes of manufacturing or engineering processes, products, designs, or services. This is achieved by evaluating their potential occurrence, understanding the root causes, and assessing their overall impact [9], [117].

Initially developed for the military, FMEA soon expanded its reach to the aerospace industry and other manufacturing domains, with various applications in the nuclear electronics and automotive fields as well [9]. In recent times, the potential of FMEA and other safety engineering tools have been explored to evaluate the design of AI-ML systems [118], [119]. Applying FMEA in the context of an AI-ML asset involves several steps, *(i)* assigning specific functions to the asset, *(ii)* drafting its structural, functional, and network diagrams, *(iii)* defining flaws that might compromise the asset's functions or networks, and *(iv)* identifying and analyzing potential threats to the AI-ML system, and evaluating their impact on functions and networks [9].

3.4 Types of Failures in ML

There are two modes of failure of ML systems. On the one hand, there exist unintentional failures, where the AI systems can fail due to the inherent design. On the other hand, there are intentional failures, which are caused by an adversary [120]. In the following chapters, unintentional and intentional failures will be explored in depth, elucidating their various manifestations.

3.4.1 Unintentional Failures

Unintentional failures occur when AI/ML systems yield an unexpected or undesirable outcome from a determined action. It predominantly stems from system failures [88]. In this bachelor thesis, different types of unintentional failures are categorized, particularly reward hacking and distributed shift.

Reward hacking is a failure mode that arises in AI/ML systems when a reinforcement learning algorithm is used. In game scenarios, such an issue reveals itself when an agent has unexpectedly higher returns as rewards, thereby endangering the system's safety [88], [121]. In [122], a new multi-step reinforcement learning approach has been introduced. This method generates a discounted future reward, diminishing the influence of immediate reward on the current state-action pair. Such an algorithm creates a defense mechanism to mitigate the consequences of reward hacking in AI/ML systems. In contrast, the distributed shift appears when an AI/ML model that previously exhibited strong performance in one setting drastically underperforms in a different environment [88]. An example is the scenario where the training and test data come from two distinct probability distributions [123].

3.4.2 Intentional Failures

Intentional failures are caused when adversaries aim to disrupt the system by either introducing private training data to misclassify the results or extracting the foundational algorithmic framework [88], [120]. In [89], adversarial objectives are categorized into four distinct classes based on the ML classifier output. These are: *(i)* Confidence reduction, in which the predictive certainty of the target model is reduced to a lower probability of classification. *(ii)* Misclassification, where the result is altered from the original class. *(iii)* Output misclassification, wherein the adversary generates inputs to fix the classifier output into a particular class. And *(iv)* input/output misclassification, where a particular input is consistently labeled as a specific class.

In [120], a taxonomy of intentional failures/attacks has been outlined based on the knowledge of the adversary. This classification deals with the extent of knowledge required to initiate an attack to make the AI/ML system fail. The more informed an adversary [89] is, the more effectively an attack can be performed.

There are different types of classified attacks depending on the adversary's access to knowledge about the system [88]. For the purposes of this thesis, only whitebox attacks and blackbox attacks will be considered since they are relevant to the topic under discussion, and a more detailed explanation would go beyond the scope of this thesis.

In the whitebox attack, adversaries have access to the parameters of the underlying architecture of the model. Moreover, they have comprehensive knowledge of the training algorithm, weights,

distribution of the training data, and biases [103], [124]. Due to this information, the adversaries can find the model's weak spots within the feature space. Subsequently, the model is manipulated by altering inputs through crafting methods [88]. Studies in [125], [126] have demonstrated that training the model with data filled with certain adversarial instances can strengthen the system against whitebox attacks.

On the contrary, in the blackbox attack, the attacker has no knowledge about the ML system. The attacker's knowledge is limited to two types of information. On the one hand, there is the hard label, wherein the adversary obtains only the classifier's predicted label. On the other, there is confidence, where the predicted label is accompanied by its corresponding confidence score. Additionally, the attacker uses information about the setting or prior inputs to discern the vulnerabilities of the model [88], [89].

3.5 Frameworks and Approaches for Security Threats in AI and ML

The well-known MITRE Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) framework [127] was developed by MITRE to catalog adversary tactics and techniques based on real-world observations. This MITRE framework offers a standardized taxonomy of adversarial actions, facilitating a shared understanding among both offensive and defensive sides of cybersecurity. Additionally, it can be utilized as a valuable tool to analyze adversary behavior [9]. However, it is essential to note there are certain limitations here. This framework lacks explicit evidence or functionality designed specifically for modeling AI threats, highlighting a gap in addressing the primary focus of this thesis. Complementary to the MITRE ATT&CK framework is MITRE ATLAS (Adversarial Threat Landscape for AI Systems) [128]. This framework serves as a knowledge repository for adversary tactics and techniques. ATLAS includes demonstrations from red teams and security groups, real-world observations, and findings from academic research. Notably, ATLAS contains some AI/ML case studies [9]. However, there is a lack of specific details on how ATLAS incorporates AI/ML case studies and a comprehensive list of security controls.

Microsoft has made further efforts to understand and address security threats in ML-based systems. They have released guidelines for mitigating and triaging AI-specific security threats [129], [130]. The approach of Microsoft is based on STRIDE threat modeling (see Section 2.5). However, it primarily focuses on a taxonomy that classifies ML failure modes into two categories, namely intentional and unintentional failures. Intentional failures are further correlated with a list of attacks reported in the literature [9], [120]. Microsoft's approach, mainly focusing on categorizing ML failure modes, does not comprehensively cover the broader spectrum of AI threats. Additionally, its correlations with reported attacks might miss emerging or less documented threats.

OWASP has developed comprehensive strategies to understand and address security threats in AI systems [131]. Their guide contains a number of principles adressing AI security, such as use limitation, fairness, data minimization, and transparency, as explained in their AI security section (see OWASP AI Security Guide [131]). These guidelines provide a basic framework for dealing with AI safety and focus on general principles. However, this thesis seeks a tailored threat modeling approach for AI-based systems. The guide emphasizes also the importance of understanding the uniqueness of AI threats and the need for tailored security measures.

The Berryville Institute of ML has introduced an architectural risk analysis for ML security [132]. This is designed to guide developers, engineers, and others in creating applications and services

that rely on ML models. Notably, the Berryville analysis categorizes 78 risks using a generic ML system as an organizing concept and then highlights the ten most significant ones [132]. These analysis provide valuable insights into AI safety, but have their limitations, such as a potentially narrow focus and a general approach to risk categorization. Moreover, they lack specificity and adaptability to the diverse and evolving landscape of AI-based systems.

Furthermore, the European Union Agency for Network and Information System Security, also referred to as ENISA, published a report [133], focusing on a comprehensive analysis of threats targeting ML-based systems. This analysis includes threats such as data poisoning, adversarial attacks, and parameter exfiltration. The report also provides a list of security controls documented in the existing literature. Yet, this report has limited relevance to this thesis, as it concentrates on specific threats and controls for ML algorithms, without extensively covering the broader AI security ecosystem. However, another report [11], also published by ENISA, establishes a foundational comprehension of cybersecurity threats to AI [11], [129]. It highlights various assets within the AI ecosystem and associates the corresponding threats with them, considering the different stages of the typical AI life cycle. Therefore, the information presented in this ENISA report serves as a crucial foundation for the present thesis. The comprehensive and well-structured lists within these reports provide an ideal basis for further research and integration into the threat modeling approach.

3.6 Threat Modeling for AI-based Systems

Another rising area of research delves into threat modeling methodologies that can aid security experts' comprehension of how AI or ML-based systems may fail. According to [9], only a few studies have adopted this approach to identify ML security risks. However, these studies examine how threat modeling methodologies, traditionally used in software engineering, can be applied to ML-based systems' security. These investigations link threats to the components produced at different stages of ML models' life-cycle, from initial requirements analysis to system maintenance [6], [9]. The following section examines individual papers that have employed different threat modeling approaches for ML-based systems.

In [10], a conventional threat modeling approach is applied to a ML system. The methodology begins with identifying threats using Data Flow Diagrams (DFDs) and the STRIDE approach, involving five sub-steps: *(i)* An architectural model is developed for the system utilizing a DFD. *(ii)* An AML threat taxonomy is crafted based on existing literature and *(iii)* mapped to the DFD. *(iv)* Conceptual gaps between AML threats and STRIDE are addressed, followed *(v)* by the use of STRIDE to pinpoint AML threats in the ML system. In the second step, the impacts of AML threats are ranked using Microsoft AI/ML bug bar's [134] threat ranking approach. Finally, in the third step, AML threat mitigations are elicited by leveraging the Microsoft AI/ML attack library. The threat taxonomy used in this paper is not comprehensive, as it does not cover all potential threats. Another challenging aspect is the mapping process, addressing conceptual gaps between AML threats and the STRIDE approach. This approach is not straightforward, posing a significant challenge due to its complexity. Specifically, harmonizing AML threats with the STRIDE model demands careful consideration and presents a non-straightforward and somewhat cumbersome task within the overall methodology. Moreover, it is important to note that this approach is still in its early stages.

On the one hand, it may require manual implementation and lack automation. On the other hand, the evaluation, based on a single case study without any participants, may not be representative. Additionally, the usability of the method has not yet been assessed. It is reportedly planned for testing by computer science students. In light of these considerations, it becomes clear that the current state of this threat modeling approach, while being rudimentary good, is not yet mature enough for practical application in AI systems.

In [135], the focus is shifted to the maintenance phase of ML models. Here, they introduce a metric based on the notion of a "gold standard" data set to evaluate the degradation of ML models in production. The maintenance phase indicates that the ML model has already been deployed. Moreover, as previously discussed in Section 2.3, threat modeling focuses on finding threats before they become part of the system. On the other hand, the use of a "gold standard" dataset metric reflects an approach to evaluate ML model degradation in production. This is in line with traditional threat modeling principles, where metrics are typically used to quantify the impact and likelihood of identified risks and vulnerabilities. While this paper focuses on the maintenance phase of ML models after deployment, the present thesis specifically deals with threat modeling before the deployment of ML models. The assumption here is that the model has not been deployed yet, aligning with the traditional principles of threat modeling, which aim to identify and mitigate risks before they become part of the system.

In another study [136], a systematic ML-oriented threat analysis for the Open Radio Access Network (O-RAN) architecture has been conducted. They identified potential threat actors in the O-RAN ecosystem and mapped them to the requisite capabilities for an attack. This study highlights the need for domain-specific threat modeling approaches that are tailored to the particular characteristics and challenges of AI systems embedded in specific architectures. However, the objective of this thesis is to develop a general threat modeling approach where the particular architecture is not a determining factor.

The paper of [9] stands out as the most recent and arguably the most significant contribution to designing a threat modeling approach for AI-based systems. This paper advocates for the crucial integration of threat modeling methodologies in the security analysis of AI systems. Moreover, it introduces the STRIDE-AI methodology, a tailored adaptation of the STRIDE threat modeling framework specifically designed for ML systems. The methodology introduces ML-specific interpretations for CIAA security properties discussed in Section 2.4. Assigning these properties to each asset allows for reference to the corresponding STRIDE threats in Table 2 from Section 2.5.1. Furthermore, it maps the failure modes of AI assets. By associating these failure modes with specific STRIDE threats and referencing known attacks, the methodology provides a framework for identifying and addressing security challenges in ML-based systems. However, this methodology introduces a complex manual mapping process, requiring the assignment and mapping of all properties to identify STRIDE threats. Further, the procedure involves associating failure modes with threats and properties. Managing this complexity can be particularly challenging, especially for large and complex ML-based systems. The lack of automation hinders scalability and makes the method time-consuming and prone to errors. Automation is of utmost importance for handling the dynamic nature of ML systems and the continuous updates required for threat analysis. The present lack of automation in this approach decelerates threat identification and makes it less adaptive to changes in the system. Additionally, the methodology's effectiveness is evaluated based on a single use case conducted without any participants. Although this provides insights into the application in a real scenario, it does not cover the entire spectrum of challenges that can arise in various ML applications.

3.7 Threat Modeling Tools

3.7.1 Diagram-Based Threat Modeling Tools

As illustrated in Table 3, the landscape of diagram-based threat modeling tools is diverse. The Table includes both commercial and non-commercial tools, each presenting distinct approaches to threat categorization. The following section briefly explains the six tools outlined in Table 3.

(i) Microsoft Threat Modeling Tool (TMT) is a mature tool offering a default library and customizable elements in diagrams. It excels in threat identification based on the library, supporting STRIDE categorization. TMT is highly customizable, allowing for personalized threat elements and properties and benefits from community-contributed libraries. *(ii)* OWASP Threat Dragon (TD) is a lightweight tool. TD suggests generic threats, requiring user input for details. It also has a workflow integration with GitHub. Moreover, it offers limited threat library customization, making it simple and suitable for development processes. *(iii)* Open Weakness and Vulnerability Modeler (OVVL) identifies threats at both design and operation levels in web applications. It uses STRIDE for design-level threats and a CVE knowledge base for operational threats. OVVL features a customizable threat library similar to TMT. *(iv)* IriusRisk embeds diagrams.net for DFD's and provides questionnaires for configuration. It covers CVE and CWE knowledge bases and provides a comprehensive threat evaluation. Moreover, it offers a comprehensive threat library, self-defined threats, priority, and cost estimation. *(v)* Similar to IriusRisk, it uses process flow diagrams (PFDs) from an attacker's perspective. It offers a comprehensive threat evaluation supported by a robust threat library and questionnaires for configuration. *(vi)* Features AI-based predictive attack simulations with a potential attacker. It measures time-to-compromise and identifies high-value assets. However, customization options are limited [137].

Table 3: Overview of Diagram-Based Threat Modeling Tools [137]

Tool	**Non-commercial**	**Customization**	**Threat Categorization**
TMT	✓	High	Categorized by STRIDE.
TD	✓	Limited	STRIDE and additional categories (LINDDUN and CIA).
OVVL	✓	High	STRIDE at design level and CVE at operation level.
IriusRisk	×	High	CVE, CWE, self-defined threats (e.g., AWS).
ThreatModeler	×	High	Similar to IriusRisk, but with process flow diagrams.
SecuriCAD	×	Limited	N/A (Predictive attack simulations. Critical attack paths and risk evaluation).

3.7.2 Text-based Threat Modeling Tools

In this section, two distinct text-based threat modeling tools are introduced. Table 4 provides an overview of these tools, and the following text explains them in more detail: *(i)* OWASP pytm describes the system model using the Python language. Users create Python objects for system components through predefined Python classes. The tool generates a system diagram and identifies threats based on the text-based model. Additionally, it offers a comprehensive threat library with attributes such as condition, likelihood, severity, and mitigation. *(ii)* In contrast, Threagile is specifically designed for agile threat modeling. In this tool, models are represented in YAML format. Threagile provides a more concise and abstract format compared to Python. It rigorously evaluates threats using system model metrics, including relative attacker attractiveness and data loss probability [137].

Table 4: Overview of Text-based Threat Modeling Tools [137]

Tool	**Model Representation**	**Threat Evaluation**
OWASP pytm	Python object-based	Comprehensive threat library
Threagile	YAML format	System model-based metrics

3.7.3 Other Threat Modeling Tools

The landscape of threat modeling tools extends beyond diagram-based or text-based categories, including both commercial and non-commercial tools, each presenting distinct threat taxonomies. Table 5 provides an overview of these tools. The subsequent section presents brief explanations of the five tools outlined in Table 5.

(i) The CAIRIS platform offers a versatile solution for specifying and modeling secure and usable systems. Supporting manual modeling through Data Flow Diagrams (DFDs) or a top-down representation similar to attack trees, it provides a customizable threat library and facilitates risk analysis. Not strictly STRIDE-based, CAIRIS offers flexibility in threat modeling approaches. *(ii)* Threatspec, on the other hand, focuses on performing threat modeling directly on the code. This tool requires threat modeling annotations as comments in the source code and can generate DFDs and reports based on the annotated code. *(iii)* SD Elements distinguishes itself by gathering system information through surveys, eliminating the need for system diagrams. Claiming to identify threats faster than traditional methods, SD Elements relies on survey-based input for threat identification, offering a speedy approach to threat assessment without the need for diagrams. *(iv)* Kenna places a strong emphasis on data-driven threat modeling. Using data science techniques such as natural language processing and predictive modeling for risk assessment, Kenna utilizes both data science techniques and user-provided data to quantitatively evaluate threats, introducing a data-driven approach to threat assessment. *(v)* Tutamen Threat Model Automator leverages existing project design diagrams. It allows users to enter metadata into diagram elements to describe the threat model, making threat modeling more accessible by utilizing the information present in the project design diagrams. This tool facilitates threat modeling using the context of existing project diagrams, streamlining the process [137].

Table 5: Other Threat Modeling Tools [137]

Tool	**Non-commercial**	**Configuration**	**Threat Taxonomy**
CAIRIS	✓	Customizable threat library and risk analysis.	Not strictly STRIDE-based.
Threatspec	✓	Source code with threat modeling annotations in comments.	Focuses on code-based threat modeling.
SD Elements	×	Survey-based input for threat identification.	Speedy threat identification without diagrams.
Kenna	×	Utilizes data science techniques and user-provided data.	Quantitative threat evaluation with data-driven approaches.
Tutamen Threat Model Automator	×	Utilizes project design diagrams with metadata input.	Facilitates threat modeling using existing project diagrams.

3.7.4 Threat Modeling Tool for AI-Based Systems

The choice of an appropriate threat modeling tool is a critical decision, and it is pivotal to select a tool that aligns with the unique requirements and challenges of AI-based systems. Accordingly, the different types of threat modeling tools are evaluated to determine their suitability for application in AI-based systems.

Text-based threat modeling tools (see Section 3.7.2) do not seem to be intuitive regarding modeling complex AI-based systems. They also lack specific support for AI-related threats, which makes them less suitable for solving the challenges of AI-based systems and adapting other threat modeling tools (see Section 3.7.3) for AI purposes will most likely be time-consuming and challenging. Generally, these types of threat modeling tools may introduce unnecessary complexity and extra effort when applied to AI-based systems, which is undesirable in the fast-paced AI development environment. Indeed, it appears that AI-based systems require a visualization-oriented approach to better understand the threats and vulnerabilities.

Hence, within the context of this thesis, which focuses on developing a threat modeling approach for AI-based systems, diagram-based threat modeling tools (see Section 3.7.1) are particularly suitable. They offer an intuitive visual representation of threats and security concerns, which is favorable when dealing with AI models. Moreover, they are more user-friendly, making them accessible to a wider audience, including non-security experts, which is essential in the AI domain. Although diagram-based tools are not explicitly designed for AI-based systems, they can be adapted to address AI-specific threats.

