UNIVERSITY OF TARTU
Institute of Computer Science
Cybersecurity Curriculum

Mihhail Karagjaur

Asset-Oriented Threat Analysis for Large Language Model Systems

Master's Thesis (21 ECTS)

Supervisor:
Raimundas Matulevičius, PhD

Tartu 2025

Asset-Oriented Threat Analysis for Large Language Model Systems

Abstract:

Large language model (LLM) deployments continue to proliferate across enterprises without systematic guidance on risk analysis of the LLM-based systems. Addressing this gap, the present study designs and validates an asset-oriented threat model, tailored to LLM systems. The research follows a design-science research paradigm. The research method incorporates (1) a systematic literature review of 45 peer-reviewed and grey sources, which led to the definition of 13 parent attack classes, a total of 24 threat variants. (2) A design of a threat model, which formalized the LLM business and system assets, their security criteria, mapped threats, security requirements, and countermeasures. (3) Two validation procedures, comprising a feasibility analysis of the threat model's applicability and an empirical test of a jailbreak attack.

The feasibility analysis determined that the proposed threat model, mapped to the Mistral Small 3.1, achieved a completeness score of 0.93 out of 1.00. Thus, indicating all but one of the seven system assets were fully represented in the real-world system. To further substantiate the applicability of the threat model, a jailbreak attack (prompt-injection) was executed with 100 prompts from the JailbreakV-28K benchmark open dataset. Without an official safety measure enabled, 78% of applicable prompts resulted in harmful output. With the safety measure enabled, the rate of harmful output was reduced to 70%. Indicating partial but insufficient mitigation.

The main artifact of the thesis is an asset-oriented threat model for LLMs. The artifact consists of the following components:

1. High-level UML class and state, and BPMN process diagrams, depicting an LLM system, and mapping elicited threats to the system's assets.
2. An interactive web page, which allows practitioners to traverse the produced threat model and to acquire information about the elicited assets, threats, and proposed countermeasures.
3. Code of the interactive web page, empirical tests, and datasets, supporting local use of the threat model and reproducibility of the jailbreak empirical test.

Findings conclude that the LLM system possesses a wide attack surface while adding unique vectors such as jailbreak and embedding inversion. The thesis provides security and AI engineers with a systematic approach to risk analysis and countermeasure selection. Although the threat model was validated on a single open-weight model, the baseline methodology is model-agnostic and extensible. Future studies could validate the threat model against a wide set of LLM systems and automate control recommendations in the scope of DevSecOps.

Keywords: Artificial Intelligence, Machine Learning, Large Language Model, Cybersecurity, Threat Modeling, Risk Analysis

CERCS: T120 - Systems engineering, computer technology

Varale orienteeritud ohuanalüüs suurte keelemudelisüsteemide jaoks

Lühikokkuvõte:

Suurte keelemudelite (LLM) kasutuselevõtt jätkub ettevõtetes, kuid puuduvad süstemaatilised juhised riskianalüüsi riskianalüüsi teostamiseks. Selle puudujäägi kõrvaldamiseks kavandatakse ja valideeritakse käesolevas uuringus varadele suunatud ohumudel, mis on kohandatud LLM-süsteemidele. Uurimus järgib disainiteaduslikku uurimisparadigmat. Uurimismeetod hõlmab (1) 45 vastastikuse eksperdihinnangu saanud ja hallide allikate süstemaatilist kirjanduse läbivaatamist, mille tulemusel määratleti 13 vanemrünnaku klassi, kokku 24 ohuvarianti. (2) Ohumudeli kavandamist, mis hõlmab endas LLMi äri- ja süsteemivarad, nende turvakriteeriumid, kaardistatud ohud, turvanõuded ja vastumeetmed. (3) Kaks valideerimiseksperimenti/-katset, mis koosnevad ohumudeli rakendatavuse teostatavuse analüüsist ja vangistusrünnaku empiirilisest testist.

Teostatavusanalüüsiga määrati kindlaks, et Mistral Small 3.1-le kaardistatud kavandatud ohumudel saavutas täielikkuse skoori 0,93 punkti 1,00-st. Sellest järeldati et kõik peale ühe süsteemi seitsmest varast olid reaalses süsteemis täielikult esindatud. Ohumudeli rakendatavuse täiendavaks tõestamiseks viidi läbi jailbreak-rünnak (prompt-injection), kasutades avatud andmekogumi JailbreakV-28K andmestikust pärit 100 käsku. Ilma turvameetmeteta andis 78% kohaldatavatest käskudest kahjuliku väljundi. Kui turvameede oli aktiveeritud, vähenes kahjulike väljundite osakaal 70 %-ni. See näitab osalist, kuid mittetäielikku leevendamist.

Lõputöö peamise tulemusena loodi varadele orienteeritud ohumudel. Ohumudel koosneb järgmistest komponentidest:

1. Kõrgetasemelised UML klassi- ja olekud ning BPMN protsessidiagrammid, mis kujutavad LLM-süsteemi ja kaardistavad väljaselgitatud ohud süsteemi varadele.
2. Interaktiivne veebileht, mis võimaldab kasutajatel läbida loodud ohumudelit, et saada teavet väljaselgitatavate varade, ohtude ja kavandatud vastumeetmete kohta.
3. Interaktiivse veebilehe ning empiiriliste testide kood, ja andmekogumid, mis toetavad ohumudeli kohalikku kasutamist ja jailbreaki empiirilise testi reprodutseeritavust.

Tulemustest järeldub, et LLM-süsteemil on lai ründepind, lisades samas unikaalseid ohuvektoreid, nagu jailbreak ja embedding inversion. Diplomitöö annab turva- ja tehisintellekti inseneridele süstemaatilise lähenemisviisi riskianalüüsiks ja vastumeetmete valimiseks. Kuigi ohumudel valideeriti ühe avatud kaaluga mudeli peal, on baasmudelite metoodika mudelite suhtes universaalne ja laiendatav. Tulevased uuringud võiksid valideerida ohumudelit laia LLM-süsteemide kogumi suhtes ja automatiseerida kontrollisoovitusi DevSecOps'i raames.

Võtmesõnad: Tehisintellekt, Masinõpe, Suur Keelemudel, Küberturvalisus, Ohu Modelleerimine, Riskianalüüs

CERCS: T120 - Süsteemitehnoloogia, arvutitehnoloogia

Table of Contents

Table of Figures

Table of Tables

1. Introduction

1.1 Motivation

Advancements within the artificial intelligence (AI) field resulted in the appearance of an AI based on novel models, such as a large language model (LLM). These systems receive wider use across the general population and enterprises, across diverse sets of applications [1], becoming a more attractive target to adversaries. For instance, adversaries can exploit vulnerabilities such as jailbreaking, model extraction, or adversarial inputs, leading to increased risks of compromise, including data breaches, misinformation propagation, and intellectual property theft. Since the release of ChatGPT and GPT-4, jailbreaking of LLMs has received broad attention from the general public, suggesting widespread awareness and use of the attack [2]. In addition, as LLMs become integrated into autonomous systems, a concern is raised about their potential misuse in unsupervised environments [1].

Researchers have studied the security of AI systems over the years, discovering attacks [3] against models and proposing defenses [4] against them. However, the sources note that defenses against discovered threats remain an open research problem [4], [5], [6]. Threats' impact has been analyzed in applications, such as UAV inspection, telecommunications, and smart grids [5], [6], [7]. In 2011, an early taxonomy of attacks against machine learning models was published [8]. The literature review covered several taxonomies of attacks against AI systems. Despite the existing work, a need for holistic coverage of risks, influencing factors, and attack relationships was underlined [9]. In addition, the conducted literature search and review have identified an absence of research work compiling a threat model, focusing on LLM systems (Chapter 3). The OWASP Top 10 for LLM Applications [10] provides a collection of top risks, rather than a comprehensive modeling of threats [1]. The absence of a threat model complicates an informed risk analysis of integrated LLM systems within enterprises, and thus countermeasure implementation, increasing the likelihood of successful cyberattacks.

1.2 Problem Statement

Existing research has explored separate types of attacks against AI and LLM systems. There remains a gap in the research attempting to address the cybersecurity landscape of an LLM-based system. The literature review (Chapter 3) showed that there are no instances of research work where a threat modeling framework has been developed or adapted to an LLM systems. The development of a threat modeling framework would allow practitioners to systematically define the existing applicable cybersecurity threats against the components of an LLM system. The absence of such a resource limits the efficiency of a risk analysis that may be conducted on an LLM-based system. This, in turn, may limit the elicitation of applicable threats and necessary countermeasures.

1.3 Scope and Goal

The primary goal of the study is to elicit applicable threats against LLM-based systems and proposed countermeasures. To achieve the stated goals, **the main research question is proposed and is as follows:** How to model threats against large language model systems to identify and categorize security threats, thereby enabling structured risk analysis and selection of countermeasures? Three sub-objectives are defined, the completion of which shall produce an answer to the posed research question:

1. Collection of theory about existing threats and countermeasures to AI and LLM systems.
2. Design of the threat model against LLM systems based on the collected theory.

3. Validation of the designed threat model to ensure its applicability to LLM-based systems.

The scope of the research is limited to:

1. **Threat elicitation**: Elicitation of existing threats and cybersecurity risks that are applicable or potentially applicable to an LLM-based system, with a systematic literature review of peer-reviewed and gray sources.
2. **Component analysis and modeling**: Identification of separate components of an LLM system, such as training and execution systems, used data, and inference application programming interfaces (APIs).
3. **Countermeasures elicitation**: Elicitation of proposed countermeasures to the existing threats.
4. **Development of a threat model**: Mapping and modeling of elicited threats to determined assets of an LLM system. The threat modeling and gathering of data about attacks and countermeasures follow the "Information systems security risk management domain model" [11].
5. The work covers the structure of the LLM and the nature of threats on a high, non-technical level.

Limitation. The work targets high-level architectural components common to AI and LLM-based systems, including input inference APIs, training systems, and supporting infrastructure. Empirical analysis and the feasibility study are limited to a single open-weights model, thus, they may under-represent the applicability of the produced threat model against differing AI system offerings.

1.4 Research Method

The thesis is constructed around the **design-science research (DSR)** paradigm, focusing on the creation of an artifact and its evaluation. The produced artifact is a method, an asset-oriented threat model. In the scope of the paradigm, the thesis executes the following three stages: theory, artifact, and validation, mirroring the proposed sub-objectives (Chapter 1.3).

Theory: a collection of theories about existing threats and countermeasures of AI and LLM systems is conducted with a **systematic literature review (SLR)** in Chapter 3. The SLR identifies, filters, and extracts data about threats and countermeasures for AI/LLM systems. The SLR follows Kitchenham guidelines [12] and consists of the following steps:

1. Definition of search strings for literature search within research paper repositories.
2. Definition of paper inclusion and exclusion criteria.
3. Definition of information extraction criteria.
4. Paper selection, based on the defined search strings and criteria.
5. Analysis of the resulting corpus and extraction of data according to the set criteria.

The literature analysis process is grounded in qualitative analysis of collected evidence from the SLR. The qualitative analysis is based on definitions of the study's subject: LLM structure, threats against it, and security measures.

Artifact: Based on the extracted data, a threat model is designed, mapping elicited LLM system assets with elicited threats in Chapter 4. The threat model is realized through the creation of a threat taxonomy and UML/BPMN diagrams. The threat model is instantiated in the form of an interactive web tool.

Validation: The produced threat model artifact is validated with two separate approaches in Chapter 5. The first approach is conducted as **a feasibility analysis**, with which the designed threat model is applied to a concrete LLM (Mistral Small). In scope of the analysis the threat model's completeness and applicability are quantified. The second approach conducts an empirical jailbreak experiment to support the applicability analysis of the produced threat model.

The results of validation would be considered as a new theoretical input, triggering adjustments to the produced artifact and its re-validation. Execution of the three steps of the design-research paradigm achieves the set sub-objectives and consequently answers the main research question of the thesis.

1.5 Research Structure

The thesis is structured as follows:

1. **Chapter 1 Introduction** - introduction to the thesis, describing the motivation, problem statement, scope, goal, limitations, and utilized research method;
2. **Chapter 2 Background** - provision of theoretical core ML concepts and the reference risk-management domain model;
3. **Chapter 3 Systematic Literature Review** - a systematic literature review of the present body of knowledge, relevant to the thesis' topic. The section details SLR protocol, elicited attack taxonomy, and gap analysis.
4. **Chapter 4 Threat Modeling Large Language Model System** – design and development of a threat model of an LLM system, producing UML/BPMN artifacts and an interactive web presentation of the threat model;
5. **Chapter 5 Validation of Threat Model** – reports the completeness validation of the produced threat model and empirical jailbreak test results;
6. **Chapter 6 Conclusion** – summarizes contributions, outlines conducted research, answers the main research question, presents thesis' novelty and limitations, and proposes directions for further research;
7. **Appendices** - provisioning of additional detailed material.
    a. **I. Glossary** – glossary of terms used within the thesis.
    b. **II. Systematic Literature Review Extracted Data**– presentation of the complete SLR data extraction table, documenting information of the elicited threats and countermeasures according to the followed domain model.
    c. **III. Combination of the SLR Extracted Data** – combined textual presentation of extracted data around the determined types of attacks, an attack taxonomy.
    d. **IV. Codebases**– link to the code of the web solution and empirical test, and datasets used for empirical analysis.
    e. **License -** non-exclusive license to reproduce the thesis and make the thesis public.

2. Background

This section introduces the concepts and technical foundations on which the remainder of the thesis relies. First, the principles of artificial intelligence and machine learning are covered. With an additional coverage of deep neural networks and large language models. In addition, coverage of the threat modeling field and security criteria is provided. Finally, the chapter concludes with the presentation of the "information systems security risk management domain model" [11], which guides the following threat modeling effort.

2.1 Machine Learning and Artificial Intelligence

Artificial intelligence (AI) is a field that has been widely adopted, enabling transformative advancements in various industries, such as healthcare, smart manufacturing, and smart cities [13]. The sources note – there is no generally agreed-upon and established definition of AI. Within sources, the definition may be context-adjusted [14]. Nevertheless, the main characteristics can be derived from a common outline of the concept:

1. AI is a set of methods capable of autonomous unsupervised decision-making [14].
2. The set of methods can adapt to complex environments and unseen circumstances [14].
3. It is an artificially created, machine-based form of intelligence, different from natural intelligence [13].

As of April 2025, the EU AI Act (Article 3: Definitions) defines an AI system as: "AI system' means a machine-based system that is designed to operate with varying levels of autonomy and that may exhibit adaptiveness after deployment, and that, for explicit or implicit objectives, infers, from the input it receives, how to generate outputs such as predictions, content, recommendations, or decisions that can influence physical or virtual environments" [14], [15], [16]. Examples of AI systems include chatbots, recommendation algorithms, image recognition, and generation systems.

While AI has enabled unprecedented technological capabilities [17], AI has been utilized for criminal and harmful activities. This includes the malicious use of AI, when AI is used to enable and augment harmful acts, for example, social engineering, generation and spread of disinformation. As well as the malicious abuse of AI, when the AI system itself is exploited with and aim of compromising its security criteria and criteria of dependent systems.

Machine learning (ML) is a subfield of AI [14] or an AI technique [17]. Sources directed to the following aspects of the ML concept:

1. The goal of machine learning is to create machines that make decisions by learning insights from data [18].
2. ML involves a process that utilizes a learning algorithm against training data, which leads to the creation of a machine learning model [18].
3. An ML system automatically infers patterns and data relations with a training algorithm from a large amount of data [3].
4. ML models generalize to new data – given input data, the ML model produces an output, which accomplishes the given task [18].

There are three prominent methods of training and an ML model:

1. **Supervised learning** – a model is trained on pairs of data and their labels. The model learns the relationship between the training data and an attached label [19], [20].
2. **Unsupervised learning** – a model extracts features from unlabeled data [19], [20].

3. **Reinforcement learning** – the model is trained to perform a task when the best solution for it is unknown. The model conducts attempts at solving the task, the better approaches are provided with a reward value in return. The idea is to improve the model's decision-making capabilities by maximizing the rewards [19].