Considering Table 3 in Section 3.7.1, only TMT and OVVL are viable options since they are non-commercial and adaptable. However, both are constrained to a specific threat modeling methodology making them less flexible. In contrast, diagrams.net, another diagram-based threat modeling tool, aligns with the criteria for ease of use and flexibility in capturing security concerns effectively, making it the ideal choice for the present thesis. The comparison in Table 6 confirms

the suitability of diagrams.net for AI-based systems. In direct comparison, diagrams.net stands out as notably more flexible. Moreover, it is assumed that diagrams.net is more widely known, making it the selected tool for use in this context.

Table 6: Comparison of Diagram-based Threat Modeling Tools for AI-purposes

Solution	AI-oriented	Non-commercial	AI-compatible	Agnostic	Widely Known
MTMT	×	✓	✓	×	×
OVVL	×	✓	✓	×	×
diagrams.net	×	✓	✓	✓	✓

3.8 Gaps in Literature and Solutions

The related work chapter has provided a comprehensive overview of existing research, methodologies, and tools within the domain of threat modeling and failure mode analysis in AI-ML systems. It has unveiled a diverse research landscape and demonstrates that threat modeling for AI-ML systems is a developing and dynamic field, with a growing body of research focusing on understanding the security challenges associated with these systems. Recently, researchers have adopted conventional threat modeling approaches, such as STRIDE, and tailored them to the unique characteristics of AI-ML systems.

However, particularly the discussions of the papers [9] and [10] revealed that this approach is not straightforward. In particular, the complex and manual mapping process for identifying STRIDE threats in ML systems lacks automation, hindering scalability and adaptability. Moreover, a significant drawback of these studies is their reliance on a single case study for evaluation, notably lacking participant involvement. Hence, there is still no systematic process or well-established technique for identifying vulnerabilities and threats targeting ML components throughout the entire AI pipeline [9]. This knowledge gap highlights the need to develop a practical and effective threat modeling approach for AI-ML systems. A challenge that the upcoming chapters of this thesis will address.

For this endeavor, the ENISA report from late 2020 [11] emerges as a critical resource. This report includes an AI asset taxonomy description and an AI threat taxonomy description, which are invaluable assets for shaping the foundation of the new threat modeling approach. The AI asset taxonomy description classifies various assets within the AI ecosystem, considering different stages of the typical AI life-cycle. In contrast, the AI threat taxonomy description associates threats with these assets. These descriptions, outlined in the ENISA report, will serve as essential reference points and guidance for the development of a systematic threat modeling process tailored to AI-ML systems. Moreover, based on the insights gained from 3.7.4, the process will be further backed by a diagram-based tool, namely diagrams.net.

Chapter 4

Architecture

This chapter focuses on the architecture of AI and ML systems, laying the groundwork for comprehending the structural components and life cycles that support these systems. The discussion heavily draws upon insights from the ENISA report [11], explaining key components and life cycles. It is essential to emphasize that this exploration serves as a comprehensive introduction to the architecture and plays a vital role in the threat modeling approach employed in this thesis. Additionally, this chapter introduces an AI architecture strongly inspired by an existing AI pipeline, integrating fundamental AI and ML concepts for both theoretical understanding and practical application within the thesis. This sets the stage for an in-depth AI and ML threat modeling analysis, leveraging architectural fundamentals to identify threats for AI systems systematically. Such a strategic approach contributes to improving security measures in AI-based systems, integral to the threat modeling approach outlined in this thesis.

Furthermore, this chapter focuses on the proposed threat modeling approach tailored for AI-ML-based systems, a notably challenging endeavour. As previously discussed, threat modeling is essential for identifying, evaluating, and mitigating potential security risks. A specialised approach is necessary in the context of AI or ML, where data, models, and algorithms are closely linked. Section 4.4 will introduce this specialized approach in detail.

4.1 AI Life Cycle

In this thesis a generic reference model for common AI systems (see Figure 6) is used [11], eliminating the need to construct an AI threat modeling architecture from scratch. This model's primary purpose is to establish a conceptual framework that fosters a shared understanding of the assets of AI systems and their interconnections. It facilitates the assignment of responsibilities to different assets and provides a structured approach to analyzing security threats. Provided that assets have been defined, threats can be mapped to them, enabling targeted security measures [11].

Data is a pivotal asset in AI, continuously transforming along the AI life cycle. Figure 7 demonstrates this data transformation along the various life cycle stages: Data ingestion, data exploration, data pre-processing, feature importance, training, testing, and evaluation. Along the AI life cycle, data transformation includes a range of assets, including actors, computational resources, and software. Furthermore, the data transformation encompasses intangible assets like processes, culture, and the influence of actors' experience and knowledge, which can bring potential non-intentional threats, such as a non-intentional bias [11].

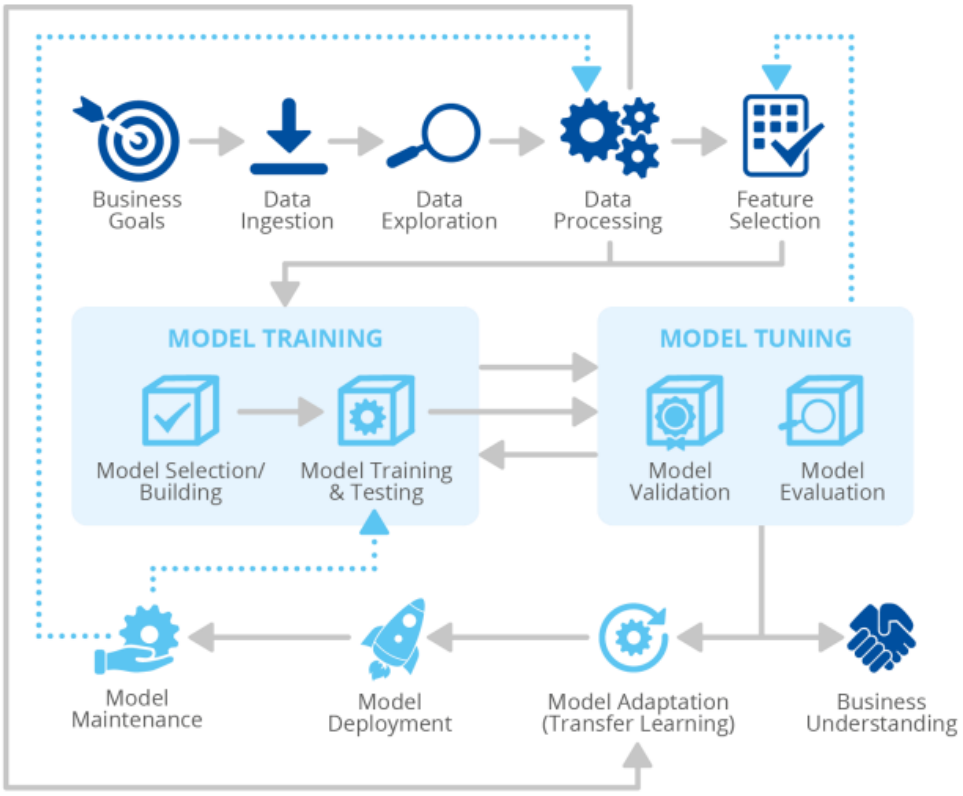

Figure 6: Generic Reference Model [11]

Figure 7: Data Transformation along AI [11]

4.2 AI Assets

In threat analysis, a critical element involves identifying the categories of assets to which threats can be posed. Assets are defined as anything that has value to an individual or organization, thereby requiring protective measures. In the realm of AI, assets are also those that are vital to fulfil the needs for which they are being used. Beyond the generic assets related to information and communication technology, such as data, software, hardware, and communication networks, AI encompasses a distinct set of assets. These include models, processors, and artefacts that can be compromised or damaged either through intentional or unintentional causes [11].

An assessment has been conducted for each stage within the AI life cycle to determine the most relevant assets. This assessment is based on the functional description of specific stages and aims to include not only the core components of AI but also the assets that facilitate the development and deployment of AI systems. Assets also encompass AI-related processes given their crosscutting nature [11]. As shown in Figure 8, assets can be classified into the following six categories : *(i)* Data, *(ii)* Model, *(iii)* Actors, *(iv)* Processes, *(v)* Environment/Tools and *(vi)* Artefacts. Furthermore, there is an overview of detailed asset taxonomy in Figure 29 in the Appendix A, providing the corresponding specific assets for each category. This taxonomy is based on the generic AI life cycle reference model discussed in the previous Section 4.1.

Figure 8: AI Assets [11]

4.3 ML-based Software System Architecture

Within the scope of this thesis, the emphasis lies especially on ML systems. To improve the comprehension and the modeling of these systems, it is valuable to compare them with traditional software systems and understand their interconnection. Figure 9 illustrates a high-level view of an ML-based software system, showing the distinct perspectives of a modern software architect when considering its subsystems. In the ML subsystem, the focus is on data, algorithms, and models. In contrast, the software subsystem deals with components, connectors, and the interactions between them. The interaction of the two subsystems is as follows: The software subsystem uses the ML models and continuously generates the required data for the ML subsystem [138].

Furthermore, each of these subsystems encompasses different stakeholders with their specific concerns. Therefore, the role of the modern software architect is to coordinate between these two subsystems, which possess distinct characteristics, properties, and team dynamics. However, this coordination process raises many questions on the different aspects of architecting, ranging from standardization of architecting practices to identifying barriers introduced by this unique architectural setup. [138].

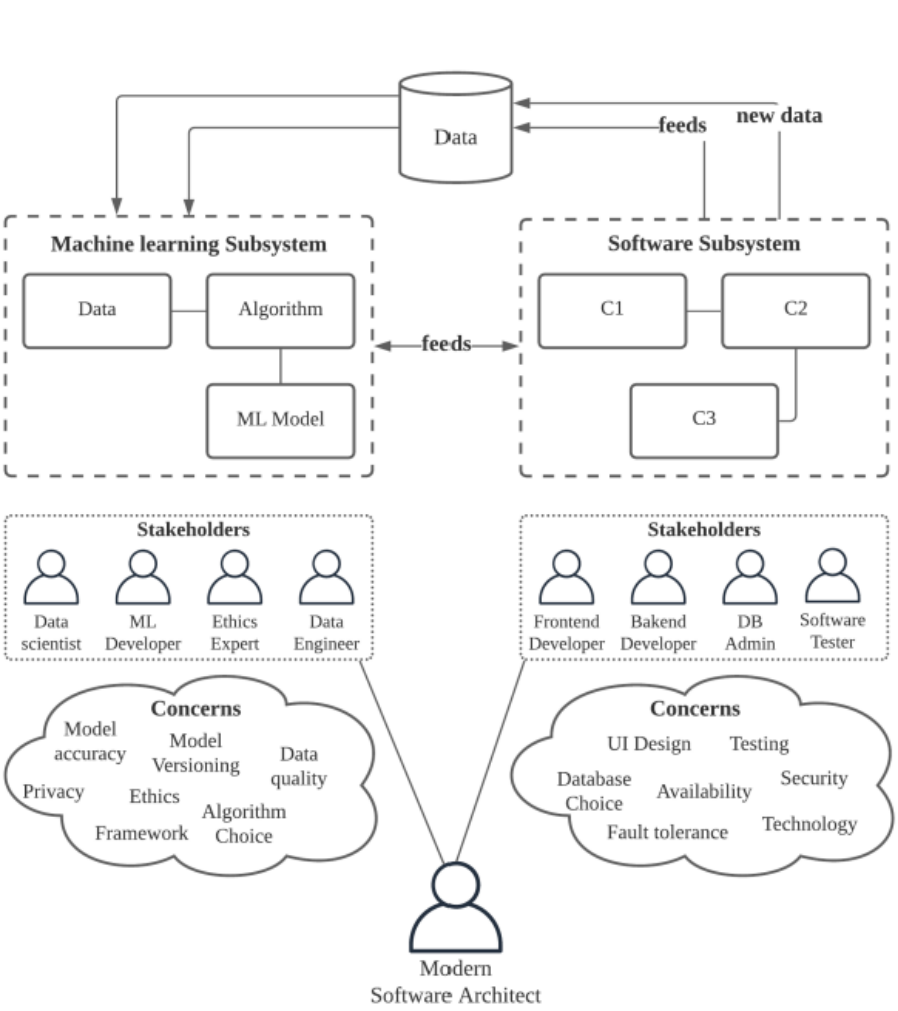

Figure 9: HighLevel ML based Software System [138]

4.4 Threat Modeling Approach for AI-based Systems

As discussed in Section 2.3, traditional threat modeling refers to the process of systematically reviewing the security of a system, where critical components are identified and their associated risks are assessed. Threat Modeling plays a crucial role in the design of information systems as it allows the identification and prioritization of issues. Furthermore, this process also evaluates the value of potential mitigation in alleviating threats [9]. A general threat modeling process comprises five sequential steps, which are listed in Table 7.

Table 7: A General 5-Step Threat Modeling Process [9], [28]

Step	Goal	Description
1	Objective Identification	Determine the system's security properties
2	Assessment	Identifies system assets and interactions
3	Decomposition	Select relevant assets
4	Threat Identification	Categorize threats to system components and assets
5	Identify Vulnerability	Analyze threats and determine vulnerability

The primary objective of the present thesis is to tailor and expand the traditional threat modeling approach to make it applicable to AI-based systems. In the context of future threat assessments for specific AI use cases, the general threat modeling methodology needs to be adjusted (see Table 8). Each step in Table 8 is explained in-depth in the following sections.

Table 8: A 5 Step Threat Modeling Approach for AI-based Systems [11]

Step	Goal	Description
1	Objectives Identification	Identify the security properties the system should have.
2	Overview/Outline	Map the system, its components, and their interactions, as well as the interdependencies with external systems (as described in Section 4.1 on AI Life Cycle).
3	Asset Identification	Determine the critical assets in terms of security that need protection (as described in Section 4.2 on Assets).
4	Threat Identification	Identify threats to assets that will lead to the assets failing to meet the aforementioned objectives.
5	Vulnerability Identification	Determine – usually based on existing attacks – whether the system is vulnerable with respect to identified threats.

4.4.1 Objective Identification

Applying traditional threat modeling methodologies to AI-based systems is not straightforward due to the different properties of AI-based systems compared to typical software systems. AI-based systems exhibit unique characteristics, including ML and dynamic decision-making, which lead to novel security concerns. Hence, it is crucial to employ CIAA properties, as outlined in Section 2.4, but with a specific focus on their applicability to AI-based systems. Consequently, the AI-specific CIAA definition [9], [11] has been adopted in this thesis. This guarantees that the threat modeling approach for AI-based systems effectively addresses AI-specific security vulnerabilities and risks, as illustrated in Table 9.

Table 9: AI-Specific Property Definitions [9]

Property	AI-Specific Definition
Authenticity	The output of the AI model can accurately be attributed to the model itself, guaranteeing that the model has generated the output.
Integrity	All the information which was used or generated along the AI Life Cycle, remains unchanged and cannot be tampered by unauthorized third parties.
Non-repudiation	There is no way to dispute or deny that the AI model's output was generated by the model, providing undeniable proof of its authorship.
Confidentiality	When utilizing an AI model for inference, it ensures that no information beyond the model's input and output is exposed or disclosed.
Availability	When provided with inputs, the AI model consistently and reliably computes useful outputs, which are clearly distinguishable from random noise.
Authorization	Only authorized parties can submit inputs to the AI model and receive the corresponding outputs.

As an initial step in the threat modeling approach, studying the AI-based system is pivotal. It is crucial to comprehend the primary functions of the system and its overarching purpose. Once the objectives are clear, the appropriate properties that need to be protected can be determined. This preliminary comprehension lays the groundwork for identifying the system's key objectives. With these objectives in mind, the subsequent tasks involve determining the specific properties that must be protected. This initial step serves as an essential foundation for the subsequent steps of the threat modeling approach.

4.4.2 Overview/Outline

In the second step, it is important to map the system, its components, their interactions, as well as the interdependencies with external systems. When engaging in the mapping process, it would be advantageous to use already beforehand the specific assets listed in the Tables 21-26 in the Appendix A as individual components of the model. Otherwise, these specific assets must be assigned retrospectively. This is explained in more detail in Section 4.4.3.

If an architectural model of the system already exists, the mapping of the system can be skipped. Nevertheless, creating a model remains a crucial aspect of the approach, contributing to a deeper understanding of the AI life cycle from Section 4.1 (see Figure 10). Drawing the model enhances familiarity with the system, improving comprehension. However, internalizing the AI life cycle or utilizing the generic AI model as a basis is immensely beneficial in either scenario. Figure 10, illustrating a high-level AI life cycle, is particularly helpful for individuals unfamiliar with AI. Here, a specific stage can be assigned to each listed component, e.g., *(i)* data management, *(ii)* model training, *(iii)* model tuning, and *(iv)* model management, as shown in Table 10. These overviews help not to lose sight of the big picture and to consistently be aware of the specific AI stage, regardless of whether or not the model exists.

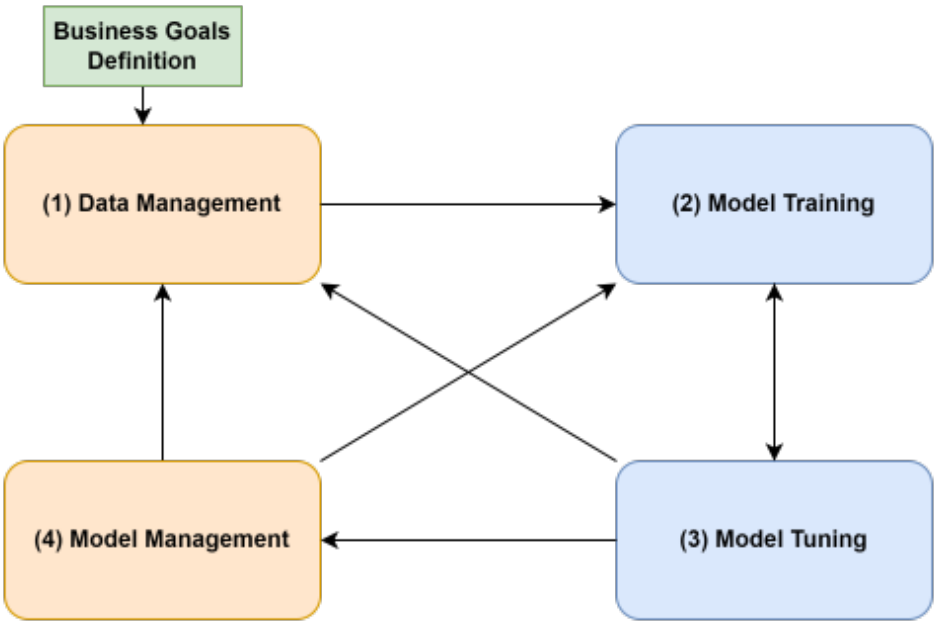

Figure 10: HighLevel AI Life Cycle

Table 10: AI Life Cycle Stages [11]

Highlevel Overview AI Life Cycle	**Specific AI Life Cycle Stage**
(1) Data Management	Business Goals Definition Data Ingestion Data Exploration Data Pre-processing Feature Selection
(2) Model Training	Model Selection / Building Model Training Model Testing
(3) Model Tuning	Model Validation Model Evaluation
(4) Model Management	Model Maintenance Model Deployment Model Adaption (Transfer Learning) Business Understanding

4.4.3 Asset Identification

Once the big picture of the system is clear, or when it is evident which properties require protection and the model is mapped, the next crucial step is to identify the critical assets. Asset identification means determining the assets that pose a potential threat to the determined properties that require protection. These assets are critical since they can cause system failure.