The creation of machine learning models involves several phases, which can be conceptualized as a pipeline or a lifecycle of an ML model. Below are provided the phases, determined from the sources:

1. **Data collection and preparation** [1], [6]: At this stage, data is captured for training and alignment of a model with a particular task. Data preprocessing is conducted to transform gathered data into a suitable format and to ensure the statistical quality of the created dataset.
2. **Model design and training** [13], [21]**.**
    a. **Model design/selection**: An appropriate ML algorithm is chosen to accomplish the task. Alternatively, a public pre-trained model can be utilized, fitting the purpose. A pre-trained model can be further trained (fine-tuned) to better align it with the intended goal. For LLMs, a "base or foundation model" can be selected, which is a model trained with a generalized dataset [1]. Such models are later fine-tuned and trained with specialized datasets [1].
    b. **Model training** [19], [22]: Model parameters are updated iteratively based on the properties of the training dataset. Example training techniques: backpropagation and gradient descent. The training is guided by optimizing a loss function, which quantifies the model's prediction errors. During the training process, the model's parameters are iteratively adjusted to minimize the error rate, while keeping the model generalized. For LLMs, a public base model can be fine-tuned with a specific dataset to fit a specific purpose [1].
    c. **Validation and testing** [19]: During the training phase, the performance of the produced model is assessed with a validation dataset. The assessment guides the following training iterations to achieve an optimal model performance.
3. **Model deployment** [1], [13]: When the optimal model is produced, it is integrated into the production application, service, or system, while selecting an appropriate platform and infrastructure for it.
4. **Inference phase (operation phase)** [6], [21]: An operation state, when the deployed trained model is utilized to make a decision based on the submitted new, unseen input data. The input data is processed with the learned logic, leading to the generation of output data.
5. **Monitoring and management** [1], [10], [13]: When the model is deployed, its performance and state are continuously monitored. This involves performance metrics, monitoring for model drift, logging of operations, and analysis of security vulnerabilities. Continuing monitoring enables promptly conducting of maintenance activities.

Machine learning methods were determined to be vulnerable to malicious attacks [3], [4]. Vulnerabilities within ML systems are exploited by adversaries to conduct an adversarial attack [4], [6]. Risks have been found, that target different phases of the ML model's lifecycle [6].

In this work, AI and ML terms are used interchangeably. The field of AI has been predominantly advanced by the development of ML algorithms in recent years [5], [8].

2.1.1 Deep Neural Networks

Deep Learning (DL) is a specialization of machine learning [1]. This field covers ML algorithms, which are built upon **deep neural networks (DNNs)** [19], [21]. DL algorithms build a network of connected layers, consisting of computational elements – artificial neurons or nodes [19]. Within the structure, each node processes the input signal by computing a weighted sum of its inputs, the sum is then passed through a non-linear activation function [14], [19]. The weights of the connections between nodes are the parameters, which are adjusted during the training process [19]. DNNs can follow different established architectures, such as Convolutional Neural Networks (CNNs), used for image and spatial data; Recurrent Neural Networks (RNNs) used for handling sequential data [1], [19]. Transformers is one of the DNN architectures, excelling at processing natural language [1], [19].

2.1.2 Large Language Models

Large Language Model (LLM) is a language model, a type of deep neural network model [2], [19]. LLMs are built with Transformed architecture [19], [23]. LLMs are initially pre-trained on a vast textual corpus of data [2]. The collected data undergoes the following preprocessing steps [10]: tokenization, cleaning, and normalization. Tokenization is a process in which an element of the input sequence is mapped to a token [19]. A token can represent a word, a fragment of a word, or a measurement [19]. The goal is to segment the text, numerically represent the data, and increase efficiency in processing. For specific needs, the base, pre-trained models can be fine-tuned with specialized datasets to align the model's functionality with the determined purpose.

Functionally, LLMs process sequential data (textual, but other media can also be supported), predicting the next token in the output sequence and forming a well-structured sentence [2], [19]. Submission of input queries to LLMs can involve prompt engineering. Prompt engineering is the concept of creating an input text (prompt) that would elicit a target output [1], [10]. Deployments with LLMs may incorporate guardrails to limit LLMs' output behavior to defined ethical and legal boundaries [10].

2.2 Threat Modeling

Threat modeling [24] is an aid used for the elicitation of security risks. Threat modeling focuses on the development of models, helping in the search for security problems and the anticipation of applicable threats. The process applies a model of threat to a model of the target system under analysis, enabling the elicitation of applicable threats. Threat models are often represented in the form of diagrams.

The threat modeling process involves four main phases [24]:

1. **Modeling of the target system.**
2. **Elicitation of threats based on the produced model** – application of developed threat models to the target system. Based on the system's details, a set of applicable threats could be determined, as well as suitable countermeasures.
3. **Address the threats** – a decision must be made, whether the applicable threat should be mitigated, eliminated, transferred, or accepted.
4. **Validate the results of the process** – test the implemented security mitigations, ensuring that the applicable threats have been mitigated.

Threat modeling produces the following deliverables [24]:

1. **Diagrams**, depicting the target system, possible threats, and countermeasures, mapped to the system's components.

2. **Identification of trust boundaries.**
3. **List of applicable threats.**
4. **Characterization of the threat agent**, agent's possible goals, level of knowledge, capabilities, and strategy [11].
5. **List of security bugs**.
6. **Documentation** of the process, which may contain attack libraries, vulnerability checklists, and analysis reports.

The threat modeling process brings the following benefits [24]:

1. Engineering and delivery of more secure products and systems.
2. Detection of security bugs.
3. Helps in understanding present security requirements.
4. Detection of defects in the built solution.
5. Helps anticipate the threats.
6. Helps address classes or groups of attacks.
7. Provides a structured framework for analyzing security problems.
8. Helps to proactively identify high-risk areas [1].
9. Helps align security and test goals.

Despite the present benefits, the threat modeling process has the following limitations [24]:

- Threat modeling is a time and resource-consuming process.
- Unsuitable approaches to threat modeling may lead teams to delay the security analysis of the system and inhibit the ability to elicit threats.
- Human-centered threats may be hard to address.

2.2.1 Security Criteria Definitions

The designed threat model operates with the established security criteria concept - CIA, standing for confidentiality, integrity, and availability. Below are the definitions of the 3 utilized security criteria:

- **Confidentiality** assures that the target data is disclosed only to the authorized parties.
- **Integrity** is the guarantee that the target data remains accurate and consistent. All unauthorized modifications or losses are prevented.
- **Availability** is the assurance that the target data is always available to the authorized parties.

2.2.2 Reference Model for Security Risk Management

The attack data extraction follows the domain model for security risk management depicted in Figure 1 [11]. The model is split into 3 sections: assets, security risks, and countermeasures. The model-driven approach to acquiring information about the attacks against LLMs enables a robust and comprehensive information gathering and analysis, as well as consistency in the referred and used concepts.

Assets is a concept encapsulating anything of value to an organization, both at the business and technical levels [11]. The *assets* can be further divided into **business assets** and **system assets (IS assets)**. *Business assets* cover immaterial concepts, such as information, data, processes, skills, and capabilities. While *system assets* encapsulate physical concepts, such as software or hardware components of an IT system, as well as personnel and facilities, which are a part of the IT system. *System assets* enable *business assets*. Assets possess a **security criterion** – a property describing a business asset's security needs.

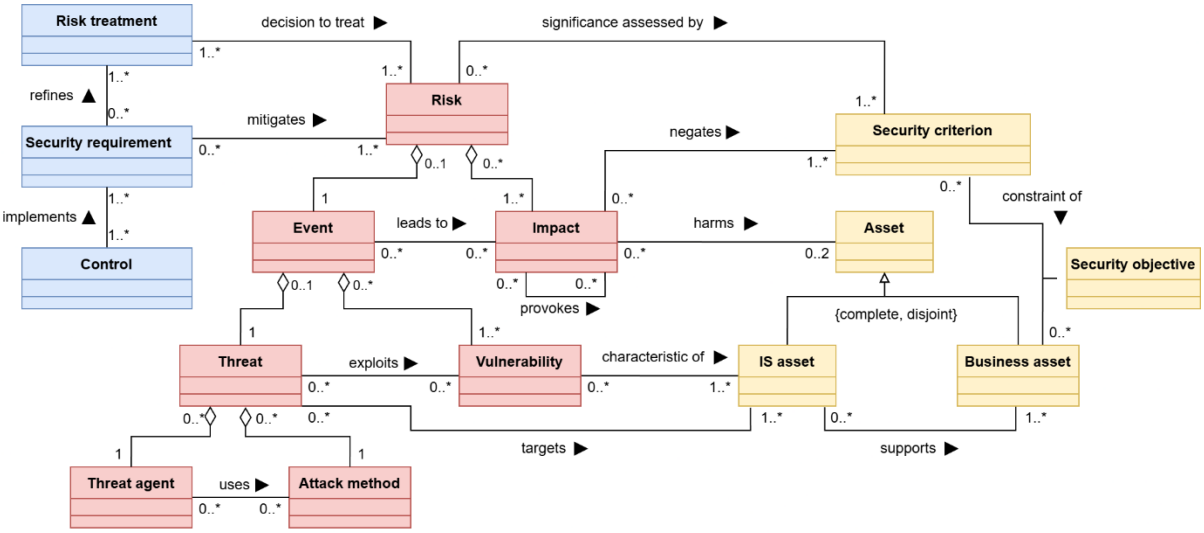

Figure 1. Information systems security risk management (ISSRM) domain model [11].

Risk is a concept defined through the combination of a **threat** with one or multiple **vulnerabilities**, resulting in a negative **impact** on one or more targeted *assets*. *Risk* represents an executed attack against a target *asset* leading to a harmful outcome [11]. *Threat* is a **threat agent**, equipped with an **attack method**. If the *threat* successfully exploits a *vulnerability* within the target *system asset*, this leads to a security **event**, which in combination with the produced *impact* defines the *risk*.

Risk treatment is a concept defining the strategy to tackle the risk [11]. The treatment meets the **security criteria** and defines **security requirements**. *Security requirement* is a condition of the target system, which is reached through the installation of risk mitigation *security controls* to mitigate a risk. *Security control* is a specific solution, improves the security of the target system.

2.3 Summary

Chapter 2 lays out the concepts for the thesis by introducing the AI technologies, security criteria principles, threat modeling, and the ISSRM domain model, which underpin the subsequent research. Chapters 3 and 4 cover the extraction of data from literature and the design of the threat model, respectively, following the presented ISSRM domain model. The data extraction and model design are supported by the presented AI and security concepts.

3. Systematic Literature Review

The goal of the systematic literature review is to achieve the set sub-objective of collecting theory about existing threats and countermeasures to AI and LLM systems. The current chapter fulfills the theory stage of the DSR paradigm. The SLR presents the-state-of the-art on threat research of AI and LLMs, which is going to guide the following threat modeling effort. The following questions have been formulated to guide the systematic literature review:

1. **Main literature search question, [RQ1]:** What is the current state of the art of the threat modeling practice relative to LLM-based systems?
2. **Sub-question 1, [RQ1.1]:** What cybersecurity attacks against artificial intelligence are currently known?
3. **Sub-question 2, [RQ1.2]:** What cybersecurity attacks currently exist specific to large language models?
4. **Sub-question 3, [RQ1.3]:** How to build a threat model, depicting existing cybersecurity attacks to a large language model?

The gathered papers are to be selected and reviewed according to the presented protocol in Chapter 3.1. During the review, key points about attacks against AI and LLM are recorded, following the concepts of the ISSRM domain model, described in Chapter 2.2.2. These records will be used for threat modeling of existing attacks against an LLM-based system.

3.1 Search Strategy and Selection

The search for relevant academic literature was conducted over multiple databases: IEEE digital library, Scopus, ScienceDirect, and ACM Digital Library. In addition, some gray literature sources are included for analysis, which were received outside of repositories dedicated to academic papers, through an unsystematic general search. All the material, sourced through the systematic process, is referenced within Appendix II.

The primary gray source of knowledge is OWASP Top 10 for LLMs [10], which was discovered through a general web search. It was included, as the project aligns with the paper's goals, and it comes from a reputable source. In addition, the project specifically focuses on LLMs, which ensures better coverage of attacks related to the targeted type of machine learning models. From the OWASP source, 2 additional sources were covered: "Information Leakage in Embedding Models" [25], and "Sentence Embedding Leaks More Information than You Expect: Generative Embedding Inversion Attack to Recover the Whole Sentence" [26].

3.1.1 Search Terms

The designed search keyword combinations were aimed at finding relevant research works related to the following topics:

- Threat modeling of AI/ML/LLM;
- Security of AI/ML/LLM;
- Vulnerabilities of AI/ML/LLM;
- Attacks against AI/ML/LLM;
- LLM-specific inference and jailbreak attacks.

A combination of keywords and was used for search of papers, covering the topic of interest. In the case of IEEE Xplore, the following keyword combination was utilized, searching through all parts of papers:

```
("modeling of artificial intelligence" OR
"modeling  of AI" OR "modeling  of machine learning" OR
"modeling  of ML" OR "modeling  of large language model"
OR "modeling  of LLM") OR
("threat model of artificial intelligence" OR "threat model of AI" OR
"threat model of machine learning" OR "threat model of ML" OR
"threat model of large language model" OR "threat model of LLM")
OR ("security of large language models" OR "security of LLM"
OR "security of artificial intelligence" OR "security of AI"
OR "security of machine learning" OR "security of ML")
OR ("vulnerabilities of large language models" OR "vulnerabilities of
LLM" OR
"vulnerabilities of artificial intelligence" OR
"vulnerabilities of AI" OR "vulnerabilities of machine learning"
OR "vulnerabilities of ML") OR
("attacks against machine learning" OR
"attacks against large language model" OR
"attacks against artificial intelligence") OR
(("large language model" OR "LLM") AND
("jailbreak" OR "jailbroken" OR "inference attack"))
```

Figure 2. A list of literature search queries.

In the case of Scopus, ScienceDirect, and ACM Digital Library, the utilized combination of keywords was further refined to narrow down a possible set of provided relevant papers.

1. The search query was split into multiple queries to focus on separate subtopics;
2. The search queries were only applied to the titles of articles;
3. The first sub-query *"("modeling of artificial intelligence" OR "modeling of AI" OR "modeling of machine learning" OR "modeling of ML" OR "modeling of large language model" OR "modeling of LLM")"*, covering the general modeling of AI/ML/LLM systems was omitted, as its previous use provided poor results in the form of provision of irrelevant research.

Below are provided the sub-queries utilized separately to search through Scopus, ScienceDirect, and ACM Digital Library:

1. ("modeling of large language model" OR "modeling of LLM")
2. ("threat model of artificial intelligence" OR "threat model of AI" OR "threat model of machine learning" OR "threat model of ML" OR "threat model of large language model" OR "threat model of LLM")
3. ("security of large language models" OR "security of LLM" OR "security of artificial intelligence" OR "security of AI" OR "security of machine learning" OR "security of ML")
4. ("security of large language models" OR "security of LLM")
5. ("vulnerabilities of large language models" OR "vulnerabilities of LLM" OR "vulnerabilities of artificial intelligence" OR "vulnerabilities of AI" OR "vulnerabilities of machine learning" OR "vulnerabilities of ML")
6. ("vulnerabilities of large language models" OR "vulnerabilities of LLM")
7. ("attacks against machine learning" OR "attacks against large language model" OR "attacks against artificial intelligence")
8. ("attacks against large language model")

9. (("large language model" OR "LLM") AND ("jailbreak" OR "jailbroken" OR "inference attack"))

3.1.2 Inclusion and Exclusion Criteria

The inclusion criteria considered for paper selection:

- Research related to security, attacks, and threat modeling of large language models;
- Research related to security, attacks, and threat modeling of artificial intelligence;
- Research related to security, attacks, and threat modeling of machine learning;
- Research related to large language models.

The exclusion criteria considered for paper selection:

- Works not written in the English language;
- Not freely accessible papers;
- The academic paper is shorter than 10 pages.