To begin with, it is crucial to recall the different AI asset categories from Section 4.2. Figure 8 provides an overview of the six different AI asset categories. For each of these categories, specific assets can be assigned to a category. Moreover, these specific assets belong to a specific AI life cycle stage, as shown in the Tables 21 to 26 found in the Appendix A. The descriptions of the specific assets are not listed in the Tables since their detailed descriptions make it impractical to include all details in this context. For a holistic overview, it is advisable to refer to the ENISA report[11].

In scenarios with an existing architecture model of an AI system or one that was mapped without using the specific assets, it becomes obligatory in this step to allocate the corresponding equivalent specific assets in the critical areas of the model. Upon identifying the critical assets within the model, it is essential to replace them with the corresponding specific assets listed in the Tables 21-26 in the Appendix A. This is because the foundation of the new threat modeling approach for AI-based systems relies on the asset taxonomy and threat taxonomy outlined in the ENISA report [11]. In this framework, specific assets are assigned to a category, and the threats can then be determined in a further step by using this category (see Section 4.4.4). Furthermore, it is evident that in scenarios where the model has been designed with the specific assets, there is no need to undertake the assignment of these specific assets all over again.

4.4.4 Threat Identification

In alignment with the asset taxonomy, the ENISA report [11] also presents a comprehensive threat taxonomy. Table 11 illustrates eight distinct categories within this taxonomy, each pursuing different objectives. These categories, in turn, encompass various specific threats. Table 12 serves as an illustrative example, presenting details such as threat category, specific threat, description, potential impact, and affected assets. It is crucial to note that each of the eight threat categories includes numerous specific threats, although they are not exhaustively listed here. For a holistic overview of the specific threats, it is advisable to refer to the ENISA report [11]. The extensive nature of these specific threats and their detailed descriptions make it impractical to include all details in this context.

Table 11: Threat Categories and Descriptions [11]

Threat Category	**Description**
Nefarious activity/abuse	Malicious acts targeting ICT systems, infrastructure, and networks aiming to steal, alter, or destroy a specified target.
Unintentional Damage	Unintentional actions leading to destruction, harm, or injury of property or persons, resulting in failure or reduced usefulness.
Legal	Legal actions by third parties to prohibit or compensate for losses based on applicable law.
Failures or malfunctions	Incomplete or total malfunction of a hardware or software asset.
Eavesdropping Interception Hijacking	Actions aiming to listen, interrupt, or gain unauthorized control of third-party communication.
Physical attacks	Actions with the aim to harm, expose, alter, disable, steal, or gain unauthorized access to physical assets, including infrastructure, hardware, or interconnection.
Outages	Unexpected service disruptions or quality degradation below required levels.
Disasters	Sudden accident or natural catastrophe causing great damage or loss of life.

The next step involves using the identified assets from Section 4.4.3 to identify potential threats. As described in Section 4.4.3, there exists a category corresponding to each specific asset. Specifically, data, model, actor, processes, environment/tools, and artefacts. Thus, these categories can be clearly identified in the asset taxonomy based on the specific assets. Notably, these same categories are also listed in the threat taxonomy as affected assets. This conformity allows for a seamless connection between the two taxonomies through the use of categories, as illustrated in Figure 11. Moreover, this connection enables a comprehensive display of all associated threats. This implies that specific threats associated with the categories from the asset taxonomy are systematically presented across all threat categories. In an additional step, threats can be filtered based on the corresponding properties described in Section 4.4.1, listed explicitly under 'Potential impact' in the threats taxonomy (see Figure 11). This process results in the display of only those threats that align with the filtered and relevant properties. Hence, the new approach effectively closes the gap in AI-ML threat modeling by optimizing the complex, mapping process.

Utilizing the asset and threat taxonomies from the ENISA report, it establishes a systematic link between specific assets and corresponding threat categories. This method not only simplifies the intricate process but also ensures a comprehensive and clear display of related threats, thereby enhancing the effectiveness and clarity of threat identification in AI-ML systems.

Table 12: Threat Taxonomy Specific Example [11]

Threat Category	**Specific Threat**	**Description of the Specific Threat**	**Potential Impact**	**Affected Assets**
Nefarious activity/abuse	Data poisoning	Injection of incorrect data into the training or validation sets, achieved through exploiting poor authentication/authorization mechanisms. The aim is to adversely affect the operation of the AI system.	Integrity, Availability	Process, Environment/tools, Model
...	...	...	...	...

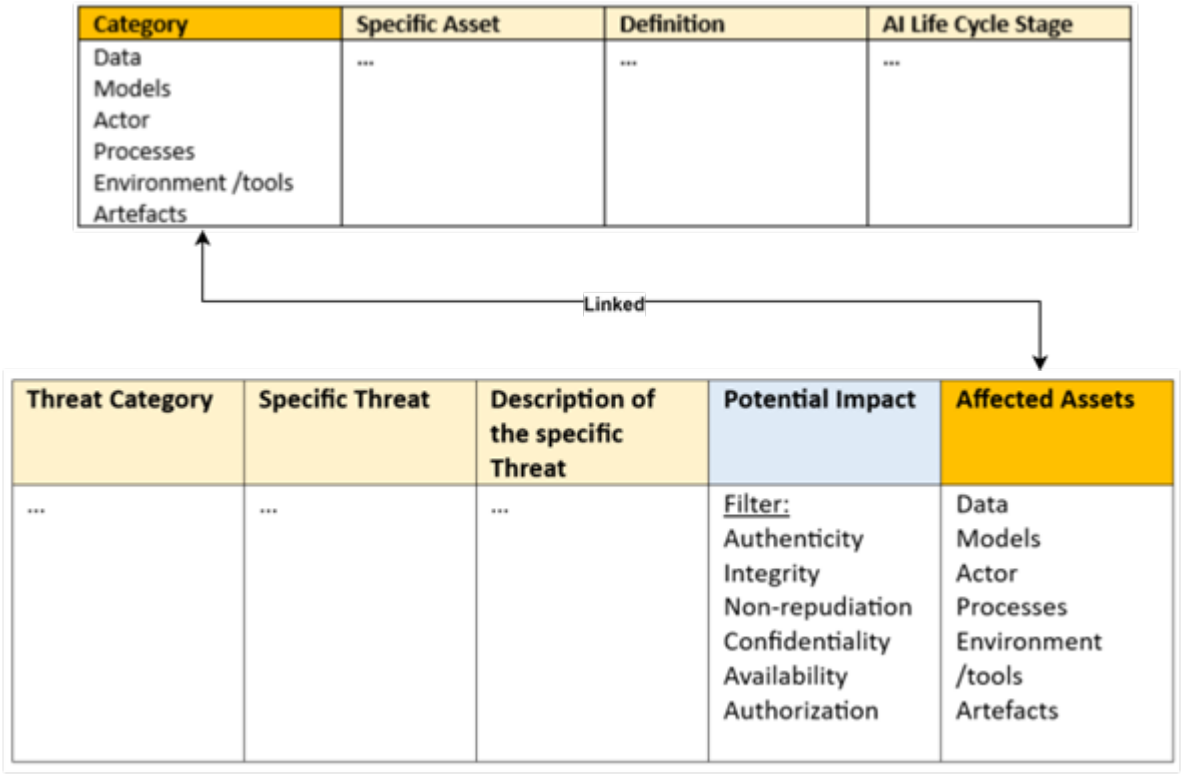

Figure 11: Linked Assets and Threats Taxonomy

4.4.5 Vulnerability Identification

Once the critical assets and their corresponding threats are identified, the final step of identifying vulnerabilities within the AI-based system follows. As explained in Section 2.1, vulnerabilities are weaknesses in a system's architecture or in its design that might be exploited by threats. To detect vulnerabilities, it is crucial to analyze the identified threats meticulously. Each specific threat exhibits weaknesses in its associated specific assets or the specific AI life cycle stages. As a possible manual vulnerability identification approach, let's consider the threat data poisoning from Table 12 in Section 4.4.4. Due to the description of the threat, it is evident that the system indicates a vulnerability linked to poor authentication/authorization mechanisms. Furthermore, the threat manifests during the AI life cycle stages of model training or model tuning. With the critical assets already identified, the comparison with the corresponding stages facilitates the identification of potential vulnerabilities within the system.

A more popular and structured technique involves the use of bug bars, presented in the form of Tables listing the criteria used to classify bugs [9]. Recently, Microsoft has introduced a bug bar [139] for ranking ML threats, specifically addressing intentional malicious activities against ML-based systems. However, explaining threat prioritization bug bars to users without security expertise can be challenging [9].

Another easier-to-apply method for prioritizing threats and identifying vulnerabilities is the DREAD model, detailed in Section 2.5.1. In Table 13, specific values can be assigned for each mnemonic element. Calculating the average value for each threat makes prioritization in descending order feasible. The identified prioritized threats can then be treated as potential vulnerabilities. As noted in [9], the DREAD system is no longer exclusively recommended due to its inherent subjectivity in the rating process. However, it is still utilized for rapid preliminary threat assessments [9].

Hence, vulnerability identification methods might vary based on user expertise. The bug bar method is particularly designed for ML threats and, therefore, the most suitable one. Combining the DREAD method with a manual approach, while the latter is aimed more at expert users, could also be a vulnerability identification strategy. On its own, DREAD remains the most straightforward method but only provides a superficial evaluation accessible to users. In the context of this thesis, participants in the experiment outlined in Section 6.2 are not asked to make this step due to its extensive nature, which would also exceed the scope of this thesis. However, these methods could be applied in future research.

Table 13: Threats Prioritization DREAD

Damage Potential	**Reproducibility**	**Exploitability**	**Affected Users**	**Discoverability**	**Average**
(0, 5, 10)	(0, 5, 10)	(0, 5, 10)	(0, 5, 10)	(0, 5, 9, 10)	...

Chapter 5

Prototypical Implementation

This chapter shows the practical implementation of the theoretical threat modeling approach for AI-based systems from Section 4.4. The core concept is to develop an automated tool that recognizes threats based on identified critical assets. The tool is designed to generate a comprehensive output, displaying the specific identified assets with their corresponding potential threats. The tool is named *ThreatFinder*, aligning with its main functionality. The details are explained in more detail in the following sections.

5.1 Diagrams.net in *ThreatFinder*

First of all, it is important to have an architecture modeling basis, i.e., to model the AI system or to modify an existing one. Ideally, the tool chosen for this task should be intuitive and user-friendly. In this context, diagrams.net is a suitable choice. As previously discussed in Section 3.7.4, diagrams.net, a diagram-based threat modeling tool, aligns with the criteria for ease of use and flexibility in effectively capturing security concerns. Although not explicitly designed for AI-based systems or threat modeling, diagrams.net is AI-compatible, making it the optimal choice as a supportive tool. The AI-based system architecture model can thus be created or imported into diagrams.net. Moreover, diagrams.net provides various options for importing an existing model, e.g. using an XML file.

To create the model from scratch or to mark the critical assets within an existing model, it is crucial to create libraries encompassing the various assets listed in Tables 21 to 26 in the Appendix A. For this purpose, one XML file was created for each of the six categories, containing all the specific assets for each category (see Figure 12 as an illustrative example). Each file contains pertinent information about the assets, such as their object label, asset name, ID, XML structure, dimensions (width and height), aspect ratio, and title. In addition, distinctive styling was applied to each XML file to enable clear differentiation between the corresponding specific assets. Figure 14 depicts the results from applying the distinct styling.

Figure 12: Category Data XML File Extract

These XML files can be uploaded to diagrams.net, where each category is associated with a different symbol. The specific assets within each category are represented by corresponding symbols, as illustrated in Figure 14. All the specific assets from the asset taxonomy listed in the ENISA report are available or applicable. When creating an architecture model of an AI system from scratch, the relevant critical assets from the libraries can be directly utilized. The architectural model can be created using the standard diagrams.net tools, incorporating critical components directly from the dedicated library. It is crucial to utilize the specific assets from the library, especially when they may pose a threat to the system. Such a modeling process from scratch aligns with the intended approach for the threat modeling process. However, in the case of an already existing architecture model, the existing critical components need to be annotated with an equivalent specific asset from the libraries, as depicted in Figure 13, where the pre-processed data set is placed over the existing model component. Nothing needs to be deleted with this action, the replacement itself is sufficient.

Thus, diagrams.net is a crucial foundation for modeling the AI-based system and identifying assets. Therefore, the initial three steps of the threat modeling approach can be executed using diagrams.net and the mentioned libraries, whereby the first step of identifying the objectives is carried out independently. The significance of the AI architecture model and the integration of the specific assets from the library with regard to threat identification will be explained in the following section.

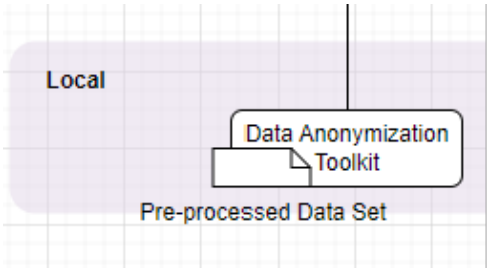

Figure 13: Asset from Library Overwriting Existing Component

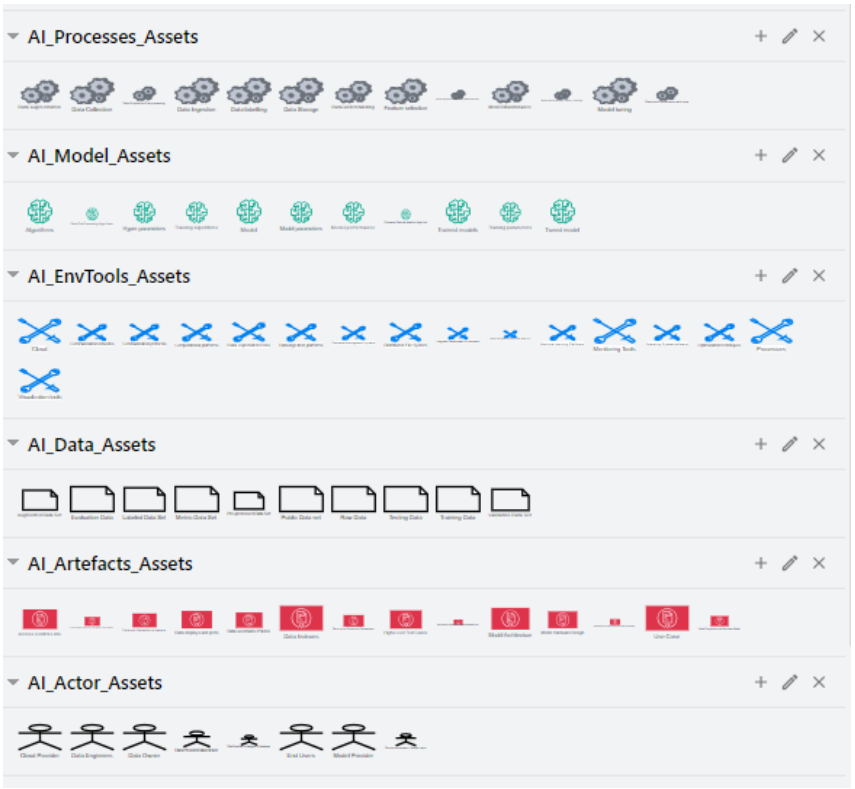

Figure 14: Imported Assets as XML

5.2 Functionality and Structure of the *ThreatFinder* Tool

Once the initial three steps have been accomplished, the subsequent and most challenging step, threat identification, must be mastered. It is important to emphasize that the threats must be identified based on of the architecture model of the AI system or the specific assets identified. To address this challenge, the complete asset and threat taxonomy were recorded in separate JSON files. Each entry in the asset taxonomy JSON file includes ”Category”, ”Asset”, ”Definition”, and ”AI Lifecycle Stage”, as shown in Figure 15. In contrast, each entry in the threat taxonomy JSON file includes ”Threat Category”, ”Threat”, ”Description”, ”Potential Impact”, and ”Affected Assets”, as shown in Figure 16. As previously shown in Section 4.4.4 in Figure 11, the JSON files can be interconnected through ”Category” from the asset taxonomy and ”Affected Assets” from the threat taxonomy.


```
[
    {
        "Category": "Data",
        "Asset": "Augmented Data Set",
        "Definition": "An augmented data set is a (usually labe
        "AI Lifecycle Stage": "Data Pre-processing"
    },
    {
        "Category": "Data",
        "Asset": "Evaluation Data",
        "Definition": "The evaluation data is used to evaluate
        "AI Lifecycle Stage": "Model Tuning"
    },
    {
        "Category": "Data",
        "Asset": "Labeled Data Set",
        "Definition": "The term “Labeled Data” refers to a set
        "AI Lifecycle Stage": "Data Pre-processing"
    },
```


Figure 15: AssetTaxonomy JSON Extract


```
[
    {
        "Threat Category": "Nefarious Activity/Abuse",
        "Threat": "Access Control List (ACL) manipulation",
        "Description": "In AI data collection scenarios, group-based A
        "Potential Impact": "Integrity",
        "Affected assets": "Artefacts"
    },
    {
        "Threat Category": "Nefarious Activity/Abuse",
        "Threat": "Adversarial examples",
        "Description": "Targeting the inference phase of ML and deep l
        "Potential Impact": "Integrity, Availability",
        "Affected assets": "Model, Data"
    },
    {
        "Threat Category": "Nefarious Activity/Abuse",
        "Threat": "Backdoor/insert attacks on training datasets",
        "Description": "Threaten the ML model's integrity by trying to
        "Potential Impact": "Integrity",
        "Affected assets": "Model, Data"
    },
```


Figure 16: ThreatTaxonomy JSON Extract

This interconnection enables the exposure of threats based on specific assets. In this process, the specific assets are assigned to their respective categories, and due to the ”Category”, which can be linked with the ”Affected Assets”, all corresponding threats are identified and presented (the algorithm for this is shown in Figure 21). More specifically, the XML architecture model, which has been edited with the relevant assets from the libraries, serves as input for the tool. The assets, represented as objects in the XML file (see Figure 17), are then compared by the tool with those in the database. If the objects match, corresponding threats are then issued.

```
<object label="Evaluation Data" assetname="Evaluation Data" id
  <mxCell style="shape=cube;whiteSpace=wrap;html=1;boundedLbl=
    <mxGeometry x="436.5" y="855" width="87" height="50" as="g
  </mxCell>
</object>
<object label="Augmented Data Set" assetname="Augmented Data S
  <mxCell style="shape=cube;whiteSpace=wrap;html=1;boundedLbl=
    <mxGeometry x="152.5" y="985" width="87" height="50" as="g
  </mxCell>
</object>
<object label="Evaluation Data" assetname="Evaluation Data" id
  <mxCell style="shape=cube;whiteSpace=wrap;html=1;boundedLbl=
    <mxGeometry x="296.5" y="984" width="87" height="50" as="g
  </mxCell>
</object>
```

Figure 17: Object Identification in XML

Figure 18 shows the structure and the technologies used by the *ThreatFinder* tool. The choice of technologies and the application structure are briefly explained: Developing a web application was a plausible solution for implementing the *ThreatFinder* tool. In this context, the front end is built with React JS, an open-source JavaScript library known for building interactive user interfaces. React stands out as a leading technology, especially due to its inherent virtual DOM feature that enhances the overall user experience with its rendering speed. Moreover, React's flexibility and modularity make code maintenance effortless and pave the way for potential future expansions of the tool [140], [141]. The backend was developed using Python and FastAPI. Python's versatility, ease of use, and readability make it an excellent choice for backend development. Additionally, the FastAPI web framework adds speed and simplicity to the process. FastAPI is intuitive and facilitates the creation of robust APIs, ensuring smooth communication between the frontend and the backend [142], [143]. For data storage, a straightforward approach has been chosen, saving the taxonomies in two JSON files as databases. JSON files are easy to read and manage, providing a convenient solution for storing and retrieving data in the *ThreatFinder* tool. This choice of technologies guarantees a responsive and interactive user interface in the front end and also ensures effective data processing and communication with the backend. In summary, these technologies are well-suited for the implementation of this project. The open-source repository for the project, along with its installation guidelines, is provided in the Appendix C.

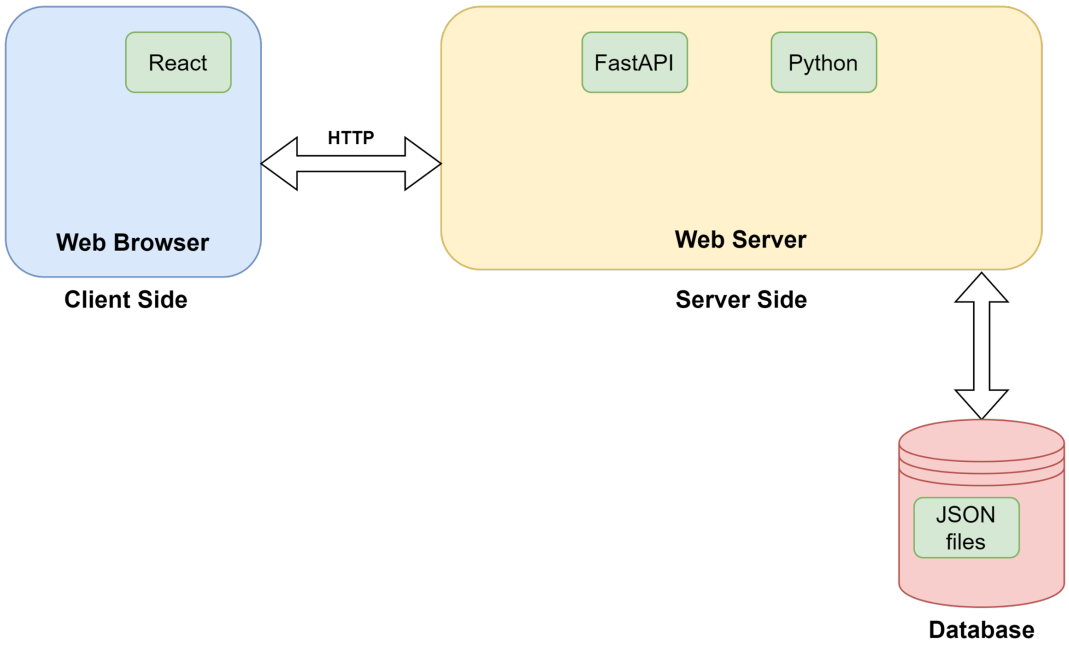

Figure 18: *ThreatFinder* Structure

The client-side interface, depicted in Figure 19, encompasses several interactive elements. Users have the option to choose and upload an XML file. Furthermore, the exhibited threats can be dynamically filtered based on various properties. Additionally, users can conveniently download the displayed threats as a PDF document. On the server side, the FastAPI application defines an endpoint */upload* to handle HTTP POST requests, as illustrated in Figure 20. This endpoint expects a file to be uploaded, specifically an XML file, and it uses the *UploadFile* class from FastAPI's File module to handle file uploads. The uploaded file content is then read, and the *find_threats* function is called to analyze the content and identify threats. Upon processing the file and identifying threats, the application returns a JSON response containing the threat results. The response is encoded using *jsonable_encoder* to ensure compatibility with JSON format. Overall, this FastAPI application serves as a backend service for handling file uploads, analyzing content, and providing threat information in response to client requests.

Furthermore, on the server-side backend, three key methods collaboratively contribute to identifying and retrieving information about threats based on the provided asset name. Figure 21 illustrates these three methods. The $find\_category\_for\_asset(asset\_name)$ method takes an asset name as input, converts it to lowercase for case-insensitive matching, and searches the asset taxonomy. It returns the category of the asset if found. Given an asset category, the $find\_threats\_for\_category(category)$ method iterates through the threat taxonomy to find threats associated with the specified category. It returns a list of threat objects, each containing information such as threat category, description, potential impact, and affected assets. The $get\_threats\_by\_asset(asset)$ method combines the previous two methods. It finds the category of a given asset using $find\_category\_for\_asset(asset\_name)$, and then retrieves threats associated with that category using $find\_threats\_for\_category(category)$. The result is a dictionary containing the identified asset, its category, and a list of threat objects associated with it. The asset taxonomy and threat taxonomy (refer to Figures 15 and 16) are stored in the database as JSON files.

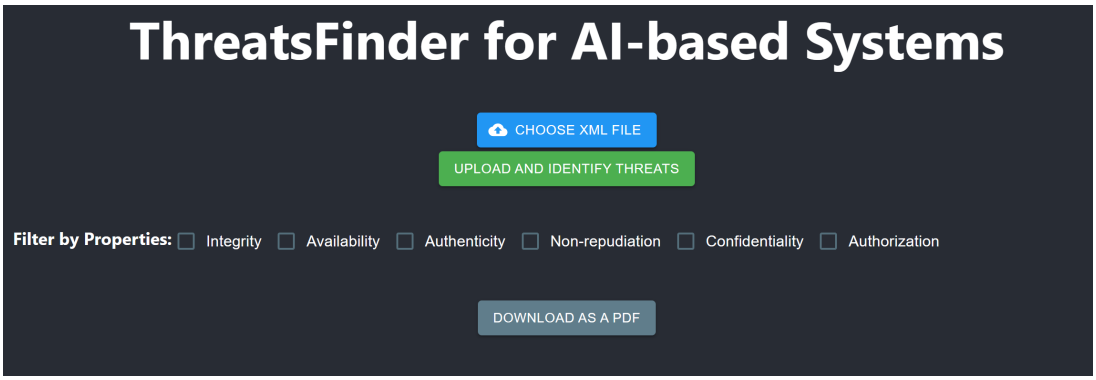

Figure 19: Frontend *ThreatFinder*

```
JSha91
@app.post("/upload")
async def upload_file(file: UploadFile = File(...)):
    content = await file.read()
    threats_result = find_threats(content)
    return JSONResponse(content=jsonable_encoder(threats_result))
```

Figure 20: FastAPI handling Files

```
1 usage  JSha91
def find_category_for_asset(asset_name):
    asset_name_lower = asset_name.lower()
    for asset_entry in asset_taxonomy:
        if asset_entry['Asset'].lower() == asset_name_lower:
            return asset_entry['Category']
    return None


1 usage  JSha91
def find_threats_for_category(category):
    threats = []
    for threat_entry in threat_taxonomy:
        if category in threat_entry['Affected assets']:
            threats.append({
                "Threat Category": threat_entry["Threat Category"],
                "Threat": threat_entry["Threat"],
                "Description": threat_entry["Description"],
                "Potential Impact": threat_entry["Potential Impact"],
                "Affected assets": threat_entry['Affected assets']
            })
    return threats


3 usages  JSha91
def get_threats_by_asset(asset):
    category = find_category_for_asset(asset)
    asset_with_threat_objects = {'Identified Asset': asset,
                                 'Category': category,
                                 'Threats': []}
    if category:
        threats = find_threats_for_category(category)
        if threats:
            asset_with_threat_objects['Threats'] = threats
    return asset_with_threat_objects
```

Figure 21: Identifying and Retrieving Threats Algorithm

Chapter 6

Evaluation

In this chapter, the *ThreatFinder* tool is evaluated to assess its functionality and usability in the context of the first-ever effective and automated threat modeling tool for AI systems. Participants with different educational backgrounds and varying AI expertise levels participated in an experimental evaluation of the tool. This experiment was conducted within the context of the AiSym4MED, a scientific project that provided the use case for the evaluation. They completed predefined tasks that assessed their familiarity with the tool, their understanding of the AI features, and the overall usability of *ThreatFinder*. The chapter focuses on the detailed results, analyzing participant responses, challenges encountered, and the implications for the tool's improvement. Furthermore, future work for optimizing the threat modeling approach for AI systems is explored, providing possible areas for improvement.

6.1 Use Case AISym4MED

AISym4MED strives to create a modern platform aimed at engineers, practitioners, and researchers analyzing healthcare data. This platform is designed to offer a reliable dataset system enhanced with controlled data synthesis, fostering experimentation and modeling within the healthcare domain. Furthermore, this platform will improve data privacy and security by integrating new anonymization techniques, attribute-based privacy measures, and trustworthy tracking systems [144]. The functionalities and the corresponding components can not yet be depicted, since it is part of a report that has not been published yet. However, in a few months a new version of the report [145] will be published, and the Figure with the functionalities can be found there. Subsequently, the functionalities and the most important components are briefly and concisely explained to facilitate an understanding of the architecture model's mechanism.

The main functionalities of the AISym4MED platform can be clustered in ML models through *(i)* three main modules: Model Training, Model Auditor and Data Synthesizer *(ii)* seven supporting modules: Data Anonymization Toolkit, Data Uploader, Cross-Borders Database, Dataset Explorer, Dataset Builder, Dataset Evaluator and Federated Learning, and *(iii)* three transversal modules needed to cover all the functionalities and expected requirements: Cyber-Security Control, User Interface, and Orchestrator [146]. The components essential to the architecture model are described as follows:

- Data Anonymization Toolkit: The Data Anonymization Toolkit addresses privacy concerns in healthcare data. This standalone toolkit that can be used outside of the platform ensures

that healthcare institutions maintain control over the anonymization process before data is uploaded to the platform and that sensitive patient data is protected from potential data breaches right from the start. Employing new anonymization techniques safeguards against re-identification attacks, where an individual's identity can be inferred from their data. The user-friendly and configurable design enables healthcare institutions to tailor the anonymization process to their specific needs, seamlessly integrating with existing data processing pipelines and other management systems [146].

- Data Uploader: The Data Uploader module facilitates a user-friendly process for entities to upload anonymized healthcare data to the platform's secure storage. It enables the manual selection of data files through an intuitive frontend. Furthermore, the module includes a Minimum Anonymization Verifier that automatically evaluates uploaded data, ensuring compliance with AISym4MED's strict anonymization standards. This verifier acts as a safeguard against inadvertent breaches or human errors. It notifies the entities to anonymize data further if personal identifiers are detected. In summary, the Data Uploader module ensures efficient and secure data contribution while upholding patient privacy [146].

- Cross-Border Database: The Cross-Border Database module is pivotal for integrating multiple heterogeneous data sources in different regions or countries and converting data formats into a common Data Storage system. Therefore, it must deal with data harmonization challenges due to the heterogeneity of data sources. It provides a unified view of healthcare data, supporting quality assessment and training of AI models. The module implements a Multi-party Data Governance architecture, ensuring data sharing complies with GDPR and privacy requirements [146].

- Dataset Explorer: This module provides support for exploring existing data. It is an essential tool for healthcare professionals, facilitating easy navigation and search within large datasets. It features a custom, scalable data mining search engine with innovative algorithms for accurate and relevant results. The module supports various search types, including Keyword-based, Context-aware, and Reverse Searches [146].

- Dataset Builder: The Dataset Builder is essential for researchers, allowing the building of a high-quality Meta-dataset for AI model training. Making use of the results from the Dataset Explorer enables the selection of relevant datasets from various sources, offering transformation and harmonization techniques for standardization. Furthermore, the Meta-dataset Creator function allows users to define and specify the attributes of the dataset they want to utilize [146].

- Dataset Evaluator: The Dataset Evaluator is a core component of the platform, providing insights into data quality and GDPR compliance. It employs state-of-the-art and project-specific techniques to assess data quality, including performance and fairness metrics. The module characterizes datasets of various types, offering different statistics on data distribution and primary limitations. The evaluation covers both real anonymized and synthetically generated datasets, requiring a robust big-data workflow. Moreover, it assists in identifying and addressing errors, inconsistencies, and biases, ensuring the highest quality Meta-dataset for AI applications. The Dataset Evaluator is, therefore, crucial for anyone using medical data in AI applications [146].

- Model Training: The Model Training module is crucial for creating and training both generative and predictive AI models. It includes a Serialized Model Handler for uploading models in defined formats like HDF5, Pickle, JSON, and others. Furthermore, this module

includes the Training Config function, which allows users to configure the ML Operations scheme (MLOPs), fine-tune hyper-parameters, and configure data sources, utilizing on-demand deployment with Graphical Processing Unit support for efficiency. Moreover, the module incorporates an MLOps Database to define and track every experiment, providing in-depth insights into training configurations, model architecture, and evaluation metrics. It enables users to monitor the progress of AI models during training, with experiments that can be compared to select the optimal model [146].

- Model Auditor: The Model Auditor module is crucial for the thorough evaluation of trained models. The main objective of the Model Auditor module is to enable the analysis and characterization of previously trained models on the Model Training module. It analyzes model performance, limitations, biases, and compliance. By integrating metrics, it harmonizes model performance, robustness, and fairness. The module provides diverse performance metrics, aiding in optimizing models and understanding breakdowns. Furthermore, it helps users to identify model biases and limitations, aligning with model objectives ensuring effective and reliable AI solutions. Additionally, it checks models to be compliant with regulations. Moreover, it generates a Model Card providing a full characterization of the audited model [146].

- Data Synthesizer: The Data Synthesizer module generates artificial data using pre-trained Generative Models from various medical domains. This covers various data types like tabular, time-series, image, and unstructured data. The Generative Model Inference allows the generation of new data instances as needed, expanding the dataset for training Predictive AI models. The module guarantees quality through a Synthetic Data Validator, employing both quantitative metrics and human feedback validation. The latter involves comparing synthetic data with real datasets, providing valuable feedback for fine-tuning Generative Models using Reinforcement Learning from Human Feedback. This comprehensive evaluation approach ensures high-quality synthetic data and, therefore, can be used for effective ML tasks [146].

- Federated Learning: The Federated Learning module enables cooperative model creation using data from several entities, facilitating the execution of data processing and model training tasks. Employing edge computingensures data privacy by keeping original data in its place of origin and prevents sharing it with other entities. This module supports model inference to audit models using local data from an external entity. Thereby, this module uses the Data Loader and Data Harmonizer functionalities for diverse data sources. To ensure the privacy of input and output data, a Privacy-Preserving functionality uses cryptographic techniques, verifying encrypted intermediate data access only by authorized parties, thereby minimizing the risk of data privacy breaches [146].

- Cyber-Security Control: This module ensures a holistic defense against various attack vectors, managing user authentication, authorization, and role-based access control. It applies a Public Key Infrastructure for secure user authentication, enabling tracking and logging of operations for non-repudiation and ensuring integrity verification in Federated Learning. Defense against outside threats includes employing a Web Application Firewall for web-based security. Furthermore, this module includes a vulnerability functionality for active security weaknesses. Additionally, the module integrates a Policy Engine, coupled with the Orchestration module, to control user, role, module, and instance aspects on the platform [146].

- User Interfaces: The Interface module serves as the main entry point to the AISym4MED platform, offering a web-based graphical user interface (GUI) that allows users to interact with all the modules and functionalities. Designed to be user-friendly and adaptive, the GUI also provides a visual representation of the results of the user's actions. Moreover, it integrates interactive visualization tools for a deeper comprehension of data. Additionally, with a user-centric approach, the design incorporates user personas, journeys, and stories to ensure adaptability to different user roles and tasks [146].

- Orchestrator: The Orchestrator module is pivotal in the AISym4MED platform. This module is responsible for deploying, scaling, and controlling the various modules across numerous instances, ensuring smooth operation and effective resource use. It dynamically allocates resources based on user needs. It thereby ensures efficient resource use and responsiveness to changing user demands. Furthermore, it allows easy monitoring and integration of new modules, promoting modularity and extensibility due to a centralized interface. Additionally, other abilities of the Orchestrator are to automate complex workflows, improve data processing efficiency and reduce execution time and costs [146].