3.1.3 Paper Selection

The selection of relevant research papers from academic digital repositories was conducted by defined keyword combinations in Chapter 3.1.1 throughout the February-April period in 2024. The quantitative result of the search is provided in Table 1. The total number of papers under review is 45. The selection was conducted with the following steps:

1. Conduct a request based on the described combination of keywords;
2. Manually inspect the provided set of research papers based on their title and abstract, following the defined inclusion and exclusion criteria, as defined in Chapter 3.1.2;
3. The collected set of papers was additionally checked for duplicates across digital repositories;
4. Papers that did not fit some of the exclusion criteria, but were found to be fitting the topic otherwise, were reintroduced.

Table 1. A table of the number of selected research papers for review.

Digital library	**Initial keyword search**	**Manual inspection and de-duplication**	**Expanded selection through "snowballing"**	**Reintroduction of an excluded**	**Sum of the selected papers per repository**
IEEE Xplore	137	25	0	1	26
ACM Digital Library	7	4	0	0	4
ScienceDirect	9	6	4	0	10
Scopus	67	5	0	0	5
[Total]					**45**

3.2 Information Extraction Criteria

The goal of the information extraction is to cover existing attacks against AI, ML, or LLMs. As well as suggested countermeasures to the covered attacks if such were presented within the covered papers. Additionally, it is of interest to extract information about the utilized or

developed frameworks and modeling approaches towards threat modeling of AI, ML, and LLM systems. The LLM systems are under the greater focus of analysis. The synthesized findings on the attacks can be found in the following Chapter 3.3, and summaries of related work are in Chapter 3.4.

The following information was extracted into a table, which is attached in Appendix II: **Paper Title, Authors, Year of being published, Type of target AI, Attack name(-s), System Asset, Business Asset, Security Criteria, Vulnerability, Threat Agent, Attack Method, Impact and Harm, Security Requirements, Security Control.** The table contains entries for separate attacks, targeting machine learning models of different types, covered within the selection of papers. Additional specifics of entry formation within the table are provided in the referenced appendix.

Specific to this research, the following concepts of the reference model (described in Chapter 2.2.2), carrying isolated data, were filled out per unique attack (risk) during the data extraction process. Their meaning was adjusted to fit the target domain of AI security.

1. **Business assets** – various types of data on which an LLM operates, or which are a part of an LLM.
2. **System assets** – hardware, software processes, and components of a system, surrounding an LLM.
3. **Security criteria –** confidentiality, integrity, and availability of the targeted business asset.
4. **Vulnerability –** a weakness in an LLM model or surrounding process, exploitation of which leads to a negative impact on the targeted business asset and subsequently on the LLM model.
5. **Threat agent –** an adversary, the capabilities of which were covered in 3 scenarios: black-box, grey-box, and white-box. Black-box scenario – an adversary does not possess access to the confidential knowledge of the model's design, data used for training and operation of the model, the target model itself, and the surrounding system. White-box scenario – an adversary who has full knowledge of the target system, uses data, and related processes, and has access to the target system. Grey-box scenario – a mixed scenario, when an adversary may have partial knowledge or knowledge of a few separate system components or data.
6. **Attack method –** a definition, description of a specific attack technique.
7. **Impact and harm –** a negative outcome from a successful attack against a business asset. It is a compromise of one or multiple security criteria of the targeted business asset, as well as a negative impact, which may affect other dependent business assets and related system assets, including the LLM itself.
8. **Security requirements –** a security requirement for the LLM system to meet, to become resistant to the related attack.
9. **Security control –** tools, algorithms, and changes in the processes, which, if implemented, lead to increased resistance against the attack.

An aggregated table (available in textual form in Appendix III) was formed in a separate sheet, only containing values for the following columns: **Attack name(-s), System Asset, Business Asset, Security Criteria, Vulnerability, Threat Agent, Attack Method, Impact and Harm, Security Requirements, Security Control**. The table combines separate entries from the main table under the same type of attacks, combining discovered values for the above-mentioned columns. The table itself is linked in Appendix IV.

In addition, summaries are presented on a subset of papers, which focused on threat modeling, rather than an attack or subset of attacks, with additional coverage of presented threat modeling ideas. Comparison between the related subset of papers and the thesis can be found in Chapter 3.4. Finally, the posed literature review questions are answered based on the results of the extraction of the relevant information in Chapter 3.6. The extracted information of a qualitative nature will be utilized in the definition of attacks against LLMs, as well as the development of the threat model.

3.3 Discovered Attacks and Their Definitions

In this chapter, the discovered attacks are presented, with their risk definition. The risk definitions are followed by a single example, taken from Appendix II. Further collected information about the attacks, including all the studied examples with security requirements and controls, can be found for each attack in Appendix III, in attack specific subchapters. For a high-level depiction, all the attacks and their subtypes are presented with a tree diagram in Figure 3.

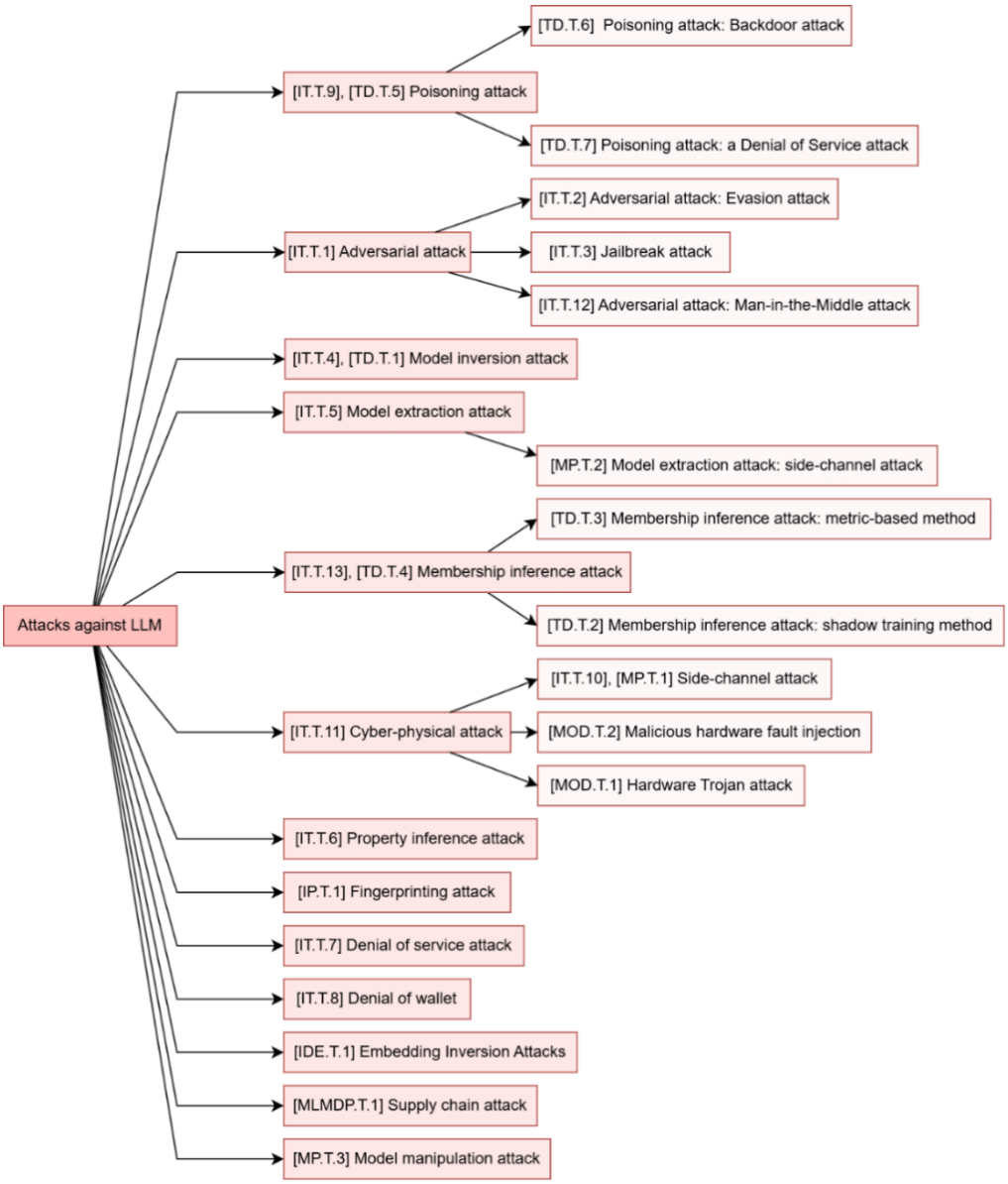

Figure 3. A tree diagram of attack types against LLMs.

The pages that follow cover each elicited attack from the literature review. Each attack is presented in the form of a risk definition following concepts of the ISSRM domain model (Chapter 2.2.2). Each risk definition contains a high-level description of the attack, as well as an example. The attack examples are represented with the following ISSRM domain model concepts:

- targeted system assets;
- targeted business assets;
- compromised security criteria, following the CIA concept of a business asset in case of a successful attack execution;
- a list of vulnerabilities, which can be exploited for attack's execution;
- assumed capabilities of a threat agent, needed for the execution of the attack;
- a high-level description of the employed attack method;
- resulting impact and harm in case of a successful attack.

[IT.T.9], [TD.T.5] Poisoning attacks involve an adversary compromising a machine learning (ML) model by manipulating the training data [4], [27], [28]. The attacker injects malicious data into the training dataset or alters the original training data [29]. The high-level goal is to maximize the generalization error in the classification process or reduce the system's performance [4], [8]. These attacks occur during the training process, aiming to shift the decision boundaries of classifiers [29].

An example of a specific risk is presented below from [8] source:

System Asset: machine learning training system.

Business Asset: training data.

Security Criteria: availability of training data.

Vulnerability:

1. The training dataset is susceptible to unauthorized modifications.
2. The target model's performance can be skewed through training on malicious samples.
3. Lack of training in data integrity checks.
4. Lack of sanitization of publicly acquired training data.

Threat Agent: white-box and grey-box scenarios.

Attack Method: An attacker chooses an initial guess for each poisoned sample. The chosen guess is used to retrain the model. The difference in the model's performance is used to update the guessed poisoned sample through the computation of the (sub)gradient-ascent algorithm. The final set of produced poisoned samples is injected into the target model's data set.

Impact and Harm: Negates the availability of the targeted machine learning model. This leads to reduction in model's accuracy.

[TD.T.6] A backdoor attack is a specific type of **poisoning attack** where adversaries modify the labels of training samples and inject these mislabeled data with backdoor triggers into the training dataset [21]. The goal is to force the trained model to assign a desired target label to new samples containing the trigger [21].

An example of a specific risk is presented below from [30] source:

System Asset: machine learning training system.

Business Asset: training data.

Security Criteria: integrity of training data.

Vulnerability:

1. The training dataset is susceptible to unauthorized modifications.
2. The model's classification boundary is prone to compromise if trained on malicious data samples.

Threat Agent: white-box scenario.

Attack Method: Procedural noise is crafted using algorithms, such as Perlin noise, Gabor noise, and Worley noise. A clean model is trained on original images to get the attention images. The attention images are fused with noise and then added to the original image. The target model is trained on the poisoned data, turning into a backdoored model.

Impact and Harm: Negates the integrity of the targeted machine learning model. This may lead to collateral damage due to the incorrect operation of the machine learning system.

[TD.T.7] A poisoning denial-of-service attack is a type of **adversarial attack** where an attacker manipulates the training data of a machine learning (ML) model with the explicit goal of disrupting the system's availability [4], [21], [31]. This involves injecting malicious data or altering existing data into the training set to degrade the model's performance and cause a denial of service [4], [6], [21], [27], [29].

An example of a specific risk is presented below from [4] source:

System Asset: machine learning training system.

Business Asset: training data.

Security Criteria: availability of training data.

Vulnerability:

1. The training dataset is susceptible to unauthorized modifications.
2. The public data that is utilized for training may contain malicious samples.

Threat Agent: white-box and black-box scenarios.

Attack Method: The attacker modifies the target dataset features and labels to maximize the loss function for targeted samples.

Impact and Harm: Negates the availability of the targeted machine learning model. This leads to misclassification of benign, legitimate input.

[IT.T.1] Adversarial attacks, often also known as evasion attacks, are malicious attempts to fool or subvert machine learning (ML) models by exploiting weaknesses in their algorithms or training data [6], [13], [32]. In this paper, these attacks are considered to occur during the testing/inference phase (evasion attacks) [4], [21]. The goal of these attacks is to undermine the performance, reliability, or security of ML systems through malicious input data [6], [21].

An example of a specific risk is presented below from [7] source:

System Asset: ML system input/API.

Business Asset: input data.

Security Criteria: integrity of input data.

Vulnerability: There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model.

Threat Agent: white-box and black-box scenarios.

Attack Method: A submission of an adversarial sample. A set of suitable perturbations is determined by solving a constrained optimization problem. Afterward, the set of perturbations is reduced based on imposed spatial and physical constraints. The suitable perturbation is applied to the target setting, resulting in an adversarial sample.

Impact and Harm: Negates the integrity of the targeted machine learning model. In the context of the infrastructure inspection, this could lead to severe infrastructure safety concerns.

[IT.T.2] An evasion attack is a type of **adversarial attack** where an adversary manipulates input data at the test or inference stage to cause a trained machine learning (ML) model to misclassify it, thus evading correct detection or classification [6], [8], [21], [27]. The core idea is to exploit vulnerabilities or blind spots in the model without altering the training data or the model's parameters [4].

An example of a specific risk is presented below from [3] source:

System Asset: ML system input/API.

Business Asset: input data.

Security Criteria: integrity of input data.

Vulnerability: There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model.

Threat Agent: white-box and black-box scenarios.

Attack Method: Iterative probing, gradient estimation, or using a surrogate model to craft adversarial examples. Characteristics of a sample are manipulated to turn it into an adversarial sample. Adversarial samples are submitted to the model to cause misclassification.

Impact and Harm: Negates the integrity of the targeted machine learning model. This may lead to misclassification of benign and malicious inputs.

[IT.T.12] A Man-in-the-Middle (MitM) attack in the context of machine learning is a type of **adversarial attack** where an attacker stealthily intercepts and alters the communication between two parties (e.g., a data source and a machine learning classifier) to deliver malicious payloads or manipulate the data, to compromise the integrity or availability of the ML system [33].

An example of a specific risk is presented below from [33] source:

System Asset: machine learning model.

Business Asset: input data.

Security Criteria: integrity of input data.

Vulnerability: There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model.

Threat Agent: white-box and black-box scenarios.

Attack Method: 1. Train a Variational Autoencoder (VAE) with a malicious decoder (MVD). 2. MVD is fine-tuned to produce adversarial samples. 3. Integrate the VAE with the trained MVD into the chain either between the raw input data source and the classifier, or swap out the existing decoder with MVD from the present VAE. 4. As data passes through, the MVD transforms the encoded representation of the input data into an adversarial example.

Impact and Harm: Negates the integrity of the targeted machine learning model. This leads to misclassification of malicious input.

[IT.T.3] A jailbreak attack is a type of security attack that exploits vulnerabilities within a constrained system (such as an aligned LLM) to bypass imposed restrictions and achieve privilege escalation [2]. In the context of LLMs, jailbreaking refers to the practice of circumventing or overriding alignment guardrails that are designed to govern the scope of content the model can produce [2].

An example of a specific risk is presented below from [2] source:

System Asset: ML system input/API.

Business Asset: input data.

Security Criteria: integrity of input data.

Vulnerability: It is possible to bypass built-in output restrictions of a VLM by providing malicious input data in an alternative medium, as an adversarial image or textual data.

Threat Agent: white-box and black-box scenarios.

Attack Method: 1. A submission of an adversarial image input alongside a textual prompt, requesting a malicious output. Adversarial image data is generated based on examples of malicious content through Projected Gradient Descent (PGD); this image data is later paired with the malicious textual prompt. 2. A submission of an adversarial textual input alongside a textual prompt, requesting a malicious output. A discrete optimization algorithm from Shin et al., an improved version of the hotflip attacks, can be utilized for adversarial text generation.

Impact and Harm: Negates the integrity of the targeted machine learning model. This may lead to legal repercussions.

[IT.T.4], [TD.T.1] A model inversion attack is a type of privacy attack where an adversary aims to reconstruct training samples from a machine learning model by exploiting the model's outputs [9]. The goal is to infer specific features or attributes of the hidden input data used to train the model [32]. This type of attack allows an adversary to directly learn information about the training dataset [9].

An example of a specific risk is presented below from [34] source:

System Asset: ML system input/API.

Business Asset: input data.

Security Criteria: confidentiality of input data.

Vulnerability: It is possible to recover features of the training data based of predictions retrieved from the target machine learning model.

Threat Agent: white-box (for feature extractor) and black-box scenarios.

Attack Method: 1. A custom feature extractor is trained upon auxiliary data. 2. The public auxiliary data is fed to the feature extractor and the target model to produce feature-prediction probability pairs. 3. An inverse model is trained on the feature-prediction probability data, mapping prediction data to the feature space. 5. The inverse model produces a feature based on the provided label data. 4. Output of a GAN model (trained on public, auxiliary data) is produced based on the inverse model's generated features. 5. The produced sample by GAN is fed to the target model to acquire new predictions. 6. Features are extracted from the generated image and compared to the features previously produced by the inverse model, accounting for the new predictions. 7. Thus, the features are updated iteratively with the differential evolution (DE) optimization algorithm. 8. A new image is generated with the GAN model based on the updated features.

Impact and Harm: Negates the confidentiality of the targeted machine learning model. This may lead to leakage of sensitive data. Exposure of the sensitive private data may lead to legal repercussions.

[IT.T.5] A model extraction attack (also known as model stealing) is a type of security attack where an adversary aims to replicate the functionality of a target machine learning model without having direct access to its internal parameters or training data [3], [18], [35], [36], [37]. The attacker interacts with the target model, typically through a prediction API, to gather information and train a substitute model that mimics the behavior of the original [3], [35], [37]. This allows the adversary to gain insights into the training data and potentially launch further attacks, such as evasion or membership inference attacks [3], [37].

An example of a specific risk is presented below from [37] source:

System Asset: ML system input/API.

Business Asset: input data.

Security Criteria: confidentiality of input data.

Vulnerability: It is possible to replicate target model's properties within a new model based on the target machine learning model's output.

Threat Agent: black-box scenarios.

Attack Method: The target model is queried with surrogate data, based on the public data, creating a dataset of surrogate data-target model output data pairs. A publicly available pre-trained Data-efficient Transformer (DeiT) model is fine-tuned based on the previously produced dataset.

Impact and Harm: Negates the confidentiality of the targeted machine learning model. This may lead to the loss of intellectual property.

[MP.T.2] Model extraction attack, utilizing **side-channel attack** methods, leverages hardware implementation vulnerabilities to determine the target machine learning model’s architecture and parameter values [35].

An example of a specific risk is presented below from [35] source:

System Asset: Processing hardware running the ML model.

Business Asset: model's parameters.

Security Criteria: confidentiality of the model's parameters.

Vulnerability: The state of a machine learning model's operational component data can be inferred through measurable attributes of the machine learning, such as power consumption, EM emission, and memory access pattern.

Threat Agent: gray-box scenario.

Attack Method: A malicious fault is introduced into the operations of the hardware, processing the target machine learning model's operations.

Impact and Harm: Negates the confidentiality of the targeted machine learning model. This may lead to the intellectual property theft.

[IT.T.13], [TD.T.4] A membership inference attack (MIA) is a type of privacy attack where an adversary tries to determine whether a specific data record was part of the training dataset of a machine learning model [6], [20], [36], [38], [39]. It exploits the tendency of ML models to behave differently on data they have been trained on compared to unseen data [6], [36], [38], [38], [39]. A successful MIA signifies that the privacy of the training data was not sufficiently protected when the trained ML model was released [39].

An example of a specific risk is presented below from [38] source:

System Asset: ML system input/API (of the GAN's discriminator model).

Business Asset: input data.

Security Criteria: confidentiality of input data.

Vulnerability: It is possible to learn additional information about the training data sample from the target model's output.

Threat Agent: white-box scenario.

Attack Method: GAN's discriminator model is used to deduce whether a particular data instance was part of the training dataset of a target model. This is achieved by exploiting differences in the model's response to known (trained on) versus unknown input data.

Impact and Harm: Negates the confidentiality of the targeted machine learning model. This leads to privacy leakage and potential legal repercussions.

[TD.T.2] The shadow training method is an approach used in **membership inference attacks** where the attacker trains multiple "shadow models" to mimic the behavior of the target machine learning model [9], [20], [35], [39]. The attacker uses these models to understand how the target model might behave differently on data it has seen during training versus data it has not [20], [39].

An example of a specific risk is presented below from [35] source:

System Asset: ML system input/API.

Business Asset: training data.

Security Criteria: confidentiality of training data.

Vulnerability:

1. It is possible to learn additional information about the training data sample from the target model's output.
2. The model tends to memorize its training data in the case of over-parametrization.

Threat Agent: white-box and black-box scenarios.

Attack Method: Multiple shadow models are built, imitating the target model's structure and training process. These shadow models are trained on datasets where the membership status is known. The outputs (e.g., confidence scores) are collected from these shadow models for both member and non-member data. An attack model is then trained to distinguish between members and non-members based on these outputs. The attacker queries the target model with data of unknown membership and uses the attack model to infer membership status against the target model, covering the same domain.

Impact and Harm: Negates the confidentiality of the targeted machine learning model. This may lead to the private data leakage and possible legal repercussions.

[TD.T.3] A metric-based membership inference attack is an approach where the attacker calculates a metric on the prediction vectors of a data record and compares it to a predetermined threshold to determine its membership status [35]. This type of attack is generally simpler and less computationally expensive than shadow model-based attacks [35].

An example of a specific risk is presented below from [35] source:

System Asset: ML system input/API.

Business Asset: training data.

Security Criteria: confidentiality of training data.

Vulnerability:

1. The model tends to memorize its training data in the case of over-parametrization.
2. It is possible to learn additional information about the training data sample from the target model's output.

Threat Agent: white-box and black-box scenarios.

Attack Method: Observer target machine learning model's behavior on training data versus unseen data. Analysis of metrics such as confidence scores, loss values, or entropy, patterns indicative of membership are identified. Statistical thresholds or anomalies in the metrics are utilized to determine the membership of the target sample. Membership is inferred when the metrics for a given input differ significantly from those expected for non-members.

Impact and Harm: Negates the confidentiality of the targeted machine learning model. This may lead to the private data leakage and possible legal repercussions.

[IT.T.11] Cyber-physical attacks against machine learning models refer to attacks that exploit the interaction between the cyber (computing and communication) components and the physical components of a system [14]. These attacks target machine learning models that are integrated into cyber-physical systems (CPS), aiming to cause physical consequences by manipulating the data, the training process, or the model itself [14].

An example of a specific risk is presented below from [14] source:

System Asset: processing hardware running the ML model.

Business Asset: input data.

Security Criteria: availability of input data.

Vulnerability: A machine learning system is susceptible to vulnerabilities in software and hardware on which it depends.

Threat Agent: None – none was stated or could be derived from the source.

Attack Method: An Exploit vulnerability in the software or hardware is exploited, on which the target machine learning system depends is exploited.

Impact and Harm: Negates the availability of the targeted machine learning model.

[MOD.T.2] A malicious hardware fault injection attack is a type of hardware-oriented security threat where an adversary intentionally introduces faults or errors into the physical hardware on which a machine learning model is running [40]. This is done to compromise the model's integrity, leading to misclassification or other undesirable behaviors [40]. Unlike software-based attacks, hardware fault injections directly manipulate the ML model's parameters and computation results by tampering with the inference process without manipulating the sample or training data [40].

An example of a specific risk is presented below from [40] source:

System Asset: processing hardware running the ML model.

Business Asset: model's operational data.

Security Criteria: integrity of the model's operational data.

Vulnerability: The state of machine learning model's operational data can be influenced through hardware state manipulation with fault injection.

Threat Agent: white-box and black-box scenarios.

Attack Method: A malicious fault is introduced into the operations of the hardware, processing the target machine learning model's operations.

Impact and Harm: Negates the integrity of the targeted machine learning model. This may lead to misclassification of benign and malicious inputs.

[MOD.T.1] A hardware trojan (HT) attack is a type of attack where malicious circuitry is covertly inserted into the hardware that implements a machine learning model [40]. This hidden circuitry, once triggered, can manipulate the model's behavior, leading to misclassification, data leakage, or other security breaches [40]. Unlike software-based attacks, HT attacks directly target the hardware to alter the model's parameters and computation results [40].

An example of a specific risk is presented below from [40] source:

System Asset: processing hardware running the ML model.

Business Asset: model's operational data.

Security Criteria: confidentiality of the model's operational data.

Vulnerability: The state of a machine learning model's operational component data can be inferred through measurable attributes of the machine learning, such as power consumption, EM emission, and memory access pattern.

Threat Agent: black-box scenario.

Attack Method: Measurable attributes of hardware, processing target machine learning model's operations, are observed. The observed and measured metrics are analyzed to infer the machine learning model's structure.

Impact and Harm: Negates the confidentiality of the targeted machine learning model. This may lead to the loss of the intellectual property.

[IT.T.10], [MP.T.1] A hardware side-channel attack is a type of attack that exploits vulnerabilities in the physical implementation of a machine learning (ML) model to extract sensitive information, such as model parameters, training data, or the model's architecture [40]. Instead of targeting the ML algorithm directly, these attacks measure and analyze side-channel information that is correlated with the ML assets [40].

An example of a specific risk is presented below from [40] source:

System Asset: processing hardware running the ML model.

Business Asset: model's operational data.

Security Criteria: confidentiality, integrity, and availability of the model's operational data.

Vulnerability: The state of a machine learning model's operational data can be influenced through a hardware trojan inserted into the integrated circuits.

Threat Agent: black-box scenario.

Attack Method: A hardware trojan is embedded into an integrated circuit. The embedded hardware trojan modifies the chain of operations based on a trigger, thus modifying the expected chain of operations during machine learning operations.

Impact and Harm: Negates the confidentiality, integrity, and availability of the targeted machine learning model. This may lead to misclassification of benign and malicious inputs, degraded performance, or private data leakage.

[IT.T.6] A property inference attack is a type of privacy attack that aims to infer confidential information or attributes about the training dataset used to train a machine learning model [35]. The attacker attempts to determine certain properties or characteristics that are present in the training data, which the model provider does not want to reveal [35]. This attack doesn't directly manipulate the model but extracts private information without disrupting the model’s normal training process [35].

An example of a specific risk is presented below from [35] source:

System Asset: ML system input/API.

Business Asset: input data.

Security Criteria: confidentiality of input data.

Vulnerability: The Model tends to memorize its training data in case of over-parametrization. It is possible to learn additional information about the training data sample from the target model's output.

Threat Agent: white-box and black-box scenarios.

Attack Method: The target model's output and parameters are analyzed to determine the properties of the utilized training dataset.

Impact and Harm: Negates the confidentiality of the targeted machine learning model. This may lead to the private data leakage and possible legal repercussions.

[IP.T.1] A fingerprinting attack aims to uniquely identify a specific machine learning model instance or to determine which model or family of models is being used in a black box setting [41]. The goal is to derive a signature that is unique to a particular model, similar to human fingerprint biometry [41].

An example of a specific risk is presented below from [41] source:

System Asset: ML system input/API.

Business Asset: intellectual property.

Security Criteria: confidentiality of intellectual property.

Vulnerability: The model's unique decision boundaries and output patterns serve as a "fingerprint"; benign inputs reveal characteristic outputs.

Threat Agent: black-box scenario.

Attack Method: A set of benign queries is sent to the target set of models. The responses from models are compared to the known responses from the targeted model for statistical similarity.

Impact and Harm: Negates the confidentiality of the targeted machine learning model.

[IT.T.7] A Denial of Service (DoS) attack aims to disrupt normal functioning and reduce the availability of a machine learning system, making it unusable for legitimate users [28], [31]. This is typically achieved by overwhelming the system with a high volume of requests or resource-intensive tasks, exhausting its computational resources [10].

An example of a specific risk is presented below from [31] source:

System Asset: ML system input/API.

Business Asset: input data.

Security Criteria: availability of input data.

Vulnerability: The target system hosting the machine learning system can be overwhelmed and made unavailable with computationally expensive requests.

Threat Agent: white-box scenario.

Attack Method: The adversary overwhelms the system by creating many computationally expensive input requests.

Impact and Harm: Negates the availability of the targeted machine learning model. This leads to potential system failures, service unavailability, an increase in processing times, and a reduction in the quality of service.

[IT.T.8] A Denial of Wallet (DoW) attack is a type of attack where an adversary exploits the cost-per-use model of cloud-based AI services by generating an excessive number of operations or resource-intensive tasks [10]. This leads to unsustainable financial burdens on the service provider, potentially causing financial strain or even financial ruin [10].

An example of a specific risk is presented below from [10] source:

System Asset: ML system input/API.

Business Asset: input data.

Security Criteria: availability of input data.

Vulnerability: High cost of model services' operation.

Threat Agent: black-box scenario.

Attack Method: Initiation of a high volume of requests, and operations.

Impact and Harm: Negates the availability of the targeted machine learning model. This may lead to unsustainable financial burdens.

[IDE.T.1] An embedding inversion attack exploits vulnerabilities to invert embeddings and recover significant amounts of source information, compromising data confidentiality [10].

An example of a specific risk is presented below from [10] source:

System Asset: supporting IT infrastructure.

Business Asset: input data embeddings.

Security Criteria: confidentiality of input data embeddings.

Vulnerability:

1. Embedding vectors retain enough information to enable the reconstruction of significant parts of the original text.
2. The leakage or unauthorized access to the embeddings.

Threat Agent: white-box and black-box scenarios.

Attack Method: Inversion of embeddings, leading to recovery of source information. Utilize gradient-based (white-box) or learning-based (black-box) methods to invert the target embeddings. The produced mappings will partially reveal the original input to the model.

Impact and Harm: Negates the confidentiality of previously provided input to the machine learning system, by extension model's confidentiality is compromised in addition. This may lead to legal repercussions.

[MLMDP.T.1] A supply chain attack targets vulnerabilities in the machine learning (ML) supply chain to compromise the integrity, security, and trustworthiness of ML models and their deployment platforms [10]. These attacks exploit weaknesses in third-party components, training data, pre-trained models, and deployment infrastructure [10].

An example of a specific risk is presented below from [10] source:

System Asset: enabling software components.

Business Asset: machine learning development process.

Security Criteria: integrity, and availability of the machine learning development process.

Vulnerability:

1. Deployment platform vulnerabilities.
2. Traditional software vulnerabilities: code and dependency flaws.
3. Corrupted, vulnerable 3rd-party pre-trained models.
4. Usage of unmaintained software, and models.

5. Vulnerable LoRa adapters, a malicious adjustment of the pre-trained model.
6. Development platform vulnerabilities.

Threat Agent: black-box scenario.

Attack Method: Infection or impersonation of software components, public machine learning models, development environment or data resource, on which the target machine learning model system depends.

Impact and Harm: Negates the integrity and availability of the targeted machine learning model. This attack may lead to the loss of integrity of training data, deployment platform, biased output, and a security breach.

[MP.T.3] A model manipulation attack involves an adversary directly altering the parameters, logic, or architecture of a machine learning model with the intent to compromise its performance, security, or integrity [27], [36]. This type of attack differs from traditional adversarial attacks that focus on crafting malicious input samples or poisoning training data [27]. Instead, the adversary gains access to the model itself and modifies it to achieve specific malicious goals [27].

An example of a specific risk is presented below from [27] source:

System Asset: machine learning model.

Business Asset: model's parameters.

Security Criteria: integrity of the model's parameters.

Vulnerability:

1. Lack of model integrity verification.
2. Unauthorized access to the model.

Threat Agent: white-box scenario.

Attack Method: A loss function is optimized, combining cross-entropy loss to misclassify target samples and weight regularization to minimize parameter deviations from the original model, ensuring that selected malicious samples are classified as benign. The final modifications are applied to the target ML model, and malicious samples are submitted.

Impact and Harm: Negates the integrity of the targeted machine learning model. This leads to a reduction in model's accuracy.