6.2 Experiment with the AISym4MED Architecture Model

An experiment was conducted to evaluate the effectiveness of the new threat modeling approach for AI-based systems described in Section 4.4. The AISym4MED architecture model and the libraries encompassing the various assets were provided to seven participants. In addition, the participants were given a video tutorial and a guide with step-by-step instructions, including the description in Section 6.1. The participants had to apply the approach explained in Section 4.4 to the provided AISym4MED architecture model. Subsequently, they had to upload their modified model and report and complete a questionnaire on Google Forms. These collected results are the foundation for the evaluation of the *ThreatFinder* tool.

6.2.1 Methodology and Goals

The participants commenced by uploading the provided AISym4MED architecture model to diagrams.net and importing the asset libraries. Following this, they studied the AISym4MED architecture model description, referencing AI properties from Table 9 of Section 4.4.1. Participants selected properties crucial for safeguarding the AISym4MED system's flawless functionality and secure environment. Subsequently, they assigned assets to the architecture model, placing equivalent assets in identified critical areas. The final steps included uploading the edited model to the *ThreatFinder* tool as an XML file, selecting corresponding properties, and sharing results as a PDF file. It is important to note that the participants had the opportunity to seek clarification during the experiment. Notably, only participants six and seven, both without a computer science background (refer to Table 14), took advantage of this option. Participant seven, in particular, raised numerous questions about asset assignments, initially assigning all assets and later modifying the assignments due to support. In contrast, participant six specifically inquired about AI, focusing on understanding the data flow within the AISym4MED architecture model.

After conducting the threat modeling approach, participants filled out a Google Forms questionnaire. They were queried about their familiarity with threat modeling tools, including diagrams.net, TMT, and OVVL, rating their proficiency on a scale of 1 to 5, ranging from *"not familiar at all"* to *"very familiar"*. Furthermore, participants were surveyed to measure

their comprehension of AI system structures, including understanding of AI model architecture and overall data flow. Response options allowed participants to indicate their familiarity at varying levels, ranging from little to no understanding, theoretical knowledge, to practical hands-on experience. Additionally, participants were asked to evaluate specific processes within the *ThreatFinder* tool, focusing on the simplicity of importing the model and libraries and assigning assets to the model using the diagrams.net tool. Moreover, they had to evaluate the clarity of the provided instructions and the description of the AISym4MED architecture model.

Finally, participants rated ten statements based on the System Usability Scale (SUS). SUS is a performance indicator measuring the long-term usability of a product. SUS serves as a standardized measure for assessing the perceived usability of computer interfaces [147]. It comprises ten questions, incorporating both positive and negative orientations, presented to participants for assessing the *ThreatFinder* tool. Participants rated these questions on a scale from 1 to 5, reflecting their agreement level, with 1 indicating strong disagreement and 5 indicating strong agreement [147]. The questions are as follows:

1. *I think that I would like to use ThreatFinder frequently (assuming that you work in the AI security field).*
2. *I found ThreatFinder unnecessarily complex.*
3. *I thought ThreatFinder was easy to use.*
4. *I think that I would need the support of a technical person to be able to use ThreatFinder.*
5. *I found the various functions in ThreatFinder were well integrated.*
6. *I thought there was too much inconsistency in ThreatFinder.*
7. *I would imagine that most people would learn to use ThreatFinder very quickly.*
8. *I found ThreatFinder very cumbersome to use.*
9. *I felt very confident using ThreatFinder.*
10. *I needed to learn a lot of things before I could get going with ThreatFinder.*

In evaluating the results, the diverse educational backgrounds of the participants (shown in Table 14) must also be considered. Six out of seven participants have a scientific background, with five having a computer science background and two possessing a Master's degree in Data Science. In the forthcoming explanations, reference is made to the participant number, as indicated in Table 14.

Taking into account the gaps identified in Section 3.8, the primary goal of the experiment is to reliably demonstrate the effectiveness of the novel automated threat modeling approach in detecting all important threats within an AI system. Another key objective is to demonstrate the user-friendly nature and inherent straightforwardness of the threat modeling approach. Additionally, it is important to note that the deliberate selection of participants aims to encompass a diverse spectrum of expertise, with the specific intention of identifying suitable participant groups for future evaluations. Despite the relatively small number of participants, the findings serve as a valuable foundation for further research.

Table 14: Participants Educational Background

Participant	Educational Background
1	Master of Data Science
2	Master of Data Science
3	Bachelor of Science (Software Systems)
4	Bachelor of Science (Software Systems)
5	Bachelor of Science (Information Systems)
6	Master of Science ETH in Pharmacy
7	Master of Law

6.2.2 Results of the Experiment

Familiarity with tools and AI-based systems

The results regarding the familiarity with different tools are shown in Figures 22-24: None of the participants reported any familiarity with OVVL. The TMT statistics show similar results, with one participant being *"not familiar"* instead of *"not familiar at all"*. This suggests that the participant may have at least heard the name TMT but possesses no further familiarity. In contrast, diagrams.net is well-known among the majority of participants. However, participants six and seven reported no familiarity with diagrams.net. Furthermore, the participants were surveyed about their familiarity with the structure of AI systems, specifically understanding the architecture of AI models and the overall data flow within AI systems. As shown in Table 15, two participants have practical hands-on experience, two possess theoretical knowledge, and three have no knowledge in this domain.

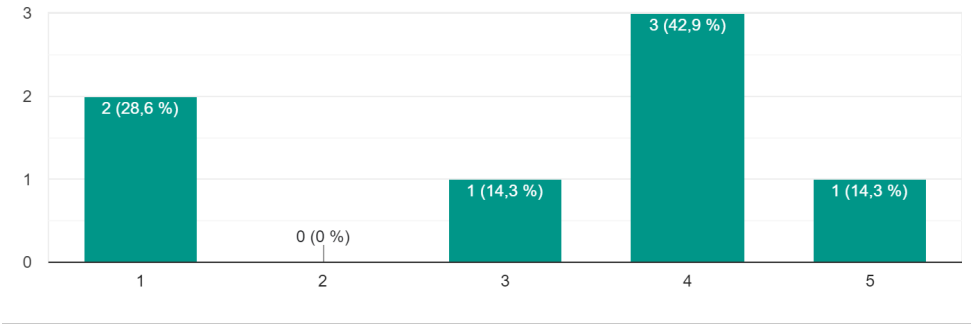

Figure 22: Familiarity with diagrams.net

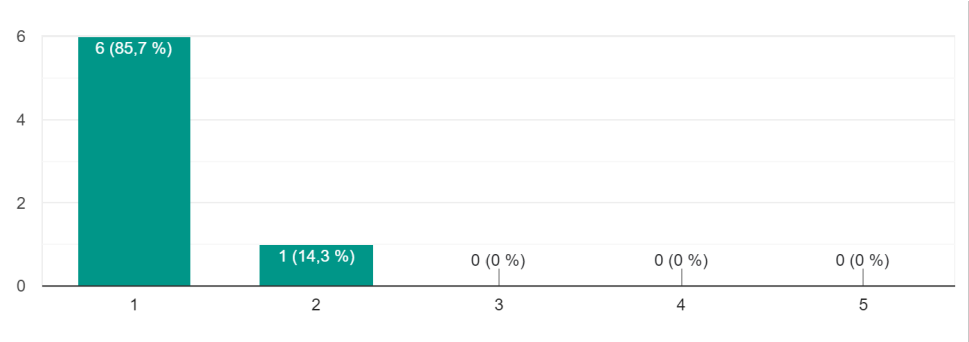

Figure 23: Familiarity with TMT

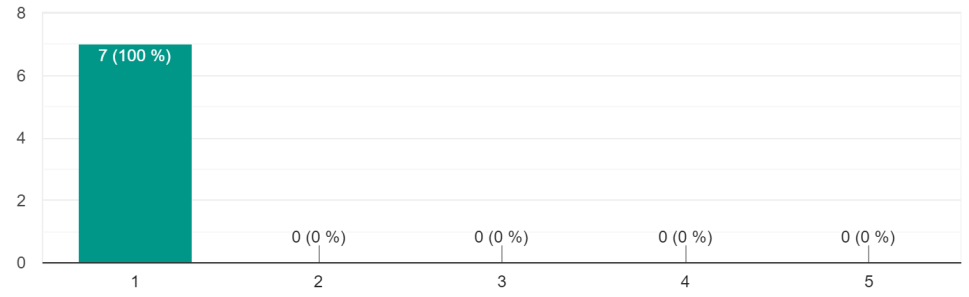

Figure 24: Familiarity with OVVL

Table 15: Participants Educational Background and AI knowledge

Participant	**Educational Background**	**AI Knowledge**
1	Master of Data Science	Practical, hands-on experience
2	Master of Data Science	Practical, hands-on experience
3	Bachelor of Science (Software Systems)	Theoretical knowledge
4	Bachelor of Science (Software Systems)	Theoretical knowledge
5	Bachelor of Science (Information Systems)	Little to no understanding
6	Master of Science ETH in Pharmacy	Little to no understanding
7	Master of Law	Little to no understanding

Experiment Execution

While the import processes were clear to all participants, challenges arose during the assignment of assets. Participant six encountered content-related challenges during the assignment process, according to his own statement acknowledging a limited understanding of the AI model as a layman. Participants one, four, and six faced challenges determining the permissible number of assets to place and whether placing multiple assets was allowed. Additionally, participants one and four encountered difficulty in locating their initially placed assets, as they had added multiple assets per component. Moreover, all participants found the subsequent process of uploading the modified model in the *ThreatFinder* tool to be straightforward.

Furthermore, participants provided feedback on the clarity of instructions and documentation provided in the guide and video for using the tool. Figure 25 indicates that while the majority found the instructions sufficient, there is room for improvement. Evaluating the AISym4MED architecture model description, as outlined in the guidelines and presented in Figure 26, yielded better results. Figure 26 indicates that three out of seven participants found the description to be very clear and easy to understand, while four participants see room for improvement.

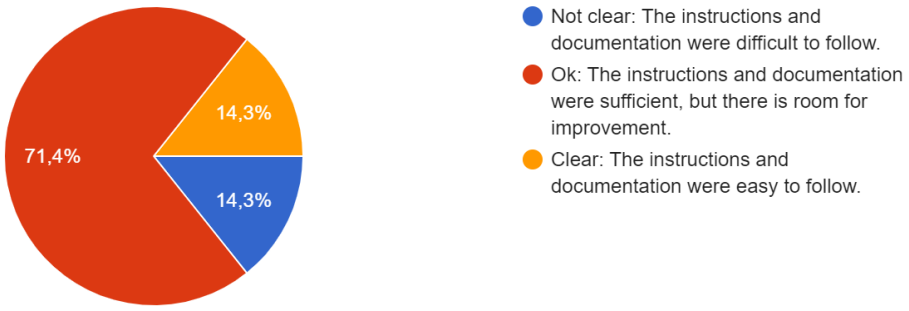

Figure 25: Clarity of Instruction and Documentation

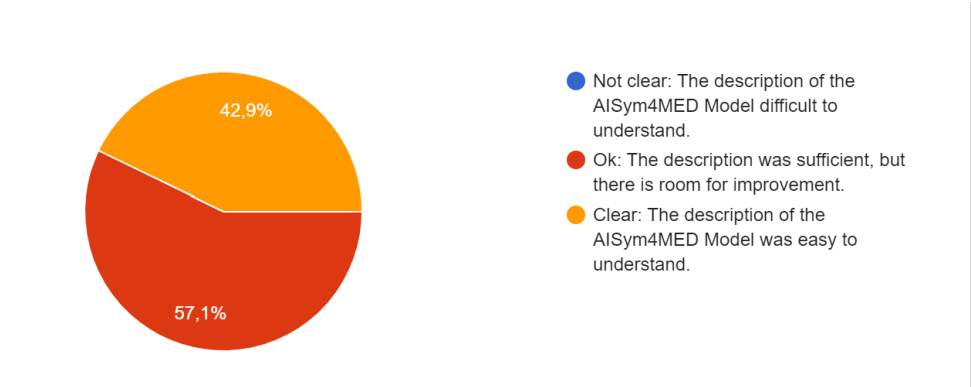

Figure 26: Clarity AISym4MED Architecture Model Description

Selected AI-Properties and System Usability Scale (SUS)

As discussed in Section 6.2.1, the participants had to choose properties they believed were crucial for safeguarding the AISym4MED system's flawless functionality and secure environment. Table 16 displays each participant's chosen AI properties.

Table 16: Selected AI Security Properties

	Authenticity	**Integrity**	**Non-repudiation**	**Confidentiality**	**Availability**	**Authorization**
1	×	✓	×	✓	✓	✓
2	✓	✓	×	✓	×	✓
3	✓	✓	✓	✓	✓	✓
4	×	✓	✓	×	✓	✓
5	×	✓	×	✓	✓	×
6	✓	✓	×	✓	×	✓
7	✓	✓	✓	✓	✓	✓

System Usability Scale (SUS)

Table 17 shows the individual SUS scores and the average of 62.14 across all scores. The SUS score for each participant was calculated using a formula via Excel derived from their responses. The *ThreatFinder* tool's overall SUS Score corresponds to a good D, as illustrated in Figure 27.

Table 17: Participants SUS Score

Participant	**Educational Background**	**SUS 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 ETH in Pharmacy	75
7	Master of Law	45
Average		**62.14**

Grade	SUS	Percentile	Adjective
A+	84.1-100	96-100	Best Imaginable
A	80.8-84.0	90-95	Excellent
A-	78.9-80.7	85-89	
B+	77.2-78.8	80-84	
B	74.1 – 77.1	70 – 79	
B-	72.6 – 74.0	65 – 69	
C+	71.1 – 72.5	60 – 64	Good
C	65.0 – 71.0	41 – 59	
C-	62.7 – 64.9	35 – 40	
D	51.7 – 62.6	15 – 34	OK
F	25.1 – 51.6	2– 14	Poor
F	0-25	0-1.9	Worst Imaginable

Figure 27: SUS Grades [147]

Threats Identification by Experts

The AISym4Med architecture model has undergone expert review, during which the system's associated threats were identified. This assessment will serve as a benchmark against which participants' threats will be compared. The experts have categorized different potential threats across different components and layers of the architecture. Each threat is associated with potential threat actors, such as malicious platform users, external threat actors, infrastructure administrators, and automated external actors like malware. The identified threats encompass a broad range of security concerns. The extensive list of these threats, each categorized by their respective system component, can be found in the Appendix B. The experts identified 44 threats within the whole system, with some threats recurring across multiple components. Although certain threats appear more than once, this thesis does not delve further into the implications of such duplication. Furthermore, it should be noted that the database of the *ThreatFinder* tool, containing 96 distinct threats, encompasses all the threats identified by the experts, and it even extends to more specific threats, thus providing a holistic overview.

6.2.3 Discussion of the Results

Familiarity with tools and AI-based systems

Figures 22 to 24 from Section 6.2.2 can be interpreted as follows: Diagrams.net emerges as the most recognized tool among the three. Participants six and seven, non-computer scientists, exhibit unfamiliarity with diagrams.net, most likely attributed to their educational backgrounds. These results confirm the claim in Section 3.7.4 about diagrams.net's awareness. Therefore, in alignment with the comprehensive evaluation in Table 6 from Section 3.7.4, diagrams.net indeed stands out as the optimal choice for a threat modeling tool. Furthermore, the participants' familiarity with AI aligns with expectations based on their diverse educational backgrounds (see Table 15 in Section 6.2.2). This diversity in AI knowledge results from intentional and selective participant recruitment, aimed at encompassing a broad spectrum of expertise in the field of AI, which has proven successful, as shown in the results presented in Table 15 in Section 6.2.2.

Experiment Execution

To assess the results in Section 6.2.2 related to experiment execution, each individual process is examined separately. Importing libraries and the AISym4MED model into diagrams.net and subsequently uploading the edited model to *ThreatFinder* proved clear and manageable for all participants. However, challenges arose during the assignment of assets to the model, aligning with the third step of the threat modeling approach outlined in Section 4.4 (Table 8). This step involves identifying critical assets and replacing them with equivalent assets. Yet, this necessitates a prior study of the AISym4MED architecture model description.

Figure 26 indicates that all participants considered the architecture model description at least sufficient. While this step serves as an evaluation measure, participants lacking familiarity with AI are not anticipated to perceive the description as clear. For instance, participant six engaged in extensive research and reading to establish a fundamental understanding of the AI domain. Consequently, participants five to seven, characterized by limited or no knowledge in the field of AI, are excluded from the further assessment of the description's clarity. Interestingly, three out of four participants with theoretical or practical knowledge found the model description clear. Participant one, while considering it sufficient, may have preferred alternative formats, such as videos or images. Nevertheless, on the whole, the architecture description can be considered clear. Additionally, it is important to emphasize that this aspect is not a concern in real-world scenarios. The targeted users starting from scratch would not require a description, and in cases where an existing architecture is used, the targeted users typically possess the necessary expertise already.

In contrast, the assessment of instruction and documentation clarity, as depicted in Figure 25 from Section 6.2.2, yielded an overall sufficient rating. Many participants criticized the lack of precise instructions, especially during asset assignment. Questions arose regarding the placement of several assets per component and the consideration of all components in the model. Participants one and four noted challenges with a poor model overview when placing multiple assets per component. Additionally, four participants wanted more context on the task, including the link between threats and assets and the significance of asset placement. It is crucial to interpret this result in context. In real-world scenarios, the intended users are likely to grasp the tasks without requiring detailed instructions. Hence, in these scenarios, extensive guidance is not necessary.

In hindsight, the instructions could have been more precise, such as illustrating an asset assignment in the tutorial using a different AI architecture model. Additionally, the critical components of the architecture could have been annotated with numbers so the participants would have had support. However, there was an initial assumption that participants would intuitively assign one and the most suitable asset per component and consider all components. Despite exceptions like participant one, who assigned all assets from the library, and participant seven, who initially assigned all assets and later modified the assignment, most participants created clear architecture models, not overfilled with assets.

Regarding the participants request for more context in the task, it is understandable but was deliberately not disclosed. As explained in Section 4.4.4, threat identification is not carried out directly via the specific assets but via the corresponding categories. Therefore, the assets' placement plays no role in this experiment. If participants were aware of this fact, they could theoretically place one asset per category anywhere, causing the display of all threats from the database. However, the threat modeling approach for AI-based systems assumes the user's model from scratch, using the libraries. Such a scenario would have been very time-consuming for the participants and would go beyond the scope of this thesis.

Furthermore, it is important to note that, unlike previous studies, this experiment uniquely involves participant interaction, marking a significant step forward in practical, user-centric evaluations of threat modeling in AI-ML systems. This approach not only provides real-world applicability insights but also identifies practical challenges and user perspectives, filling a critical gap in the existing literature.

Selected AI-Properties

This task aimed to assess participants' ability to identify essential properties of an AI system, particularly those requiring protection. Given AISym4MED's focus on healthcare data, participants should have recognized that healthcare data is very private and correspondingly sensitive and cannot be disclosed or uncovered under any circumstances. That is why confidentiality must be guaranteed. Ensuring data integrity throughout the entire AI life cycle is equally critical for maintaining trustworthiness and accuracy in healthcare. Data must remain unchanged and unmanipulated. Hence, confidentiality and integrity are the most important properties that the participants should have identified.

As illustrated in Section 6.2.2 in Table 16, six out of seven participants identified integrity and confidentiality as crucial security properties, suggesting that this recognition is not dependent on educational background. Notably, participant three, with professional experience in IT security, chose all security properties, potentially to emphasize a comprehensive protection of the system. Similarly, participant seven, who has a legal background, advocated for comprehensive protection, which is plausible given his background. While correct, this approach adds complexity to threat prioritization, as safeguarding all properties entails addressing a broader range of potential threats. In contrast, participants two and six cleverly addressed the task by selecting integrity and confidentiality along with authorization and authenticity, recognizing the importance of these properties in the context of the system. Authorization ensures control over AI architecture model use, aligning with legal and ethical data access regulations. On the other hand, authenticity contributes to the credibility and reliability of health data and AISym4MED models.

At this point, it should be noted that the database, aligned with the ENISA report's threat taxonomy, includes only integrity, confidentiality, and availability as potential impact properties.

Consequently, the selection of authenticity, authorization, and non-repudiation by participants did not affect threat filtering. While aware of this, participants were intentionally presented with all properties to observe their choices. However, according to the ENISA report[11], the properties are interconnected. For instance, authenticity may be compromised with integrity issues. Authorization may be impacted when both confidentiality and integrity are affected. Moreover, non-repudiation may be impacted by integrity concerns. This further emphasizes that the selection of confidentiality and integrity effectively encompasses all crucial aspects, considering the interconnected nature of properties.

Looking ahead, there is room for improvement in the assignment of properties to threats within the threat taxonomy. Future work could involve clearly assigning all relevant properties to specific threats, thereby improving the property filtering process. Additionally, further properties mentioned in the ENISA report[11] could be considered, such as robustness, trustworthiness, safety, transparency, explainability, accountability, and data protection. However, introducing these additional properties requires expert involvement and could lead to overlooking certain relevant threats due to overly stringent filtering.

System Usability

Table 17 shows an average SUS score of 62.14, which corresponds to a D. This grade indicates room for improvement, as shown in Figure 27 in Section 6.2.2. Upon closer examination, the results reveal nuances that mitigate the initial perception of sufficient performance. Further insights are provided by discussing individual data points below.

Examining participant six, who lacks knowledge of AI or computer science and is unfamiliar with diagrams.net, reveals the lowest ratings in questions 4, 7, 9, and 10, along with the lowest individual SUS score. Hence, his score significantly lowers the overall average. In contrast, participant six, lacking AI expertise but with a scientific background, produced a high SUS score of 75, particularly positively rating questions 4, 7, 9, and 10. Notably, this participant invested substantial time in understanding the architecture model and did some research on the AI life cycle. If we consider only participants one to six, who have a scientific background, for the SUS score calculation, the average score rises to 65. This significantly alters the outcome, shifting it up by two grades as indicated in Figure 27. These findings suggest that *ThreatFinder* might be challenging for users lacking a scientific background, such as participant seven. However, individuals lacking a computer science background but possessing a scientific one, such as participant six, can excel through dedicated effort.

Closely examining participants one to three is intriguing, given their substantial expertise. Participants one and two demonstrate proficiency in AI, and participant three stands out as the sole participant with sound expertise in IT security. Participant two's ratings resulted in a good SUS score, participant three's in an excellent one, while participant one's responses resulted in a sufficient score. Notably, there are specific observations about participant one's answers: For question 4, regarding the need for technical support, participant one responded with "agree", reflecting a negative sentiment. Interestingly, participant seven, with distinct prerequisites, shared a similar response. In contrast, all other participants answered negatively, indicating a positive score for this question (see Figure 28). However, this response contradicts his answer to question 7, where he strongly agrees that people would learn the tool quickly. Furthermore, participant one gave negative responses to questions 5, 8, and 9, primarily related to the use of diagrams.net. Notably, his written feedback highlighted an apparent dislike for diagrams.net within *ThreatFinder* and that it also led to confusion for him. However, he positively

acknowledged the user interface of *ThreatFinder*. Considering these circumstances, adjusting for inconsistencies, his overall score would likely fall around the ”good” range for the *ThreatFinder*.

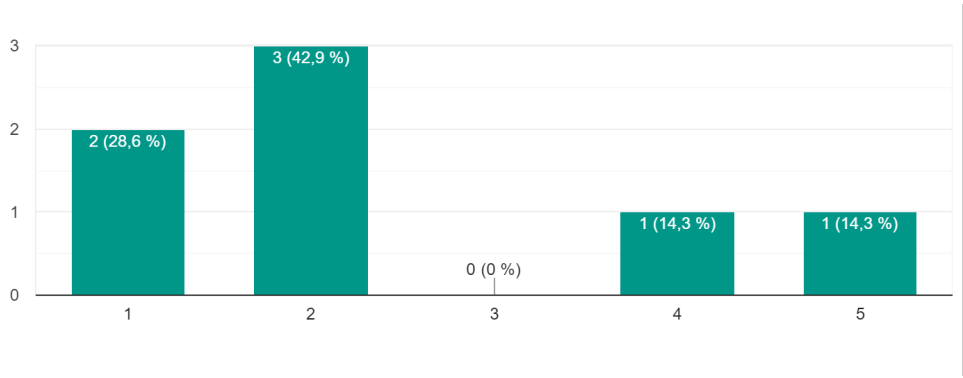

Figure 28: Support of a Technical Person needed

Analyzing participant five’s responses proves challenging, given the consistently neutral stance across all ten questions. This could stem from a lack of prior knowledge in the domain and being the sole participant perceiving the instructions as unclear (see Figure 25 in Section 6.2.2). As previously discussed, in a real-world context, considering the targeted user base, such a scenario is unlikely, as extensive instructions may not be necessary. Moreover, participant five stands out as the sole contributor without feedback, and his modified architecture model is notably minimal, with the fewest placed assets. Hence, the outcomes of participant five, like participant seven, should be treated differently, not being accorded equal weight in comparison to other results. This also applies to participant four, whose responses revealed several inconsistencies. While questions 2-4 indicate a perception of *ThreatFinder* as easy to use without requiring technical assistance, questions 8 and 9 suggest a contrasting view, expressing difficulty and a lack of confidence in using the tool.

Considering all factors, the effective average SUS score is expected to be higher than 62.14. Participants two, three and six consistently demonstrate positive scores without inconsistencies. However, participants five and seven, significantly impact the average, justifying cautious interpretation due to their limited knowledge in this field and no obvious efforts made. Furthermore, participants one and four introduce complexities with inconsistencies, necessitating nuanced consideration. Considering each participant’s unique situation is crucial for a more precise evaluation of *ThreatFinder*’s usability. However, considering only participants with a scientific background or AI expertise, the SUS score is notably higher, reflecting a ”good” rating. This outcome suggests that the implementation of this approach effectively addresses the literature’s identified need for a practical and effective threat modeling method for AI-ML systems.

Threats Identification by Participants

Evaluating the threats identification of the participants presented challenges. The specific asset selection was not crucial, as the threats are associated with categories. Thus, choosing assets across all categories without any filter would cover all the threats of the database, including the relevant 44 threats identified by the experts. Table 18 shows that five out of seven participants utilized assets from all six categories. Further analysis is required to determine if these five participants also chose suitable properties to identify all relevant threats. Based on this assessment, only participants one, two, three, and six successfully identified all relevant threats. Hence, the approach's effectiveness is affirmed, with four out of seven participants identifying all 44 threats. Participants one through three, who possess the most expertise, effectively identified all threats, highlighting the advantage of AI knowledge in applying the threat modeling approach. Notably, participant six, a layperson, also achieved this. However, participant four missed crucial filters, and participant seven neglected to use assets from the actor category.

With the effectiveness established, it is intriguing to examine the efficiency of participants who identified all relevant threats. This can be done by comparing the number of identified threats to that of the experts. Table 19 presents the total threats identified by participants. Participant one's results are not indicative due to the use of all available assets, resulting in an inflated threat count. Similarly, Participant six's data is excluded from further consideration; the initial use of all assets, modified later, suggests the high threat identification might be coincidental, especially when considering previous responses and educational background. Upon examining the results of participants two and three, who conducted the experiment correctly and possess the required expertise, a significant number of false positives are still apparent (see Table 19). Although the experts, with their identification of 44 threats, may not have captured every threat relevant to the system, the discrepancy between their count and the number of threats identified by these two participants is notably large. Therefore, this approach comes with a trade-off, generating numerous false positives and noise in the process. Identifying a larger number of threats can be beneficial for a thorough analysis of a system. However, the present number of false positives creates excessive work, making an increase in efficiency indispensable.

Overall, the participants successfully identified all potential threats in the AI system using the architectural model, demonstrating the approach's effectiveness. The applied approach addresses the difficulties noted in Section 3.8 by automating a straightforward threat identification process in ML systems, enhancing scalability and adaptability. This marks a significant improvement, effectively addressing the previously identified gaps. However, there remains room for improvement in efficiency, as the threats could be further filtered. The six existing categories could be further divided, considering that threats are tied to specific categories. Such subdivision would lead to a more focused association, with each category encompassing a smaller, more manageable number of threats. Consequently, this would naturally result in a reduction of noise. Furthermore, exploring participants' prioritization of threats and the vulnerabilities they might have uncovered would be very insightful. Undertaking such a task was beyond the scope of this study. However, it presents an opportunity for exploration in future work.

Table 18: Used Categories by Participant

Participant	Data	Model	Artefact	Processes	Env/tools	Actor
1	✓	✓	✓	✓	✓	✓
2	✓	✓	✓	✓	✓	✓
3	✓	✓	✓	✓	✓	✓
4	✓	✓	✓	✓	✓	✓
5	×	✓	×	✓	✓	×
6	✓	✓	✓	✓	✓	✓
7	✓	✓	✓	✓	✓	×

Table 19: Comparison of Threats Participants and Experts

Participant	Threats Identified	Noise
1	1668	1624
2	901	857
3	296	252
6	1232	1188

6.3 Reflections and Future Work

The evaluation of the *ThreatFinder* tool emphasizes its effectiveness as the first-ever automated threat modeling approach for AI systems. It confirms the strengths of the novel approach that addresses key challenges observed in existing threat modeling methodologies for AI-based systems. In particular, the approach exceeds the limitations found in works such as [10] and [9]. Unlike these papers, this approach employs a comprehensive threat taxonomy, ensuring broader coverage of potential threats specific to ML systems. Moreover, the mentioned papers rely on complex manual mapping processes, hindering scalability and adaptability. In contrast, the proposed new approach leverages automation in threat modeling for AI-based systems. This automated approach improves the identification and mitigation of security challenges, ensuring a more efficient and error-resistant analysis in the dynamic landscape of ML systems. In addition, the inclusion of participant feedback not only evaluates usability but also provides insights into the real-world applicability of the approach. This sets it apart from the evaluations in the referenced papers. Furthermore, the novel approach addresses the concerns raised in the study [136] by focusing on a general threat modeling framework. It provides a threat modeling solution applicable to diverse AI system contexts. Overall, the evaluation confirms that a novel, automated, and user-centric threat modeling approach has been introduced that effectively addresses the dynamic nature of AI systems.

Participants successfully navigated the processes, demonstrating the tool's clarity in importing the architecture model and libraries. Difficulties arose during the assignment of the assets, revealing areas for improvement in instructions. However, it is important to highlight that the genuine threat modeling approach demands the careful design of the architectural model from its inception. Consequently, the assignment of assets would not be applicable within this context. Additionally, this observation brings to light the added complexity and potential challenges associated with using an existing architecture model in the context of threat modeling.

Employing and comprehending an established architecture model may be more difficult and cumbersome. In contrast, there is a compelling case for constructing the architecture model from scratch and directly implementing assets with corresponding threats. This approach offers greater clarity and aligns more seamlessly with the demands of effective threat modeling for AI systems.

The overall SUS score of 62.14 showed a satisfactory level of user-friendliness. However, a differentiated analysis showed that the effective score yields are higher. For instance, considering the three participants with the highest expertise and accounting for individual response inconsistencies, the overall evaluation tends towards the "Good" range. In a nutshell, the simple and effective approach lays a crucial foundation for future work, providing valuable insights for the advancement of the first-ever automated threat modeling approach for AI systems. However, it is important to note that the findings from this evaluation should be approached with caution. The experiment involved only seven participants, which may not fully represent the broader user base. This limitation suggests that while the results are promising, they might not be entirely representative of all potential users. Future studies, therefore, should include a larger and more targeted participant pool to validate and refine the approach further, ensuring its applicability and robustness in various real-world scenarios.

Furthermore, as previously discussed, the original threat modeling approach intended users to model the system from scratch, which would have been beyond the scope of this thesis. Such an endeavor would necessitate AI architects with expertise to conduct the modeling, leveraging their familiarity with the subject matter. In future work, a potential direction could be to guide users in modeling the system based solely on a provided description. The guidance could be exclusively presented in the form of a video tutorial featuring a simple architecture model from scratch as an example to ensure clarity and avoid any potential ambiguities in execution. Moreover, the evaluation highlights that individuals without scientific expertise and a willingness to invest effort may not be the intended users. Hence, subsequent experiments should exclusively target individuals with expertise in the field of AI for meaningful insights.

The most crucial point concerning possible improvements to this new approach was the filtering process. To enhance threat filtering and focus on the most important threats, further subdivision of asset categories and the comprehensive extension of AI properties in the database could be considered for a more efficient result. This entails modifying asset and threat taxonomies for a seamless connection (see Figure 11 in Section 4.4.4). While potentially increasing efficiency with fewer assets per category, this approach requires significant effort and expertise, and there is a risk of overlooking threats. Additionally, expanding libraries to encompass a comprehensive number of assets for modeling AI systems stands as a promising avenue for enriching the tool's capabilities and accommodating diverse modeling scenarios. Additionally, future research could explore methods for prioritizing threats and identifying related vulnerabilities within this threat modeling approach.

Chapter 7

Summary and Conclusions

The main objective of the thesis was to design and implement an effective threat modeling approach for AI-based systems. To achieve this, the thesis evolved through a series of carefully structured chapters, each contributing to this approach's overall discussion and development. The first chapter laid the groundwork for the thesis, introducing the core topic and outlining the study's motivation, objectives, and structure. In Chapter 2, a solid theoretical foundation was established for understanding threat modeling in AI and ML systems. This chapter underscored the importance of recognizing and mitigating vulnerabilities and threats within these systems. Doing so set the stage for a deeper comprehension of the key concepts underpinning a threat modeling approach specifically tailored for AI-based systems.

Chapter 3 thoroughly examined the current state of research and methodologies in threat modeling within AI-based systems. This chapter was rich in content, covering related topics such as AML, cyberattacks in the ML pipeline, FMEA, and various types of failures in ML systems. Furthermore, it examined various frameworks, focusing on the ENISA report. Moreover, and most importantly, the chapter provided an insightful review of existing threat modeling approaches for AI-based systems, highlighting their limitations. Additionally, it included a comparison of existing threat modeling tools and evaluated their applicability to AI-based systems. This evaluation identified diagrams.net as the most suitable tool for modeling in this context. This extensive research highlights a significant gap in the field. In particular, the lack of a systematic and well-established threat modeling approach for AI-based systems. This insight laid the foundation for developing a novel, practical and reliable threat modeling approach.

Chapter 4 focused on the architecture of AI and ML systems, framing an effective threat modeling approach. A general five-step threat modeling methodology was introduced and meticulously adapted for AI-based systems. Each step was thoughtfully tailored to align with the specific properties and dynamic nature of AI systems, with particular emphasis on the novel and straightforward process of threat identification.

Chapter 5 marked the transition from the theoretical approach to its practical application, detailing the development of *ThreatFinder*. *ThreatFinder* is an automated tool designed for threat identification in AI systems. This chapter illustrated the creation and integration of the tool using different technologies. It covered the use of diagrams.net, the employment of XML files, and delved into the functionality and structure of the tool, including the aspects of its web application development.

The thesis final chapter presented an evaluation of the *ThreatFinder* tool. The evaluation utilized the AISym4MED platform as a practical use case. Furthermore, this final chapter gained depth from contributions by participants from diverse educational backgrounds, whose interactions with the tool and subsequent feedback were meticulously analyzed. This analysis provided a detailed assessment of the tool's effectiveness and user-friendliness, enriching the thesis with practical insights.

In conclusion, the *ThreatFinder* tool is a groundbreaking, automated threat modeling approach tailored for AI-based systems. The tool's design overcomes the constraints observed in existing methodologies by integrating a comprehensive threat taxonomy, a straightforward approach, and automated processes. Thus, it offers a more thorough and effective threat analysis for ML systems. The evaluation results, derived from participant feedback, underscore the tool's effectiveness in revealing critical threats and confirming usability and practical relevance. This novel approach not only optimizes the threat modeling process but also addresses general framework concerns, positioning it as an adaptable solution for various AI system contexts. Furthermore, the thesis identifies potential improvement areas, such as enhancing the threat filtering mechanism through refined asset categorization and expanding AI property coverage in the database. Moreover, future work could include the development of more intuitive modeling guidance, especially for users less familiar with the AI field. However, it is recommended for future evaluations to focus on participants with AI expertise for more in-depth assessment. This would allow for modeling from scratch, ensuring a more profound engagement with the proposed threat modeling approach and generating more insightful outcomes. Nevertheless, this thesis contributes significantly to AI security, offering a reliable foundation for advancing threat modeling in AI-based systems.

Bibliography

[1] D. Silver *et al.* "Alphazero: Google deepmind ai beats champion program by teaching itself to play". Accessed: 2023-12-12. (2017), [Online]. Available: `https://www.theguardian.com/technology/2017/dec/07/alphazero-google-deepmind-ai-beats-champion-program-teaching-itself-to-play-four-hours`.