3.4 Related Work

7 academic papers were excluded from the data extraction process of attacks against machine learning models. Their focus is set on a wider coverage of attacks against an AI system. These papers either cover thread modeling or act as surveys of present attacks. This section presents a table of comparison (Table 2) between these papers and this thesis. The comparison is made on scope, methodology, framework used to organize or threat-model attacks, and final contribution.

The current thesis supplies a concrete threat model artifact, following an ISSRM domain model, focusing on LLM AI models. The domain model is synthesized based on the conducted literature review.

Table 2. Threat related research comparison.

Paper	Scope	Studied threats or security aspects	Methodology	Organizing framework	Contribution
[21]	Deep neural networks.	Evasion and poisoning attacks, defenses against the studied attacks.	Systematic review.	Five-stage “Advanced Persistent Threat” (APT) lifecycle for attacks and defenses.	An APT lifecycle mapping of studied attack families and emerging defenses.
[42]	An array of ML algorithms in the context of malware C&C detection.	Variants of evasion and poisoning attacks, existing defense studies.	Domain-specific review.	Attacks are grouped by their goal (poisoning, evasion) and affected algorithms (clustering, classifiers).	Organization of attacks according to the set framework, coverage of studied defenses, coverage of attack execution difficulty, coverage of C&C detection limitations.
[18]	Model type agnostic, focus on industry ML deployments.	The gap between academic attack studies and real attacks.	Systematic analysis of recent research paired with case studies based on real experience.	Not applicable.	Provides a set of observations regarding recent studies, provides a set of recommendations. Specifically, to set precise attacker viewpoints on ML components, to incorporate cost-driven assessments, to disclose the source code, to collaborate with practitioners, and to cover entire ML systems.
[36]	Deep neural networks in the context of face	Backdoor attacks and defenses.	Survey of literature on backdoor attacks and defenses previously demonstrated on FRS	Classification of backdoor attacks based on their attack channels, injection	A threat model of FRS pipelines, classification of FRS-oriented backdoor attacks, and defenses.

	recognition systems (FRS).		DNNs.	methods, and trigger specifications.	
[6]	ML usage in the context of Things (IoT).	Benefits, vulnerabilities, and weaponization of ML.	Domain-specific survey.	Categorization of ML in IoT following a tripartite framework. Good – benefits, bad – attacks and vulnerabilities of ML, ugly – malicious use of ML.	Coverage of positive and negative uses of ML.
[17]	Malicious use and abuse of AI in the context of cybercrime.	Elicitation of targeted AI model vulnerabilities and AI-enabled cybercrime activities.	Literature, report, and incident review focused on cybercrime and abuse of AI.	4 types of AI abuses, 4 AI malicious uses typology.	Interdisciplinary typology of AI abuse and malicious use and policy-oriented discussion of AI-assisted cybercrime.
[43]	General ML coverage.	AI assets and their failure modes.	Literature survey, use case application of the extended STRIDE framework.	AI-oriented extension of STRIDE.	Extension of the STRIDE framework to AI threats, modeling of threats to ML data assets.
This thesis	General ML and LLM systems.	24 attacks, targeted ML assets, proposed countermeasures.	Systematic literature review, design-science artifact in the form of a threat model, feasibility analysis of the produced threat model, and empirical test of one threat.	System and business asset-oriented threat model based on the ISSRM domain model.	Asset-oriented threat model tailored to LLMs.

3.5 Limitations

The current main limitation is the limited coverage of research materials. The current systematic literature review cannot be fully comprehensive, as not every possible academic paper repository was covered. The selection of papers based on their title and abstract may have introduced the author's bias into the selection procedure, possibly resulting in further quality assessment bias.

Moreover, at the current stage, data extraction bias could be present, as the operation has not followed any established standard procedure. The scope of SLR an indiscriminate extraction of data from the papers was conducted, but certain details could have still been omitted.

3.6 Answers to Research Questions

The current chapter will answer the set literature review question. The answers act as a summary of SLR findings, complementing the extracted data, presented in Appendices II and III. The SLR chapter also fulfilled the DSR stage – theory, by eliciting data for the subsequent artifact stage.

Main literature review question, [RQ1]: "What is the current state of the art of the threat modeling practice relative to LLM-based systems?". Paper selection process and analysis of the selected process showed that there is a gap in the academic body of knowledge, relative to threat modeling of LLM systems. A general absence of security-oriented frameworks for LLM-based systems has been observed. This thesis addresses this identified gap by threat modeling of identified attack types against an LLM system based on the reference domain model, defined in Chapter 2.2.2.

Sub-question question 1, [RQ1.1]: "What cybersecurity attacks against artificial intelligence are currently known?". The review of 38 peer-reviewed papers and gray literature identified 24 distinct types of attacks targeting AI/ML systems:

1. Poisoning attack;
    a. Poisoning attack: Backdoor attack;
    b. Poisoning attack: a Denial of Service attack;
2. Adversarial attack;
    a. Adversarial attack: Evasion attack;
    b. Adversarial attack: Man-in-the-Middle attack;
    c. Jailbreak attack;
3. Model inversion attack;
4. Model extraction attack;
    a. Model extraction attack: side-channel attack;
5. Membership inference attack;
    a. Membership inference attack: shadow training method;
    b. Membership inference attack: metric-based method;
6. Cyber-physical attack;
    a. Side-channel attack;
    b. Malicious hardware fault injection;
    c. Hardware Trojan attack;
7. Property inference attack;
8. Fingerprinting attack;
9. Denial of service attack;
10. Denial of wallet attack;
11. Embedding inversion attack;

12. Supply chain attack;
13. Model manipulation attack.

Based on the reviewed sources, while covered attacks were studied against different ML models from LLMs, the threats are still considered to be applicable to LLMs as well. As LLMs are a type of deep machine learning algorithm, they inherit the fundamental vulnerabilities targeted by the elicited attacks [2], [10], [21]. Attacks targeting other ML models exploit common weaknesses in learning and inference processes, making them transferrable to LLMs [21], [28], [40]. For example, an adversarial attack studied in the context of computer vision ML models [8], [21] applies to LLMs in the form of a jailbreak (prompt-injection) attack [1], [2], [10]. In addition, poisoning attacks [8], focused on malicious modification of the training data, apply to LLMs as well [1], [10].

The listed attacks can affect security criteria such as confidentiality, integrity, and availability of elicited components of the target AI/ML systems. The size of the list indicates an increased need for consideration of cybersecurity countermeasures and threat modeling frameworks to guide interested parties in risk assessment and countermeasure implementation.

Sub-question question 2, [RQ1.2]: "What cybersecurity threats currently exist specific to large language models?". From the reviewed literature, two unique attacks were elicited, research in the scope of LLM models – **Jailbreak** and **Embedding Inversion** attacks. The **Jailbreak attack** [2], [10], [23] (also known as Prompt Injection) is a specific type of attack on the LLMs. The goal of the attack is to form a malicious input that bypasses present safeguards and makes the model produce a desirable output or take malicious action. An **Embedding Inversion attack** [10], which exploits vulnerabilities to invert embeddings and recover input data information, thus compromising input data confidentiality.

Sub-question question 3, [RQ1.3]: "How to build a threat model, depicting existing cybersecurity threats to a large language model?". The paper selection procedure has not discovered research covering this topic. No peer-reviewed papers from the collected corpus offer a threat model of LLMs. The closest academic precedent [43] adapts the STRIDE threat modeling framework to AI data assets without coverage of LLMs. From gray literature, "OWASP Top 10 for LLM Applications 2025" [10] enumerates a set of attacks but does not map them to assets of an LLM-based system. The thesis fills this gap by designing an asset-oriented threat model based on the ISSRM domain model in the following Chapter 4.

4. Threat Modeling Large Language Model System

The goal of this chapter is to realize the main artifact output of the thesis and contribute to the answer to the main research question. This chapter fulfills the artifact stage of the DSR paradigm. An asset-oriented threat model is designed for an LLM system, based on the conducted systematic literature search (Chapter 3) and assembled corpus of evidence (Appendix II). The threat model follows the ISSRM domain model (Chapter 2.2.2). The following steps are undertaken within this chapter:

1. Definition of business and system assets and their relationship. Their combinations shall represent an LLM system at a high level.
2. Mapping of 24 threat variants, elicited through SRL, to the defined system assets. The mappings are grouped by the targeted business asset.
3. The produced threat-asset mapping pairs are connected to security requirements, which have an assigned list of security controls. The controls are listed in Appendix III for each threat separately.
4. An interactive web page is made to provide an overview of all the threat modeling-related information. The web page should provide convenient access and an overview of the synthesized data for threat modeling of LLM systems.

UML class and state, and BPMN process diagrams are designed to present synthesized knowledge on the inter-asset and threat relationships. The style of the UML diagrams follows the outlined standard for the "Unified Modeling Language"[1].

4.1 Asset Model

The relationships between the LLM system's constituent assets are depicted within the UML class diagram, Figure 4. The model portrays a high-level overview of the LLM-based system. The diagrams portray 7 business assets and 7 system assets, which are targeted by the attacks elicited from the conducted SLR. The core portrayed component is the "Machine learning model" system asset, which represents the LLM model itself. The ML model is supported by other surrounding systems assets, to enable its operation in the form of:

- input provisioning and preprocessing for inference operations;
- execution of the model, thus enabling its inference operations;
- model design and development;
- training and testing of the developed model.

Business assets are linked to the associations between system assets, indicating the operations on or transfer of the business assets between the system assets. In addition, two enumerations are defined – "Input data status" and "ML model state", indicating separate states of the input data processing and ML model development. Detailed asset and association definitions are provided within the following Chapter 4.2.

[1] Unified Modeling Language (UML) description, UML diagram examples, tutorials and reference for all types of UML diagrams - use case diagrams, class, package, component, composite structure diagrams, deployments, activities, interactions, profiles, etc.: https://www.uml-diagrams.org/

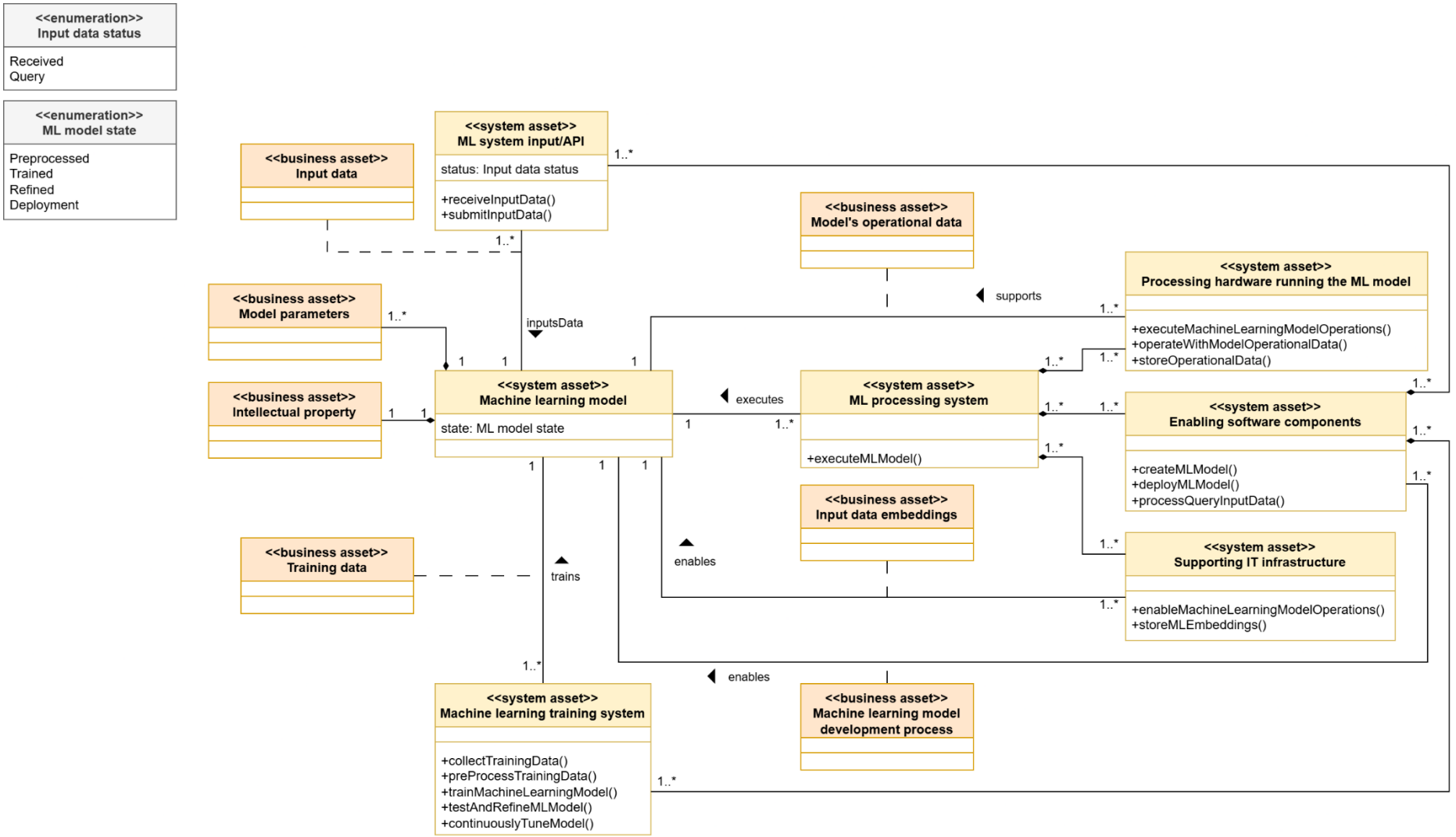

Figure 4. An LLM system asset relationship diagram.

In Figure 5, the input system asset has an attribute of **Input data status**, which can have a two possible values – **Received** and **Query**. The **Received** state represents the state of the input data when it was received before getting preprocessed with the **processQueryInputData()** method. After preprocessing, the state becomes – **Query**, representing the final input data state before its submission to the model for inference operations.

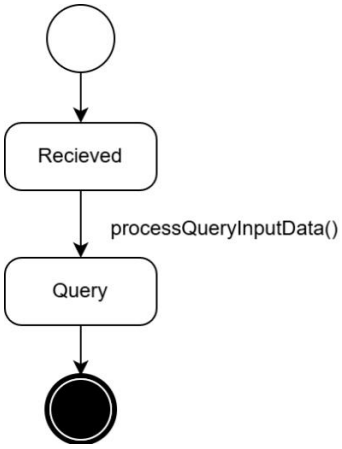

Figure 5. State change of the "input data status".

Figure 6 depicts value changes of the "ML model state", which is an enumeration that can contain a value from the predetermined set of values. The predetermined values are: **Preprocessed**, **Trained**, **Refined,** and **Deployment**, reflecting separate stages of the model development. Initially, the model is designed with a set architecture and a set of parameters, untrained – this is the **Preprocessed** state, an initial design, and an empty model. Afterwards, the **Preprocessed** model is trained on the target datasets, producing an intermediary state – **Trained**, when the model has adjusted parameters, but the training is not yet finalized. The model is trained, tested, and refined, producing a final version of the model suitable for use in a production environment. In this case, the model reaches the **Refined** state. Finally, the refined model is deployed to the target environment and is used for conducting necessary operations, this is the final **Deployed** state.

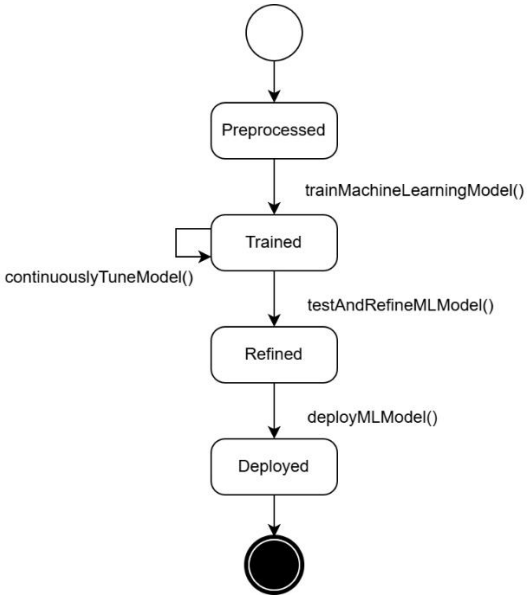

Figure 6. State change of the "ML model status".

4.2 Asset Definitions

Machine learning model: a software artifact produced from a training process. This algorithmic structure conducts inference or prediction operations based on the provided input data. The model processes the input data based on the learned rules, and captured patterns from training data [9], [18], [28], [44]. In the context of LLMs, this asset processes textual input data and produces text output, although other data mediums may be applicable as well. The "1" notation, applied to the associations at the model's side, indicates that the associated assets apply only to 1 specific model in 1 instance.

The model has a defined attribute "ML model state", which is an enumeration that can contain a value from the predetermined set of values. The values are – **Preprocessed**, **Trained**, **Refined,** and **Deployment**. This attribute describes the state of the target model as it is processed through the defined process pipelines. Initially, the model is designed with a set architecture and a set of parameters, untrained – this is the **Preprocessed** state, an initial design, and an empty model. Afterwards, the **Preprocessed** model is trained on the target datasets, producing an intermediary state – **Trained**, when the model has adjusted parameters, but the training is not yet finalized. The model is further trained, tested, and refined, producing a final model suitable for use in a production environment. In this case, the model is in the **Refined** state. Finally, in the **Deployed** state, the refined model is deployed to the target environment and is used for conducting inference operations.

Relationship with component business assets:

- **Model parameters:** integral components of a model, internal variables, critical to its operation and utility. In the context of this thesis, the following components are considered as model parameters: layers, type of activation function, layer connections, parameters, and weights [10], [21]. The values of parameters are adjusted during the training process. The "1..*" notation indicates that a model may have one or many sets of parameters. If the machine learning model is removed, the parameters are removed as well.
- **Intellectual property:** legal rights and protections, proprietary methods, training methodologies, curated training data, and designs associated with the produced machine learning model and its components [1], [10]. Intellectual property underlines company's competitive advantage, it can incorporate developed trade secrets and patents. The "1*" notation indicates that a model may "have one intellectual property".

ML system input/API: a combination of software and hardware systems, acting as an external interface to the target model. External users or client applications utilize the interface to submit textual input data to conduct inference operations on it and to receive the operation's results in response. An example implementation – a REST API or a web interface, is used for data submission [18], [20], [45]. The input system can implement additional processes for input data preprocessing or filtering, authentication, and access control.

The association between the input and the model depicts that there may be multiple inputs operating with one model, involving the input data in the process. The input pipeline preprocesses the received input before it is fed to the model for processing [1], [6], [18], [21]. The preprocessing may involve necessary transformations, tokenization, cleaning, filtering, normalization, feature extraction, or noise reduction for purposes of query processing optimization or as a part of security controls [6], [18]. This is depicted with the method **processQueryInputData()** under the **Enabling software components** system assets. The **ML system input/API** is the defined component, part of the **Enabling software components**.

Many different input implementations in operation act as many different software components, which are a part of the **ML processing system**.

The input system asset has an attribute of **Input data status**, which may have two possible values – **Received** and **Query**. The **Received** state represents the state of the input data when it was received and had not yet been preprocessed with the **processQueryInputData()** method. After preprocessing, the state becomes – **Query**, representing the final input data form before its submission to the model for inference.

For LLMs, the primary input is textual prompts, and queries [1], [10]. In multimodal setups, the LLM models process additional information mediums as their input, such as images, which could be provided in combination with the textual input [1], [2], [10].

The input has the following defined methods:

1. **receiveInputData()** – method representing the process of receiving the input from the setup input interfaces.
2. **submitInputData()** – after preprocessing the received input data, it is submitted for processing by the target model.

Involved business data within the association between the input and the model:

- **Input data:** real-time user-supplied, external textual data, which is ingested by the system and passed to the model for inference and analysis [44]. For the LLMs the input data primarily consists of textual queries, which are submitted to the model to produce a target response [1], [10]. However, with the development of multimodal models, the text may incorporate additional types of data medium, such as images.

Processing hardware running the ML model: a physical or virtual hardware platform, infrastructure executing the inference operations of the running trained model [14], [40]. Such a platform can consist of server clusters of CPUs, GPUs or specialized accelerators. GPUs became a cornerstone for processing machine learning operations, as their parallel architecture accelerates matrix multiplications and linear algebra calculations, underpinning machine learning algorithms [40]. As LLMs require substantial computational resources, LLMs are often trained with cloud-based servers fitted with GPU clusters [40]. There may be many instances "*" of the processing hardware asset, dedicated to the processing of one machine learning model.

The system asset has the following defined methods:

1. **executeMachineLearningModelOperations()** – a method, depicting the process of executing machine learning software operations for inference.
2. **operateWithModelOperationalData()** – based on the inference operations and conducted calculations – set up, update, and utilize the temporary model's operational data.
3. **storeOperationalData()** – store the setup model's operational data in RAM for further to enable further required inference operations.

The association between the system asset and the machine learning model operates with the following business asset:

- **Model's operational data:** transient and intermediate data and calculations generated and executed upon, during the model's operational, inference stage, like the target model's parameters [21]. The operational data is held in system memory during the model's operations, it may contain temporary results and sensitive information. In the

case of LLMs, the calculations for the preprocessed input query involve calculations across the model's layers, attention mechanisms, and processing of token embeddings [2].

Enabling software components: the machine learning model utilizes, relies on, and consists of software frameworks, libraries, and runtime environments necessary for its development, training, and operation. These components consist of the necessary tools for building, training, evaluating, managing, and deploying machine learning models. Example software frameworks: TensorFlow, Keras, and PyTorch [1], [46]. For LLMs, the Hugging Face Transforms resource provides a collection of pre-trained models, tokenizers, and other components utilized for LLM development [1]. There can be many software components "*", which are utilized for the development and maintenance of one model.

Defined methods of the system asset:

1. **createMLModel()** – a method representing the process of building the initial **Preprocessed** machine learning model.
2. **deployMLModel()** – method defining a process of deployment and start of utilization of the **Deployed**, trained, and tuned model.
3. **processQueryInputData() –** after the model's input ingests the user's input data through intermediary APIs and services, this method preprocesses it. The preprocessing may involve necessary tokenization, filtering, and normalization to ensure the desired quality of models' inferences [6], [18].

Involved business asset within the association with the machine learning model:

- **Machine learning model development process**: a systematic, continuous process of machine learning development, incorporating software components together to create a model, enabling its training, and conducting inference operations. The goal is to build a machine learning model that will be able to solve the target business-specific problem to achieve a target objective. The process may involve activities like data collection, data preprocessing, model design, building and training, quality assurance, deployment, monitoring, feedback, and auditability [6], [13].

Supporting IT infrastructure: a broader computational environment, consisting of servers, networking, database services, cloud services, and applications [40]. The IT infrastructure enables auxiliary operations needed for the operation of the target machine learning model. In the portrayed case, it enables the storage and provisioning of input data embeddings for inference operations and machine learning operations [1]. As LLMs are resource-demanding models, the infrastructure may be structured to enable high-performance networking and scalable storage solutions. This will enable swift communication between computing and storage resources and will accommodate large storage and operations with large datasets. There can be multiple IT infrastructure components "*", which enable the operation of one machine learning model.

Defined asset methods:

1. **enableMachineLearningModelOperations()** – the IT infrastructure accommodates the communication between computing resources to enable the execution of processes, including inference operations of the machine learning model.
2. **storeMLEmbeddings()** – storage of input data embeddings, and business assets for later additional processing.

Involved business assets in association with the system asset and machine learning model:

- **Input data embeddings**: numerical vector representation of the input textual data, created either during the training or inference processes [19]. This representation allows machine learning models, including LLMs, to conduct defined calculations on the data, determine semantic relationships, and draw an inference [19]. These embeddings can be stored with the supporting IT infrastructure for later re-use. The stored embedding may be utilized at later stages for model fine-tuning or enhancement of the context of new input queries [10], [19].

Machine learning training system: a combination of processes and procedures, hardware and software infrastructure, which is dedicated to the training, re-training, fine-tuning, and creation of the machine learning model, updating the model's parameters in the process [18], [21]. The system may involve training data preprocessing, its collection, configuration of optimization algorithms, and evaluation of the trained model's performance against testing datasets [19], [22]. There may be multiple "*" training systems, dedicated to training one model.

Defined methods:

1. **collectTrainingData()** – collection of the training data, suitable for the machine learning model. The data may be taken from the prepared public data sets, collected through web resources with web crawlers, or from internal services.
2. **preProcessTrainingData()** – the training data is filtered and normalized to fit the training of the model, according to the model's design.
3. **trainMachineLearningModel()** – the process of training the model.
4. **testAndRefineMLModel()** – the process of testing intermediary model iterations and continued training.
5. **continuouslyTuneModel()** – the new iterations of training of finalized models to incorporate the new data and model design changes.

Business asset, related to the association with the machine learning model:

- **Training data**: datasets utilized to train, re-train, or fine-tune the target machine learning model [19]. In the context of LLMs, this could be a large collection of textual data. Utilized by the training process to train the model. LLMs can be trained in a two-stage process. Initially, the model is pre-trained on the general-purpose datasets [19]. Afterward, the model is fine-tuned on specific datasets, fitting the model's purpose.

ML processing system: a higher-level representation of a collection of systems and processes, utilized for the operation of the deployed machine learning model. The system contains the "processing hardware running the ML model", and "enabling software components", "supporting IT infrastructure" system assets as its components. If the system is removed, all the components are removed as well, as they are all necessary for the operation of the target model. There may be many "*" instances of the ML processing systems, containing multiple instances of its components.

4.3 Asset Security Criteria Definitions

Below is provided the meaning for outcomes from the compromise of a security criterion of a business asset and the machine learning model itself, which is depicted as the system asset. Impacts from compromise of "**Machine learning model**" system asset's CIA security criteria:

- **Confidentiality:** theft of the model's parameters or architecture design can leak intellectual property and trade secrets, enabling competitors or adversaries to replicate or misuse the designed ML model.

- **Integrity:** malicious unauthorized modification of the model's weights or logic can result in incorrect or harmful outputs. This integrity compromise breaks the trust in the model's reliability and can have cascading effects on dependent systems and clients.
- **Availability:** overloading or shut off of the environment, executing the ML model makes it inaccessible to clients. An unavailable model halts all dependent systems, services, client usage or automated decision processes, leading to business disruption.

Impacts from compromise of "**Model parameters**" business asset's CIA security criteria:

- **Confidentiality:** exfiltration of model parameters (or weights) may expose trade secrets and the acquired "knowledge", which was costly to acquire through the ML training process. Competitors or malicious actors may develop derivatives models or misuse the original model.
- **Integrity:** modification of model's parameters or injection of malicious updates into the parameter set may result it inaccurate or biased inference process outputs. This unauthorized modification may lead to unpredictable and harmful output.
- **Availability:** prevention of legitimate access to the model's weights, for example by encrypting or corrupting parameter files, may halt inference or retraining. The affected ML model becomes inoperable without these components, disrupting all dependent systems.

Impacts from compromise of "**Intellectual property**" business asset's CIA security criteria:

- **Confidentiality:** unauthorized disclosure of proprietary algorithm designs, or unique data sets may reduce competitive advantage. Leaked IP may get acquired by competitors, causing financial and reputational damage.
- **Integrity:** unauthorized modification of design documents that represent the IP can mislead the R&D process, thus hampering company's innovation.
- **Availability:** prevention of access to IP information may stall internal model development or system updates. Lack of IP availability may lead to project halting, causing potential financial losses.

Impacts from compromise of "**Training data**" business asset's CIA security criteria:

- **Confidentiality:** exposure of training datasets containing personal data may violate privacy regulations and result legal repercussions.
- **Integrity:** training data poisoning or unauthorized modification may lead to degradation of the model's utility. In addition, model's adjusted behavior may result in faulty decisions, thus negatively impacting dependent systems.
- **Availability:** prevention of access to the training data, stops the model's training process. Without training, the model's utility may reduce overtime.

Impacts from compromise of "**Input data**" business asset's CIA security criteria:

- **Confidentiality:** unauthorized access to clients' input data may reveal personal data or business secrets. This can result in a data privacy breach leading to legal repercussions.
- **Integrity:** unauthorized modification of the input data or injection of adversarial samples can result in generation of incorrect or harmful output data. Modified output may lead to incorrect operations, impacting depended systems.
- **Availability:** prevention of access to the input data prevents execution of inferences. Clients will be unable to receive an output data to their query; this may impact dependent systems and result in reduced confidence in the provided service.

Impacts from compromise of "**Model's operational data**" business asset's CIA security criteria:

- **Confidentiality:** unauthorized access to the model's operational data may result in leakage of transient operational data, containing personal data or business secrets. This may lead to data privacy breach, leading to legal repercussions.
- **Integrity:** unauthorized modification of the operation data may lead to generation of corrupted output data. This integrity compromise may result in the integrity compromise of systems dependent on the model's output data.
- **Availability:** unavailability of hardware resources to store and operate on the model's operational data may lead to the service's unavailability. This compromise may impact dependent systems.

Impacts from compromise of "**Input data embeddings**" business asset's CIA security criteria:

- **Confidentiality:** unauthorized access to the input data embeddings can lead to exposure of the sensitive data, which was produced from the original input data. This may lead to legal repercussions.
- **Integrity:** tampering with the integrity of the input data embeddings can result in incorrect inference results due to operations on the tampered embeddings. Alternatively, if the embeddings are utilized in model fine-tuning, the re-trained model can become less accurate due to re-training with the tampered embeddings. This may lead to generation of corrupt output data, causing faulty operation of dependent systems.
- **Availability:** prevention of access to the embeddings may prevent a machine learning model from conducting inference operations on the related input query. In addition, in case of re-training with the embeddings, the re-training can be conducted with incomplete datasets, resulting in a less accurate model.

Impacts from compromise of "**Machine learning model development process**" business asset's CIA security criteria are as follows:

- **Confidentiality:** unauthorized access to the development process can lead to the theft of the training data, model design, and intellectual property of the machine learning system. This may lead to the loss of competitive advantage and data privacy breach, thus leading to legal repercussions.
- **Integrity:** tampering with the development tools and processes can lead to the production of faulty models, failing to meet accuracy and utility goals.
- **Availability:** prevention of access to the development tools and processes can prevent model building, training, fine-tuning, and deployment.

4.4 Threats and Business Asset Relations

Within this chapter, multiple threat models are determined. Depicting relations between found threats and separate business assets, which are targeted for initial compromise. The compromise of any of the business assets leads to a further compromise of the targeted machine learning model. Each threat has a "targetSystemAsset()" method, which indicates a connection to the system assets, vulnerabilities of which they exploit, to be able to compromise the business asset's security criteria.

The models are based on the original model connecting the assets in Figure 4. The asset model is partitioned into distinct subsets of assets to depict the threats more effectively. The models provide a high-level overview. Specific vulnerabilities, attack methods, and security measures

related to a particular threat can be found in Appendix III. The combination of models and textual information from the appendix is intended to help machine learning developers and security analysts get a holistic overview of the existing threats. As well as threats' mapping to business and system assets, and applicable security controls.

An asset-oriented approach is taken to depict a clear picture of existing threats and the targeted assets. This should allow for conducting further risk assessments only with the threats applicable to the present assets, requiring additional protection. In combination with the information from Appendix III, an organization could also consider researched security controls to tackle its elicited security requirements.

4.4.1 Threats and Training Data Relations

Figure 7 visualizes a threat model with 7 threats, elicited from the literature review, which targets the training data for the initial compromise. In the current subset of assets, "ML system input/API" supports the "Machine learning model" by passing the user's input data for inference processing. "Machine learning training system" is responsible for training the target "Machine learning model". The presented threats conduct the compromise of the target model through corruption and compromise of the training data's security criteria.