[2] L. F. Sikos, *AI in Cybersecurity*. Springer, 2018, vol. 151.

[3] Ö. A. Aslan and R. Samet, "A comprehensive review on malware detection approaches", *IEEE access*, vol. 8, pp. 6249–6271, 2020.

[4] IBM. "Ai and cybersecurity: A new paradigm". Accessed: 2023-12-12, IBM Institute for Business Value. (2022), [Online]. Available: `https://www.ibm.com/thought-leadership/institute-business-value/en-us/report/ai-cybersecurity`.

[5] D. Arp, E. Quiring, F. Pendlebury, *et al.*, "Dos and don'ts of machine learning in computer security", in *31st USENIX Security Symposium (USENIX Security 22)*, 2022.

[6] L. Mauri and E. Damiani, "Stride-ai: An approach to identifying vulnerabilities of machine learning assets", in *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, IEEE, 2021, pp. 147–154.

[7] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial machine learning", in *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, 2011, pp. 43–58.

[8] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction {apis}", in *25th USENIX security symposium (USENIX Security 16)*, 2016, pp. 601–618.

[9] L. Mauri and E. Damiani, "Modeling threats to ai-ml systems using stride", *Sensors*, vol. 22, no. 17, p. 6662, 2022.

[10] C. Wilhjelm and A. A. Younis, "A threat analysis methodology for security requirements elicitation in machine learning based systems", in *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, IEEE, 2020, pp. 426–433.

[11] European Union Agency for Cybersecurity (ENISA), *Ai cybersecurity challenges, threat landscape for artificial intelligence*, 2020.

[12] I. Tarandach and M. J. Coles, "Threat modeling: A practical guide for development teams", *(No Title)*, 2021.

[13] P. Shedden, W. Smith, and A. Ahmad, "Information security risk assessment: Towards a business practice perspective", 2010.

[14] E. Bertino, L. Martino, F. Paci, and A. Squicciarini, *Security for web services and service-oriented architectures*. Springer, 2010, vol. 4.

[15] J. M. Kizza, W. Kizza, and Wheeler, *Guide to computer network security*. Springer, 2013, vol. 8.

[16] H. G. Brauch, "Concepts of security threats, challenges, vulnerabilities and risks", *Coping with global environmental change, disasters and security: Threats, challenges, vulnerabilities and risks*, pp. 61–106, 2011.

[17] K. Dahbur, B. Mohammad, and A. B. Tarakji, "A survey of risks, threats and vulnerabilities in cloud computing", in *Proceedings of the 2011 International conference on intelligent semantic Web-services and applications*, 2011, pp. 1–6.

[18] R. Raniner and C. Cegielski, *Introduction to information systems: Enabling and transforming business*, 2010.

[19] M. Abomhara and G. M. Køien, "Cyber security and the internet of things: Vulnerabilities, threats, intruders and attacks", *Journal of Cyber Security and Mobility*, pp. 65–88, 2015.

[20] A. J. Duncan, S. Creese, and M. Goldsmith, "Insider attacks in cloud computing", in *2012 IEEE 11th international conference on trust, security and privacy in computing and communications*, IEEE, 2012, pp. 857–862.

[21] P. Baybutt, "Assessing risks from threats to process plants: Threat and vulnerability analysis", *Process Safety Progress*, vol. 21, no. 4, pp. 269–275, 2002.

[22] C. Tankard, "Advanced persistent threats and how to monitor and deter them", *Network security*, vol. 2011, no. 8, pp. 16–19, 2011.

[23] F. Li, A. Lai, and D. Ddl, "Evidence of advanced persistent threat: A case study of malware for political espionage", in *2011 6th International Conference on Malicious and Unwanted Software*, IEEE, 2011, pp. 102–109.

[24] NIST. "Guide for conducting risk assessments". Accessed: 2023-09-05. (2012), [Online]. Available: `https://doi.org/10.6028/NIST.SP.800-30r1`.

[25] TechTarget. "Risk assessment vs. threat modeling: What's the difference?" Accessed on: 2023-12-15. (2023), [Online]. Available: `https://www.techtarget.com/searchsecurity/tip/Risk-assessment-vs-threat-modeling-Whats-the-difference`.

[26] J. Von Der Assen, M. F. Franco, C. Killer, E. J. Scheid, and B. Stiller, "Coretm: An approach enabling cross-functional collaborative threat modeling", in *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*, IEEE, 2022, pp. 189–196.

[27] S. Hussain, A. Kamal, S. Ahmad, G. Rasool, and S. Iqbal, "Threat modelling methodologies: A survey", *Sci. Int.(Lahore)*, vol. 26, no. 4, pp. 1607–1609, 2014.

[28] S. Myagmar, A. J. Lee, and W. Yurcik, "Threat modeling as a basis for security requirements", 2005.

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

[30] H. A. Alatwi and C. Morisset, "Threat modeling for machine learning-based network intrusion detection systems", in *2022 IEEE International Conference on Big Data (Big Data)*, IEEE, 2022, pp. 4226–4235.

[31] R. Hasan, S. Myagmar, A. J. Lee, and W. Yurcik, "Toward a threat model for storage systems", in *Proceedings of the 2005 ACM workshop on Storage security and survivability*, 2005, pp. 94–102.

[32] B. Lundgren and N. Möller, "Defining information security", *Science and engineering ethics*, vol. 25, pp. 419–441, 2019.

[33] M. Conti, N. Dragoni, and V. Lesyk, "A survey of man in the middle attacks", *IEEE communications surveys & tutorials*, vol. 18, no. 3, pp. 2027–2051, 2016.

[34] N. Shevchenko, T. A. Chick, P. O'Riordan, T. P. Scanlon, and C. Woody, "Threat modeling: A summary of available methods", Carnegie Mellon University Software Engineering Institute Pittsburgh United ..., Tech. Rep., 2018.

[35] R. Khan, K. McLaughlin, D. Laverty, and S. Sezer, "Stride-based threat modeling for cyber-physical systems", in *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, IEEE, 2017, pp. 1–6.

[36] A. Y. Aleryani, "Comparative study between data flow diagram and use case diagram", *International Journal of Scientific and Research Publications*, vol. 6, no. 3, pp. 124–126, 2016.

[37] A. A. Jilani, M. Usman, and A. Nadeem, "Comparative study on dfd to uml diagrams transformations", *arXiv preprint arXiv:1102.4162*, 2011.

[38] A. Dennis, B. Wixom, and D. Tegarden, *Systems analysis and design: An object-oriented approach with UML*. John wiley & sons, 2015.

[39] P. Rob, C. Coronel, A. Silberschatz, H. Korth, and S. Sudarshan, "Database systems: Design, implementation", *Management. Seventh Edition. Course Technology*, 2006.

[40] S. Hernan, S. Lambert, T. Ostwald, and A. Shostack, "Threat modeling-uncover security design flaws using the stride approach", *MSDN Magazine-Louisville*, pp. 68–75, 2006.

[41] N. R. Mead, F. Shull, K. Vemuru, and O. Villadsen, "A hybrid threat modeling method", *Carnegie MellonUniversity-Software Engineering Institute-Technical Report-CMU/SEI-2018-TN-002*, 2018.

[42] L. Sion, K. Yskout, D. Van Landuyt, and W. Joosen, "Solution-aware data flow diagrams for security threat modeling", in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018, pp. 1425–1432.

[43] R. Scandariato, K. Wuyts, and W. Joosen, "A descriptive study of microsoft's threat modeling technique", *Requirements Engineering*, vol. 20, pp. 163–180, 2015.

[44] B. Potteiger, G. Martins, and X. Koutsoukos, "Software and attack centric integrated threat modeling for quantitative risk assessment", in *Proceedings of the Symposium and Bootcamp on the Science of Security*, 2016, pp. 99–108.

[45] T. UcedaVelez, "Real world threat modeling using the pasta methodology", *OWASP App Sec EU*, 2012.

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

[47] T. UcedaVélez, "Threat modeling w/pasta: Risk centric threat modeling case studies", Technical Report. Open Web Application Security Project (OWASP), Tech. Rep., 2017.

[48] ThreatModeler, *Threat modeling methodology*, Accessed: 2023-07-29, 2023. [Online]. Available: `https://threatmodeler.com/threat-modeling-methodology/`.

[49] T. UcedaVelez, *Real world threat modeling using the pasta methodology*, Accessed: 2023-07-29, 2012. [Online]. Available: `https://owasp.org/www-pdf-archive/AppSecEU2012_PASTA.pdf`.

[50] S. Intelligence, *Threat modeling in the enterprise, part 2: Understanding the process*, Accessed: 2023-07-29, 2023. [Online]. Available: `https://securityintelligence.com/threat-modeling-in-the-enterprise-part-2-understanding-the-process/`.

[51] V. Saini, Q. Duan, and V. Paruchuri, "Threat modeling using attack trees", *Journal of Computing Sciences in Colleges*, vol. 23, no. 4, pp. 124–131, 2008.

[52] G. Anthes, "Artificial intelligence poised to ride a new wave", *Communications of the ACM*, vol. 60, no. 7, pp. 19–21, 2017.

[53] I. H. Sarker, "Ai-based modeling: Techniques, applications and research issues towards automation, intelligent and smart systems", *SN Computer Science*, vol. 3, no. 2, p. 158, 2022.

[54] L. Deng, "Artificial intelligence in the rising wave of deep learning: The historical path and future outlook [perspectives]", *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 180–177, 2018.

[55] S. Amershi, A. Begel, C. Bird, *et al.*, "Software engineering for machine learning: A case study", in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, IEEE, 2019, pp. 291–300.

[56] J. Bosch, H. H. Olsson, and I. Crnkovic, "Engineering ai systems: A research agenda", *Artificial Intelligence Paradigms for Smart Cyber-Physical Systems*, pp. 1–19, 2021.

[57] S. Martınez-Fernández, J. Bogner, X. Franch, *et al.*, "Software engineering for ai-based systems: A survey", *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 2, pp. 1–59, 2022.

[58] H. AI, *High-level expert group on artificial intelligence*, 2019.

[59] F. Khomh, B. Adams, J. Cheng, M. Fokaefs, and G. Antoniol, "Software engineering for machine-learning applications: The road ahead", *IEEE Software*, vol. 35, no. 5, pp. 81–84, 2018.

[60] I. Ozkaya, "What is really different in engineering ai-enabled systems?", *IEEE software*, vol. 37, no. 4, pp. 3–6, 2020.

[61] C. Kästner and E. Kang, "Teaching software engineering for ai-enabled systems", in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training*, 2020, pp. 45–48.

[62] D. Sculley, G. Holt, D. Golovin, *et al.*, "Hidden technical debt in machine learning systems", *Advances in neural information processing systems*, vol. 28, 2015.

[63] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions", *SN computer science*, vol. 2, no. 3, p. 160, 2021.

[64] S. Dua and X. Du, *Data mining and machine learning in cybersecurity*. CRC press, 2016.

[65] J. Blumenstock, "Machine learning can help get covid-19 aid to those who need it most", *Nature*, 2020.

[66] I. H. Sarker, "Cyberlearning: Effectiveness analysis of machine learning security modeling to detect cyber-anomalies and multi-attacks", *Internet of Things*, vol. 14, p. 100 393, 2021.

[67] I. H. Sarker, A. Kayes, S. Badsha, H. Alqahtani, P. Watters, and A. Ng, "Cybersecurity data science: An overview from machine learning perspective", *Journal of Big data*, vol. 7, pp. 1–29, 2020.

[68] S. Saharan, N. Kumar, and S. Bawa, "An efficient smart parking pricing system for smart city environment: A machine-learning based approach", *Future Generation Computer Systems*, vol. 106, pp. 622–640, 2020.

[69] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 427–436.

[70] G. Elsayed, S. Shankar, B. Cheung, *et al.*, "Adversarial examples that fool both computer vision and time-limited humans", *Advances in neural information processing systems*, vol. 31, 2018.

[71] Q. Wang, W. Guo, K. Zhang, *et al.*, "Adversary resistant deep neural networks with an application to malware detection", in *Proceedings of the 23rd ACM sigkdd international conference on knowledge discovery and data mining*, 2017, pp. 1145–1153.

[72] I. Rosenberg, A. Shabtai, Y. Elovici, and L. Rokach, "Adversarial machine learning attacks and defense methods in the cyber security domain", *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–36, 2021.

[73] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?", in *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, 2006, pp. 16–25.

[74] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning", *Machine Learning*, vol. 81, pp. 121–148, 2010.

[75] B. Schneier, "Attacking machine learning systems", *Computer*, vol. 53, no. 5, pp. 78–80, 2020.

[76] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning", in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 2154–2156.

[77] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman, "Sok: Security and privacy in machine learning", in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, 2018, pp. 399–414.

[78] N. Papernot, "A marauder's map of security and privacy in machine learning", *arXiv preprint arXiv:1811.01134*, 2018.

[79] L. Muñoz-González and E. C. Lupu, "The security of machine learning systems", *AI in Cybersecurity*, pp. 47–79, 2019.

[80] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning", *IEEE transactions on neural networks and learning systems*, vol. 30, no. 9, pp. 2805–2824, 2019.

[81] E. Tabassi, K. J. Burns, M. Hadjimichael, A. D. Molina-Markham, and J. T. Sexton, "A taxonomy and terminology of adversarial machine learning", *NIST IR*, vol. 2019, pp. 1–29, 2019.

[82] International Organization for Standardization, "ISO/IEC TR 24028:2020; Information Technology—Artificial Intelligence—Overview of Trustworthiness in Artificial Intelligence", ISO, Geneva, Switzerland, Tech. Rep., 2020.

[83] H. MAMUN and H. MOLYNEAUX, "Towards a robust and trustworthy machine learning system development", *arXiv preprint arXiv:2101.03042*, 2021.

[84] K. Patel, J. Fogarty, J. A. Landay, and B. Harrison, "Investigating statistical machine learning as a tool for software development", in *Proceedings of the SIGCHI conference on human factors in computing systems*, 2008, pp. 667–676.

[85] J. Horkoff, "Non-functional requirements for machine learning: Challenges and new directions", in *2019 IEEE 27th international requirements engineering conference (RE)*, IEEE, 2019, pp. 386–391.

[86] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks", in *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*, Springer, 2017, pp. 3–29.

[87] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples", *arXiv preprint arXiv:1801.09344*, 2018.

[88] 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.

[89] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey", *arXiv preprint arXiv:1810.00069*, 2018.

[90] K. Fenrich, "Securing your control system", in *50th Annual ISA. POWID Symposium/17th ISA POWID/EPRI Controls & Instrumentation Conference*, 2007, p. 11.

[91] M. Comiter, "Attacking artificial intelligence", *Belfer Center Paper*, vol. 8, pp. 2019–08, 2019.

[92] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data", in *Artificial intelligence and statistics*, PMLR, 2017, pp. 1273–1282.

[93] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning", in *International conference on artificial intelligence and statistics*, PMLR, 2020, pp. 2938–2948.

[94] L. Muñoz-González, B. Biggio, A. Demontis, *et al.*, "Towards poisoning of deep learning algorithms with back-gradient optimization", in *Proceedings of the 10th ACM workshop on artificial intelligence and security*, 2017, pp. 27–38.

[95] B. Nelson, M. Barreno, F. J. Chi, *et al.*, "Exploiting machine learning to subvert your spam filter.", *LEET*, vol. 8, no. 1-9, pp. 16–17, 2008.

[96] Ilmoi. "Poisoning attacks on machine learning: A 15-year old security problem that's making a comeback". Accessed: 2023-08-22. (2019), [Online]. Available: `https://towardsdatascience.com/poisoning-attacks-on-machine-learning-1ff247c254db`.

[97] J. Natarajan, "Cyber secure man-in-the-middle attack intrusion detection using machine learning algorithms", in *AI and Big Data's Potential for Disruptive Innovation*, IGI global, 2020, pp. 291–316.

[98] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain", *arXiv preprint arXiv:1708.06733*, 2017.

[99] Z. Yang, J. Zhang, E.-C. Chang, and Z. Liang, "Neural network inversion in adversarial setting via background knowledge alignment", in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 225–240.

[100] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures", in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.

[101] X. Wu, M. Fredrikson, S. Jha, and J. F. Naughton, "A methodology for formalizing model-inversion attacks", in *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, IEEE, 2016, pp. 355–370.

[102] S. Hidano, T. Murakami, S. Katsumata, S. Kiyomoto, and G. Hanaoka, "Model inversion attacks for prediction systems: Without knowledge of non-sensitive attributes", in *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, IEEE, 2017, pp. 115–11 509.

[103] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: Information leakage from collaborative deep learning", in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 603–618.

[104] R. N. Reith, T. Schneider, and O. Tkachenko, "Efficiently stealing your machine learning models", in *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society*, 2019, pp. 198–210.

[105] M. Juuti, S. Szyller, S. Marchal, and N. Asokan, "Prada: Protecting against dnn model stealing attacks", in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, 2019, pp. 512–527.

[106] B. Wang and N. Z. Gong, "Stealing hyperparameters in machine learning", in *2018 IEEE symposium on security and privacy (SP)*, IEEE, 2018, pp. 36–52.

[107] T. Takemura, N. Yanai, and T. Fujiwara, "Model extraction attacks on recurrent neural networks", *Journal of Information Processing*, vol. 28, pp. 1010–1024, 2020.

[108] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network", *arXiv preprint arXiv:1503.02531*, 2015.

[109] Y.-C. Hsu, T. Hua, S. Chang, Q. Lou, Y. Shen, and H. Jin, "Language model compression with weighted low-rank factorization", *arXiv preprint arXiv:2207.00112*, 2022.

[110] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models", in *2017 IEEE symposium on security and privacy (SP)*, IEEE, 2017, pp. 3–18.

[111] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An {end-to-end} case study of personalized warfarin dosing", in *23rd USENIX security symposium (USENIX Security 14)*, 2014, pp. 17–32.

[112] A. Chaabane, G. Acs, M. A. Kaafar, *et al.*, "You are what you like! information leakage through users' interests", in *Proceedings of the 19th annual network & distributed system security symposium (NDSS)*, Citeseer, 2012.

[113] N. Z. Gong, A. Talwalkar, L. Mackey, *et al.*, "Joint link prediction and attribute inference using a social-attribute network", *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 2, pp. 1–20, 2014.

[114] M. Kosinski, D. Stillwell, and T. Graepel, "Private traits and attributes are predictable from digital records of human behavior", *Proceedings of the national academy of sciences*, vol. 110, no. 15, pp. 5802–5805, 2013.