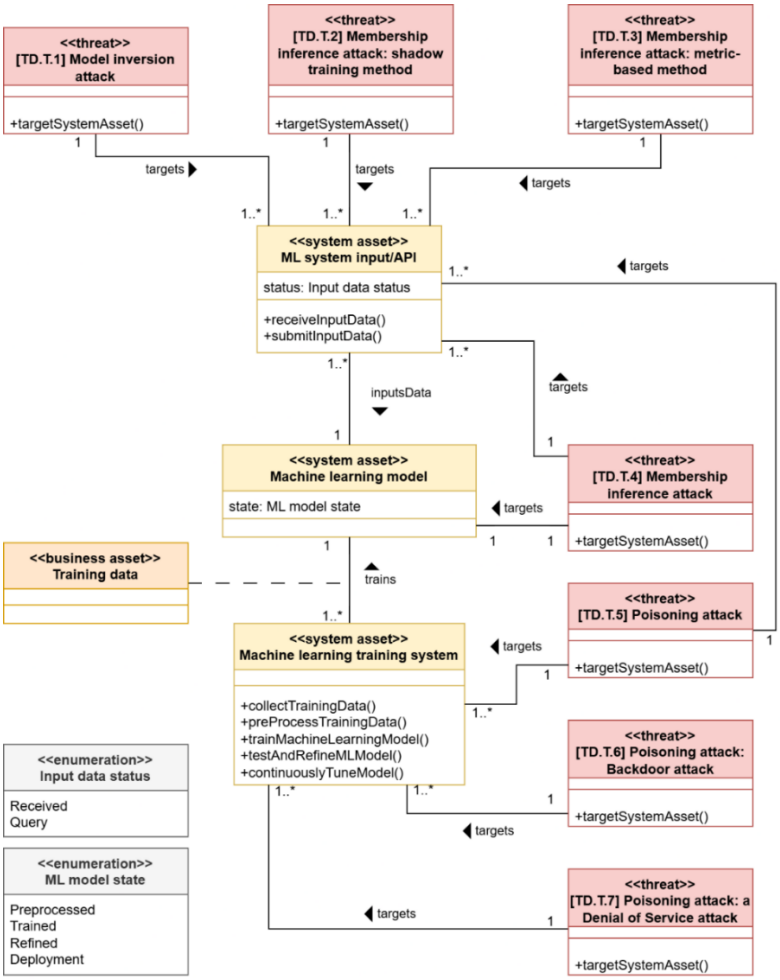

Figure 7. Relationship of attacks and training data asset.

Table 3 lists seven threats focused on the compromise of the training data business asset. For each threat, the table maps, compromised CIA security criteria of the targeted business asset, targeted system assets, and a security requirement. Each threat has its code, consisting of the

targeted business asset name's letters; type of entry, **T** – for threat, **SR** – for security requirement, and a unique number for each entry. The coding scheme facilitates cross-reference to the threat definitions in Chapter 3 and Appendix III.

Table 3. Mapping between attacks and security requirements for the threats targeting the training data.

Threat	System asset	Security criteria	Security requirement
[TD.T.1] Model inversion attack	ML system input/API	Confidentiality	[MLSI.SR.1] Resist model inversion attack
[TD.T.2] Membership inference attack: shadow training method	ML system input/API	Confidentiality	[MLSI.SR.9] Resist membership inference attack
[TD.T.3] Membership inference attack: metric-based method	ML system input/API	Confidentiality	[MLSI.SR.9] Resist membership inference attack
[TD.T.4] Membership inference attack	ML system input/API; Machine learning model	Confidentiality	[MLSI.SR.9] Resist membership inference attack
[TD.T.5] Poisoning attack	Machine learning training system; Machine learning model	Integrity, availability	[MLTS.SR.1] Resist poisoning attack
[TD.T.6] Poisoning attack: Backdoor attack	Machine learning training system	Integrity	[MLTS.SR.2] Resist poisoning attack: backdoor attack
[TD.T.7] Poisoning attack: a Denial of Service attack	Machine learning training system	Availability	[MLTS.SR.3] Resist poisoning attack: denial of service

4.4.2 Threats and Input Data Relations

Figure 8 visualizes a threat model with 13 threats, determined from the conducted systematic literature review, which targets the input data for the initial compromise. The compromise of the training data is conducted either through "ML system input/API", "Processing hardware running the ML model" system assets or by targeting the "Machine learning model" itself. "ML system input/API" asset facilitates the transfer of input data for inference operations. "Processing hardware running the ML model" asset executes the model, enabling the inference operations on the submitted input data.

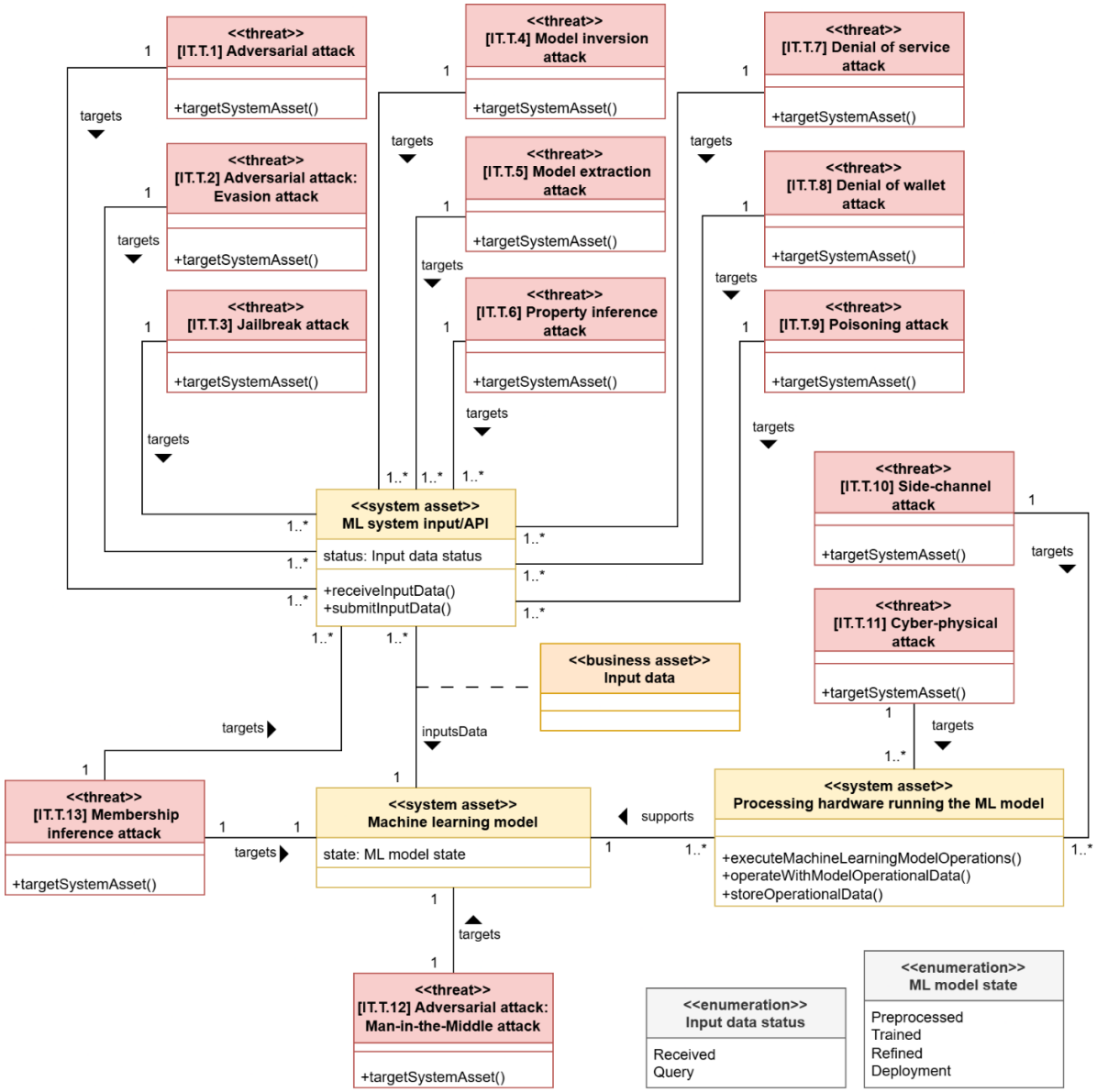

Figure 8. Relationship of attacks and input data asset.

Table 4 aggregates thirteen distinct attacks exploiting the input interface, the underlying hardware, or the model itself. Each row presents a mapping between threats, compromised security criteria of the input data business asset, targeted system assets, and a prescribed security requirement.

Table 4. Mapping between attacks and security requirements for the threats targeting input data.

Threat	System asset	Security criteria	Security requirement
[IT.T.1] Adversarial attack	ML system input/API	Integrity	[MLSI.SR.4] Resist adversarial attack
[IT.T.2] Adversarial attack: Evasion attack	ML system input/API	Integrity, availability	[MLSI.SR.10] Resist adversarial evasion attack

[IT.T.3] Jailbreak attack	ML system input/API	Confidentiality, integrity	[MLSI.SR.3] Resist jailbreak attack
[IT.T.4] Model inversion attack	ML system input/API	Confidentiality	[MLSI.SR.1] Resist model inversion attack
[IT.T.5] Model extraction attack	ML system input/API	Confidentiality	[MLSI.SR.8] Resist model extraction attack
[IT.T.6] Property inference attack	ML system input/API	Confidentiality	[MLSI.SR.2] Resist property inference attack
[IT.T.7] Denial of service attack	ML system input/API	Availability	[MLSI.SR.6] Resist denial of service attack
[IT.T.8] Denial of wallet attack	ML system input/API	Availability	[MLSI.SR.7] Resist denial of wallet attack
[IT.T.9] Poisoning attack	ML system input/API	Integrity, availability	[MLSI.SR.5] Resist poisoning attack
[IT.T.10] Side-channel attack	Processing hardware running the ML model	Confidentiality	[PH.SR.4] Resist side-channel attack
[IT.T.11] Cyber-physical attack	Processing hardware running the ML model	Availability	[PH.SR.1] Resist cyber-physical attack
[IT.T.12] Adversarial attack: Man-in-the-Middle	Machine learning model	Integrity	[MLM.SR.1] Resist adversarial attack: Man-in-the-Middle attack
[IT.T.13] Membership inference attack	ML system input/API; Machine learning model	Confidentiality	[MLSI.SR.9], [MLM.SR.2] Resist membership inference attack

4.4.3 Threats and Model Parameters Relations

Figure 9 isolates 3 threats, determined from the conducted systematic literature review, which target the model's parameters business asset for the initial compromise. The compromise of the model's parameters is conducted either through "Processing hardware running the ML model" or "Machine learning model" system assets. "Processing hardware running the ML model" executes the model's operations, holding the parameters in memory, and making them susceptible to side-channel attacks. Table 5 presents a mapping between the 3 elicited threats, targeted security criteria of the model parameter business asset, targeted system assets, and security requirements. The table catalogs model manipulation, side-channel, and model extraction: side-channel attacks. The attacks are focused on the business asset's and model's confidentiality or integrity compromise.

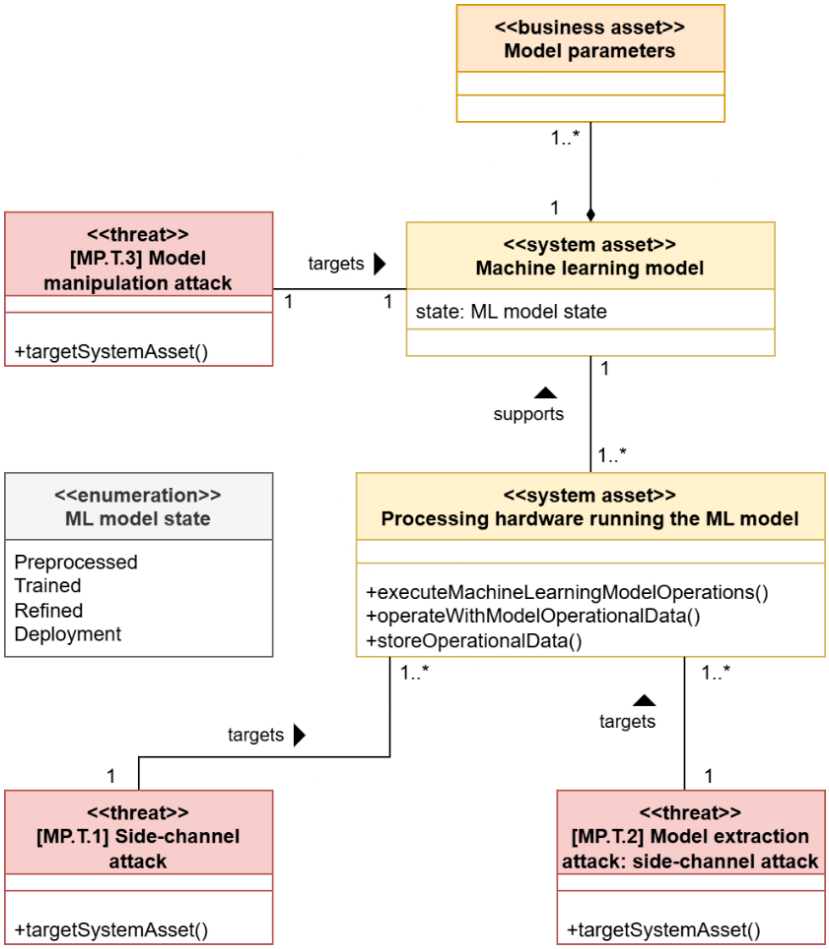

Figure 9. Relationship of attacks and model parameters asset.

Table 5. Mapping between attacks and security requirements for the threats targeting model parameters.

Threat	System asset	Security criteria	Security requirement
[MP.T.1] Side-channel attack	Processing hardware running the ML model	Confidentiality	[PH.SR.4] Resist side-channel attack
[MP.T.2] Model extraction attack: side-channel attack	Model extraction attack: side-channel attack	Confidentiality	[PH.SR.3] Resist model extraction: side-channel attack
[MP.T.3] Model manipulation attack	Machine learning model	Integrity	[MLM.SR.3] Resist model manipulation attack

4.4.4 Threats and Model's Operational Data Relations

Figure 10 visualizes 2 threats, determined from the conducted systematic literature review, which target the model's operational data business asset for the initial compromise. The compromise of the model's operational data is conducted either through "Processing hardware running the ML model" or "Machine learning model" system assets. Table 6 presents the hardware trojan and malicious fault injection attacks, targeted security criteria of the business asset, and prescribed security requirements.

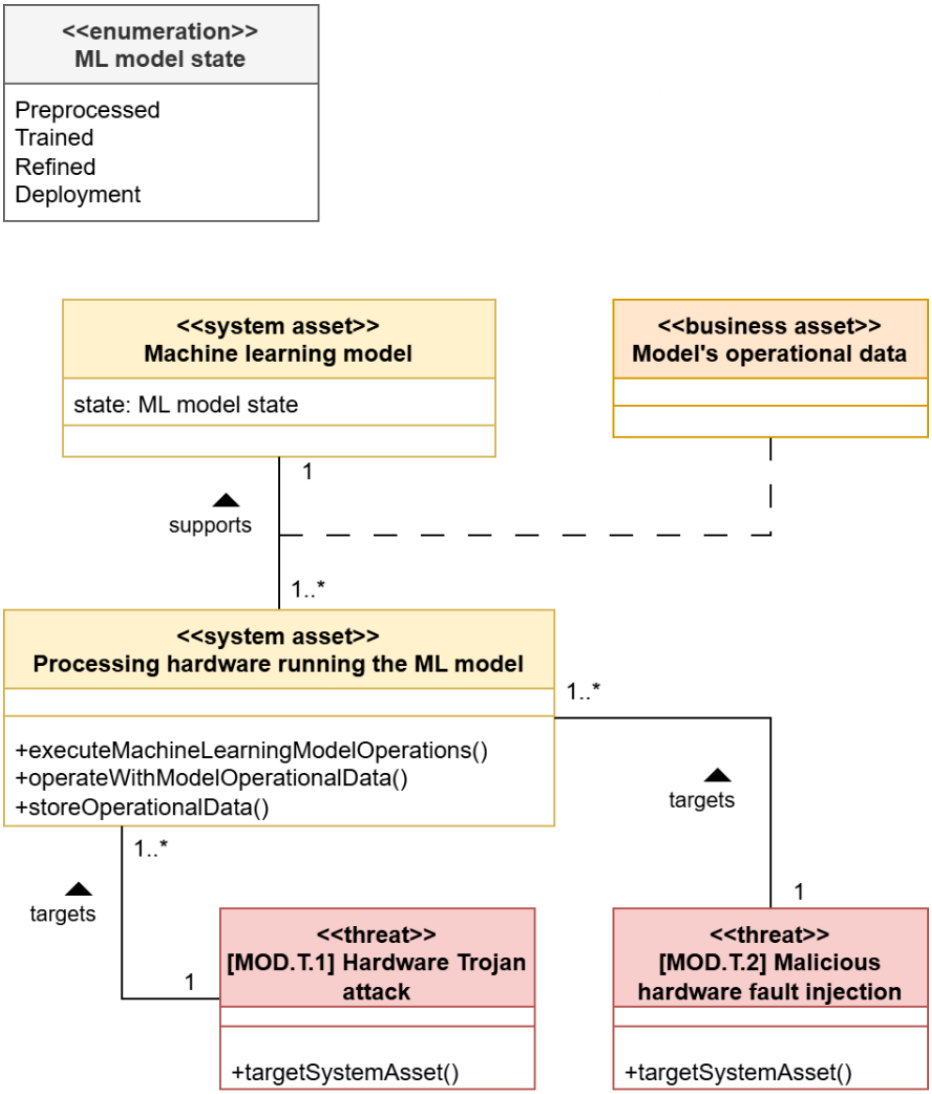

Figure 10. Relationship of attacks and model's operational data asset.