[115] C. Spreafico, D. Russo, and C. Rizzi, "A state-of-the-art review of fmea/fmeca including patents", *computer science review*, vol. 25, pp. 19–28, 2017.

[116] N. R. Sankar and B. S. Prabhu, "Modified approach for prioritization of failures in a system failure mode and effects analysis", *International Journal of Quality & Reliability Management*, vol. 18, no. 3, pp. 324–336, 2001.

[117] C. Kara-Zaitri, "Disaster prevention and limitation: State of the art; tools and technologies", *Disaster Prevention and Management: An International Journal*, vol. 5, no. 1, pp. 30–39, 1996.

[118] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Model poisoning attacks in federated learning", in *Proc. Workshop Secur. Mach. Learn.(SecML) 32nd Conf. Neural Inf. Process. Syst.(NeurIPS)*, 2018, pp. 1–23.

[119] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks", *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.

[120] R. S. S. Kumar, D. O. Brien, K. Albert, S. Viljöen, and J. Snover, "Failure modes in machine learning systems", *arXiv preprint arXiv:1911.11034*, 2019.

[121] D. Hadfield-Menell, S. Milli, P. Abbeel, S. J. Russell, and A. Dragan, "Inverse reward design", *Advances in neural information processing systems*, vol. 30, 2017.

[122] Y. Yuan, Z. L. Yu, Z. Gu, X. Deng, and Y. Li, "A novel multi-step reinforcement learning method for solving reward hacking", *Applied Intelligence*, vol. 49, pp. 2874–2888, 2019.

[123] J. Leike, M. Martic, V. Krakovna, *et al.*, "Ai safety gridworlds", *arXiv preprint arXiv:1711.09883*, 2017.

[124] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses", *arXiv preprint arXiv:1705.07204*, 2017.

[125] C. Szegedy, W. Zaremba, I. Sutskever, *et al.*, "Intriguing properties of neural networks", *arXiv preprint arXiv:1312.6199*, 2013.

[126] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks", *arXiv preprint arXiv:1706.06083*, 2017.

[127] *Mitre att&ck (adversarial tactics, techniques, and common knowledge)*, `https://attack.mitre.org/`, Accessed: 2023-09-19.

[128] *Mitre atlas (adversarial threat landscape for artificial-intelligence systems)*, `https://atlas.mitre.org/`, Accessed: 2023-09-19.

[129] Microsoft. "Threat modeling for ai/ml systems and dependencies". Accessed: 2023-08-11. (2019), [Online]. Available: `https://learn.microsoft.com/en-us/security/engineering/threat-modeling-aiml`.

[130] A. Marshall, J. Parikh, E. Kiciman, and R. Kumar, "Threat modeling ai/ml systems and dependencies", *Security documentation*, 2019.

[131] *Ai security and privacy guide*, Accessed: 2023-12-22. [Online]. Available: `https://owasp.org/www-project-ai-security-and-privacy-guide/#how-to-deal-with-ai-security`.

[132] G. McGraw, H. Figueroa, V. Shepardson, and R. Bonett, "An architectural risk analysis of machine learning systems: Toward more secure machine learning", *Berryville Institute of Machine Learning, Clarke County, VA. Accessed on: Mar*, vol. 23, 2020.

[133] European Union Agency for Cybersecurity (ENISA), *Securing machine learning algorithms*, 2021.

[134] A. Marshall, J. Parikh, E. Kiciman, and R. Kumar. "Ai/ml pivots to the security development lifecycle bug bar". Accessed: 2023-08-11. (2019), [Online]. Available: `https://docs.microsoft.com/en-us/security/engineering/bug-bar-aiml`.

[135] 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.

[136] R. Bitton, D. Avraham, E. Klevansky, *et al.*, "Adversarial machine learning threat analysis in open radio access networks.", 2022.

[137] Z. Shi, K. Graffi, D. Starobinski, and N. Matyunin, "Threat modeling tools: A taxonomy", *IEEE Security & Privacy*, vol. 20, no. 4, pp. 29–39, 2021.

[138] H. Muccini and K. Vaidhyanathan, "Software architecture for ml-based systems: What exists and what lies ahead", in *2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)*, IEEE, 2021, pp. 121–128.

[139] Microsoft. "Ai/ml bug bar". Accessed on: 2023-12-10. (2019), [Online]. Available: `https://learn.microsoft.com/en-us/security/engineering/bug-bar-aiml`.

[140] "Stack overflow developer survey 2023". Accessed on: 2023-12-18, Stack Overflow. (2023), [Online]. Available: https://survey.stackoverflow.co/2023/.

[141] Positiwise. "10 main advantages of react.js development". Accessed on: 2023-12-20. (2023), [Online]. Available: https://positiwise.com/blog/10-main-advantages-of-react-js-development.

[142] Thinkitive. "6 reasons to choose python for backend development". Accessed on: 2023-12-19. (2023), [Online]. Available: https://www.thinkitive.com/blog/6-reasons-to-choose-python-for-backend-development/.

[143] S. Ramírez. "Fastapi documentation". Accessed: 2023-12-15. (2023), [Online]. Available: https://fastapi.tiangolo.com/.

[144] "Aisym4med project". Accessed: 2023-12-03. (2023), [Online]. Available: https://aisym4med.eu/about-the-project/.

[145] C. Bello, E. Albertin, J. Iliarte, and T. Torrijos, "Platform architecture, requirements, and feedback loops (version 2.0)", Tech. Rep., 2024, Version to be delivered.

[146] C. Bello, E. Albertin, J. Iliarte, and T. Torrijos, "Platform architecture, requirements, and feedback loops (version 1.0)", Tech. Rep., 2023.

[147] GitLab. "System usability scale (sus)". Accessed on: 2023-12-04. (2023), [Online]. Available: https://handbook.gitlab.com/handbook/product/ux/performance-indicators/system-usability-scale/.

Abbreviations

AI	Artificial Intelligence
AISym4MED	AI Systems for Medical Applications
AML	Adversarial Machine Learning
API	Application Programming Interface
APT	Advanced Persistent Threat
ATLAS	Adversarial Threat Landscape for AI Systems
ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge
BadNet	Backdoored Neural Network
CAIRIS	Computer-Aided Integration of Requirements and Information Security
CIA	Confidentiality, Integrity, and Availability
CIAA	Confidentiality, Integrity, Availability, Authentication
CWE	Common Weakness Enumeration
CVE	Common Vulnerabilities and Exposures
DFD	Data Flow Diagram
DL	Deep Learning
DoS	Denial of Service
DOM	Document Object Model
DREAD	Damage potential, Reproducibility, Exploitability, Affected users, Discoverability
e.g.	example given
ENISA	European Union Agency for Cybersecurity
FastAPI	Fast Asynchronous API
FMEA	Failure Mode and Effects Analysis
GDPR	General Data Protection Regulation
GPU	Graphical Processing Unit
GUI	Graphical User Interface

HDF5	Hierarchical Data Format version 5
ICT	Information and Communication Technology
i.e.	id est
ISO/IEC JTC 1/SC 42	International Organization for Standardization/International Electrotechnical Commission Joint Technical Committee 1/Subcommittee 42
IT	Information Technology
JSON	JavaScript Object Notation
MLOps	Machine Learning Operations
ML	Machine Learning
NIST	National Institute of Standards and Technology
O-RAN	Open Radio Access Network
OVVL	Open Weakness and Vulnerability Modeler
OWASP	Open Web Application Security Project
PASTA	Process for Attack Simulation and Threat Analysis
SD Elements	Security Compass's Software Security Requirements Management Platform
STRIDE	Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege
SUS	System Usability Scale
TMT	Microsoft Threat Modeling Tool
TD	Threat Dragon

List of Figures

List of Tables

Appendix A

Additional Tables and Figures

Figure 29: AI Asset Taxonomy [11]

Table 21: Data Assets and Associated Stages [11]

Category	**Specific Asset**	**Specific AI-Life Cycle Stage**
Data	Augmented Data Set	Data Pre-processing
	Evaluation Data	Model Tuning
	Labeled Data Set	Data Pre-processing
	Metric Data Set	Feature Selection
	Pre-processed Data Set	Data Pre-processing
	Public Data set	Data Exploration; Data Ingestion
	Raw Data	Data Ingestion
	Testing Data	Model Training
	Training Data	Model Selection / Building; Model Training; Transfer Learning
	Validation Data Set	Model Tuning

Table 22: Model Assets and Associated Stages [11]

Category	**Specific Asset**	**Specific AI-Life Cycle Stage**
Model	Algorithms	Model Training
	Data Pre-Processing Algorithms	Data Pre-processing
	Hyper-parameters	Model Tuning
	Training Algorithms	Model Selection/Building
	Model	Model Training; Model Tuning; Model Selection/Building; Model Deployment; Model Maintenance
	Model parameters	Model Training
	Model performance	Model Training; Model Tuning
	Subspace (Feature) selection Algorithm	Feature Selection
	Trained models	Model Training; Transfer Learning
	Training parameters	Model Selection / Building; Model Training
	Tuned Model	Model Tuning

Table 23: Actor Assets and Associated Stages [11]

Category	Specific Asset	Specific AI-Life Cycle Stage
Actor	Cloud Provider	Data Ingestion; Model Training; Model Tuning
	Data Engineers	Data Ingestion; Data Exploration; Data Pre-processing; Feature Selection; Model Selection/Building, Model Training; Model Tuning; Model Deployment; Model Maintenance
	Data Owner	Business Goal Definition; Data Ingestion; Data Exploration
	Data Provider/Data Broker	Data Ingestion
	Data Scientists/AI designer/AI developer	All stages
	End Users	Business Goal Definition; Data Ingestion; Data Exploration; Model Maintenance; Business Understanding
	Model Provider	Transfer Learning
	Service consumers/Model users	Model Maintenance; Business Understanding

Table 24: Processes Assets and Associated Stages [11]

Category	Specific Asset	Specific AI-Life Cycle Stage
Processes	Data augmentation	Data Pre-processing; Data Exploration
	Data Collection	Data Ingestion
	Data Exploration/Pre-processing	Data Exploration; Data Pre-processing
	Data Ingestion	Data Ingestion
	Data labelling	Data Pre-processing
	Data Storage	Data Ingestion
	Data understanding	Data Exploration
	Feature selection	Feature Selection
	Model adaptation – transfer learning / Model deployment	Transfer Learning
	Model Maintenance	Model Maintenance
	Model selection/building; training; and testing	Model Selection / Building
	Model tuning	Model Tuning
	Reduction/Discretization technique	Feature Selection

Table 25: Environment/Tools Assets and Associated Stages [11]

Category	**Specific Asset**	**Specific AI-Life Cycle Stage**
Environment/tools	Cloud	Data Ingestion; Model Training; Model Tuning
	Communication networks	Data Ingestion
	Communication protocols	Data Ingestion
	Computational platforms	Data Pre-processing; Feature Selection; Model Selection/Building; Model Training; Model Tuning; Transfer Learning; Model Deployment; Model Maintenance
	Data exploration tools	Data Exploration
	Data ingestion platforms	Data Ingestion
	Database management system	Data Ingestion
	Distributed File System	Data Ingestion
	Integrated Development Environment	Data Pre-processing; Feature Selection; Model Selection / Building; Model Training; Model Tuning
	Libraries (with algorithms for transformation, labeling, etc)	Data Exploration; Data Pre-processing; Feature Selection; Model Selection/Building; Model Training; Model Tuning
	Machine Learning Platforms	Data Exploration; Data Pre-processing; Feature Selection; Model Selection / Building; Model Training; Model Tuning; Model Deployment; Model Maintenance
	Monitoring Tools	Data Pre-processing; Feature Selection; Model Selection/Building; Model Training; Model Tuning, Transfer Learning; Model Deployment; Model Maintenance
	Operating System/software	Model Deployment; Model Maintenance
	Optimization techniques	Model Tuning
	Processors	All Stages
	Visualization tools	Data Exploration

Table 26: Artefacts Assets and Associated Stages [11]

Category	**Specific Asset**	**Specific AI-Life Cycle Stage**
Artefacts	Access Control Lists	Data Ingestion
	Composition artefacts: AI models compositions	Data Pre-processing; Feature Selection; Model Selection/Building; Model Training; Model Tuning; Model Deployment; Model Maintenance
	Data and Metadata schemata	Data Ingestion; Data Exploration; Data Preprocessing
	Data displays and plots	Data Exploration
	Data Governance Policies	Data Ingestion
	Data Indexes	Data Ingestion; Data Exploration; Data Preprocessing
	Descriptive Statistical Parameters	Data Exploration
	High-Level Test Cases	Business Goal Definition; Model Deployment
	Informal/Semi-formal AI Requirements, GQM (Goal/Question/Metrics) model	Business Goal Definition
	Model Architecture	Model Selection/Building, Model Deployment
	Model Hardware Design	Model Selection/Building; Model Deployment
	Model Frameworks, Software, Firmware or Hardware Incarnations	Transfer Learning; Model Deployment; Model Maintenance
	Use Case	Business Understanding
	Value Proposition and Business Model	Business Understanding

Appendix B

Threat Lists by Experts

Threats by System Component

Data Anonymization and Backend

Component	Threats
Data Anonymization Toolkit	Spoof, Tamper, Disclose or Leak Data Before Deanonymization (Locally)
Backend	Credential Theft
Backend	Membership Attack
Backend	Linkage Attack
Backend	Differential Attack

Frontend

Component	Threats
Frontend (GUI)	Spoof, Tamper, Disclose or Leak Data in Frontend
Frontend (GUI)	A malicious user can repudiate that low-quality/deceptive data was uploaded
Frontend (GUI)	Remote Code Execution based on Injected Data
Frontend (GUI)	Cross-site Scripting
Frontend (GUI)	Cross-site Request Forgery

Data Transit (Browser Backend)

Component	Threats
Data Transit	Spoof, Tamper, Disclose or Leak Data in Transit
Data Transit	Unauthorized Platform Access: Access synthetic/anonymized data
Data Transit	Unauthorized Platform Access: Leverage compute resources
Data Transit	Unauthorized Platform Access: Inject Data (leading to Data Poisoning)
Data Transit	DoS of Platform Interface
Data Transit	Data is uploaded to impersonated platform
Data Transit	Unfair usage by malicious/greedy user (rate limiting)
Data Transit	Unauthorized Platform Access: Inject Data (leading to RCE in backend or frontend)

Dataset and ML Engine

Component	Threats
Dataset Engine	Insecure design
Dataset Engine	Credential Failures (Dataset Explorer give access to unauthorized users)
ML Engine	Synthetic dataset bias
ML Engine	Balance of real/generated dataset
ML Engine	Low performance on synthetic dataset risks (Low representativity)

Federated Learning

Component	Threats
Federated Learning	Model Privacy Attack (Federated Learning)
Federated Learning	Parameter (Hyperparameter) theft
Federated Learning	Model Auditing Failure
Federated Learning	Algorithm Bias (Fairness risk)
Federated Learning	Model Generalization risk
Federated Learning	Model Training Overfitting (Sandbox code)

Cross-Border Database (Logic) and Model DB Instance

Component	Threats
Cross-Border Database (Logic)	Federated learning Poison Attack
Cross-Border Database (Logic)	Parameters (weights/gradients) leakage in communications
Model DB Instance	Privacy inference attack (inverse inferences)

Orchestrator, Multi-Cloud, and Containerized Architecture

Component	Threats
Orchestrator	Multi-cloud communication failure

Monitoring & Logging

Component	Threats
Monitoring	Log Deletion/Obfuscation by attacker
Logging	Accessing Log Data (Inferring Privacy-sensitive data, credentials etc)
Logging	Insufficient Logging (i.e., unintended threat/design weakness)

Cloud-Based Database Instances

Component	Threats
Cloud-Based DB Instances	Spoof, Tamper, Disclose or Leak Anonymized Data on Database Level
Cloud-Based DB Instances	Database or Sample Unavailability
Cloud-Based DB Instances	Data Injection leading to Remote Code Execution

Infrastructure Level

Component	Threats
Infrastructure	Spoof, Tamper, Disclose or Leak Anonymized Data on Infrastructure Level
Infrastructure	Spoof, Tamper, Disclose or Leak Models on Infrastructure Level
Infrastructure	Infrastructure Service Unavailability
Infrastructure	Cloud Credential Theft (Spoof, Tamper, Disclose, Leak, Fraud, Impersonate)
Infrastructure	Spoof, Tamper, Disclose or Leak Anonymized Data or Models by other tenants

Appendix C

Installation Guidelines

This appendix provides the necessary steps to install and run the *ThreatFinder* prototype. These instructions have been tested in various environments, including MacOS, Windows and Linus, ensuring compatibility with a broad range of environments.

Prerequisites

Before installation, ensure the following prerequisites are met:

1. Access to the *ThreatFinder* Project Source Code:
    - The source code is available on GitHub. Ensure you have access to the repository https://github.com/JSha91/AiThreats.
2. Required Libraries and Tools:
    - Node.js: A JavaScript runtime built on Chrome's V8 JavaScript engine.
    - npm (Node Package Manager): Used for installing dependencies.
    - Python: Preferably version 3.11.7 for backend development.
    - FastAPI: A modern, fast web framework for building APIs with Python.
    - React JS: For frontend development.
    - Additional Python Libraries: Such as (jsonableencoder)

Build and Run Prototype

Setting up the Backend

1. Navigate to the Backend Project Directory:

```
cd path/to/solution-backend
```

2. Install Python Dependencies:

```
pip install -r requirements.txt
```

3. Running the Backend Server:

```
uvicorn main:app --reload
```

Setting up the Frontend

1. Navigate to the Frontend Project Directory:

```
cd path/to/solution-frontend
```

2. Install Node Dependencies:

```
npm install
```

3. Running the Frontend Application:

```
npm start
```

Connecting the Components

1. Import necessary libraries in the threat modeling tool in *ThreatFinder*.
2. Draw your architecture model from scratch in *ThreatFinder* via the threat modeling tool in the frontend interface.
3. Export the architecture model as an XML file.
4. Upload XML file to *ThreatFinder* via the frontend interface.
5. Display, filter, download, and analyze identified threats.

By following these guidelines, users can successfully set up and utilize the *ThreatFinder* tool for automated threat modeling in AI-based systems.

Appendix D

Contents of the CD

1. This thesis as PDF
2. This thesis as LATEX source in a .zip file
3. Midterm presentation slides as PDF
4. The source code of this thesis
5. The responses from the participants used for the evaluation, in a directory called *responses*