Table 6. Mapping between attacks and security requirements for the threats targeting model's operational data.

Threat	System asset	Security criteria	Security requirement
[MOD.T.1] Hardware Trojan attack	Processing hardware running the ML model	Confidentiality, integrity, availability	[PH.SR.2] Resist hardware trojan attack
[MOD.T.2] Malicious hardware fault injection	Processing hardware running the ML model	Integrity	[PH.SR.5] Resist malicious fault injection attack

4.4.5 Threat and Intellectual Property Relation

Figure 11 visualizes 1 threat, determined from the conducted systematic literature review, which targets the intellectual property business asset for the initial compromise. The compromise of the intellectual property is conducted through the "ML system input/API" system asset. Table 7 presents a mapping for the fingerprinting threat, which compromises the intellectual property of the machine learning model by querying the input interface.

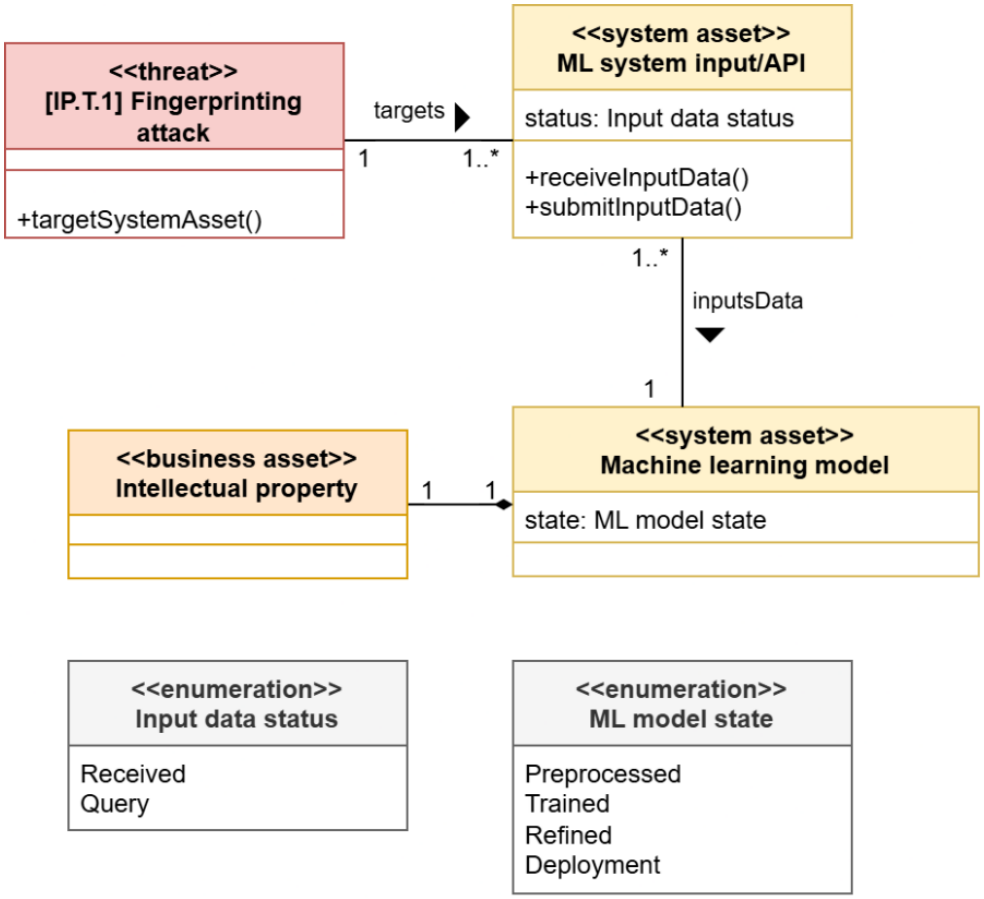

Figure 11. Relationship of attacks and intellectual property asset.

Table 7. Mapping between attacks and security requirements for the threats targeting intellectual property.

Threat	System asset	Security criteria	Security requirement
[IP.T.1] Fingerprinting attack	ML system input/API	Confidentiality	[MLSI.SR.11] Resist fingerprinting attack

4.4.6 Threat and Input Data Embeddings Relation

Figure 12 visualizes 1 threat, elicited from the conducted systematic literature review, which targets the input data embeddings business asset for the initial compromise. The compromise of the input data embeddings is conducted through the exploitation of the "Supporting IT infrastructure" system asset. Table 8 presents a mapping for the embedding inversion attack, which compromises the confidentiality of the input data embeddings business asset.

Table 8. Mapping between attacks and security requirements for the threats targeting input data embeddings.

Threat	System asset	Security criteria	Security requirement
[IDE.T.1] Embedding Inversion attacks	Supporting IT infrastructure	Confidentiality	[SITI.SR.1] Resist embedding inversion attack

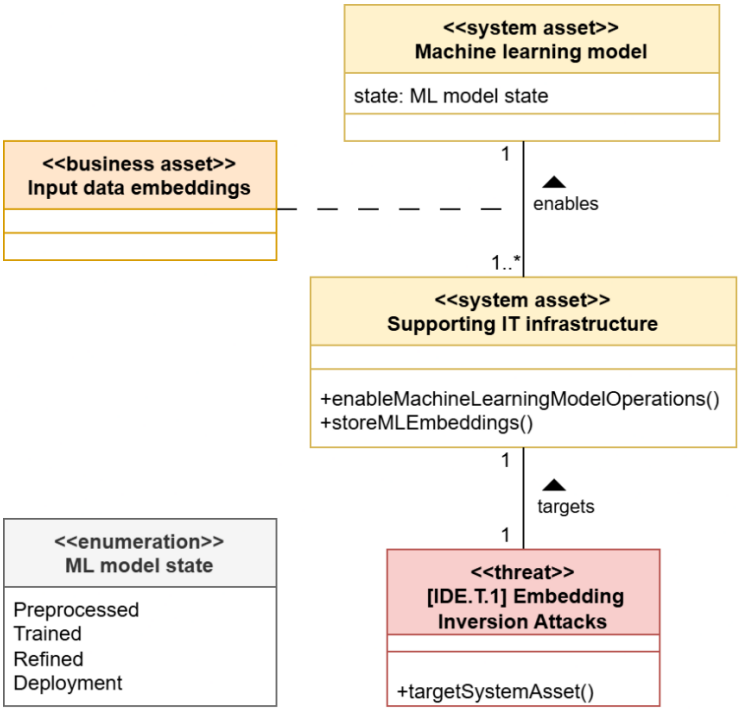

Figure 12. Relationship of attacks and input data embeddings asset.

4.4.7 Threat and Machine Learning Model Development Process Relation

Figure 13 visualizes 1 threat, elicited from the conducted systematic literature review, which targets the machine learning development process business asset for the initial compromise. The compromise of the intellectual property is conducted through the "Enabling software components" system asset.

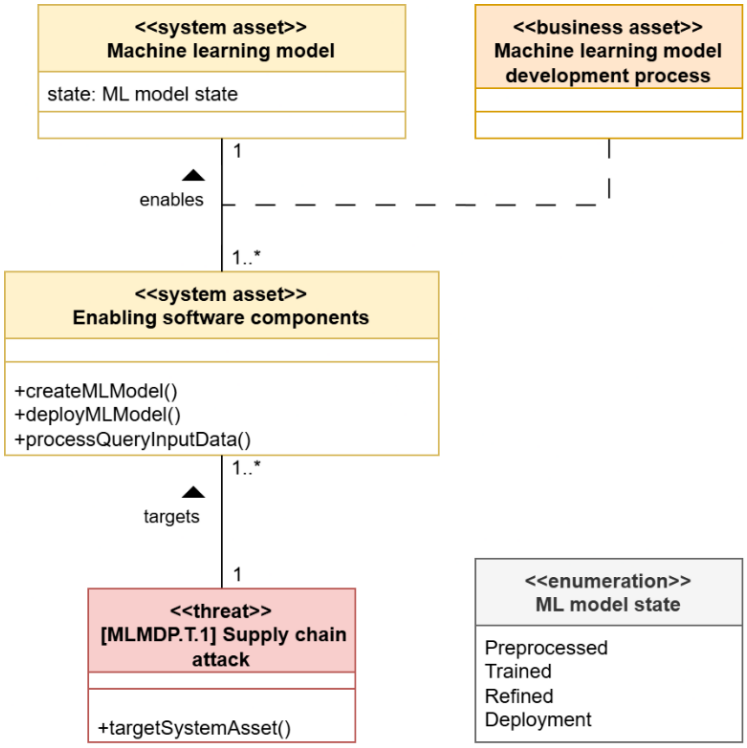

Figure 13. Relationship of attacks and machine learning model development process asset.

Table 9 presents a mapping for the supply chain threat, which targets business assets' integrity and availability criteria. The threat undermines vulnerabilities within the software components used for the model's development, thus affecting the security criteria of the development process.

Table 9. Mapping between attacks and security requirements for the threats targeting machine learning model development process.

Threat	System asset	Security criteria	Security requirement
[MLMDP.T.1] Supply chain attack	Enabling software components	Integrity, availability	[ESC.SR.1] Resist supply chain attack

4.5 Model-Related Processes

The chapter contains 3 BPMN process diagrams, depicting 3 main AI lifecycle processes: training, inference, and fine-tuning. The depiction of the processes is constructed around the system assets, defined within the main asset UML diagram in Figure 4.

Together, the elements of the BPMN diagrams depict **how** an organization starts training, inference, or fine-tuning pipeline. And how the organization conducts operations with each related system asset, and manages the creation of a final, deployed model, inference, and fine-tuning operations. In addition, the diagrams present a set of applicable security requirements for the system assets. In combination with threat diagrams and the lists of security controls from Appendix III, an organization can determine applicable threats to its processes and security controls, which could mitigate the risk.

4.5.1 Training Process

In this sub-chapter, the diagram (

Figure 14) depicts the training process of a machine learning model. There are 2 main BPMN swim lanes, depicting:

1. "**Enabling software components**" (middle lane) – represents the corresponding asset and the processes, which are undertaken with the asset. The components are deemed to be an irreplaceable part of conducting machine learning-related operations, such as training, development, building, and evaluation. The components may include software frameworks, libraries, and runtime environments necessary for model training and development.
2. "**Machine learning training system**" (lower lane) – stands for the collection of tools, services, activities, and processes responsible for the training of a machine learning model.

An auxiliary top lane represents an engineer, who jump-starts the training process. The lane does not correspond to any assets within the UML asset diagram (Figure 4); it is added to depict the starting point of the training process.

Overall, an engineer initiates the training process of a machine learning model. Through enabling software components, a preprocessed ML model is created, with untrained parameters based on the internally developed design. The internal design is represented by the "intellectual property". The preprocessed model is transferred to the training system, which has prepared the training data for the training process, including testing data. The testing data plays a key

role in testing and refining the model after a training iteration. After training, a trained ML model is produced, and it gets deployed after passing defined tests.

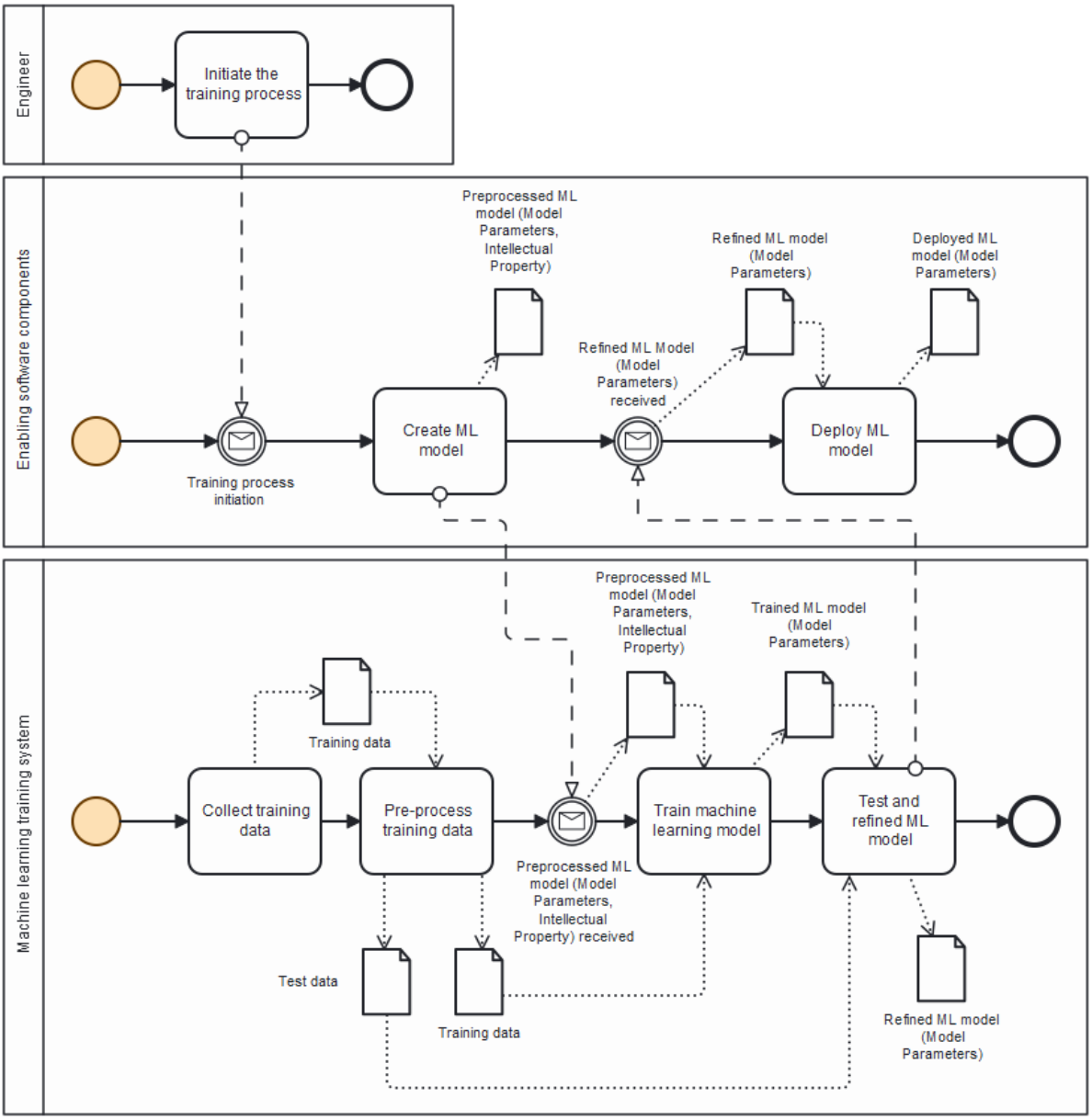

Figure 14. Business process modeling of the training process for an ML model.

4.5.2 Inference Process

The current chapter provides a BPMN diagram (Figure 15), depicting the inference process of a machine learning model. The process involves 4 system assets.

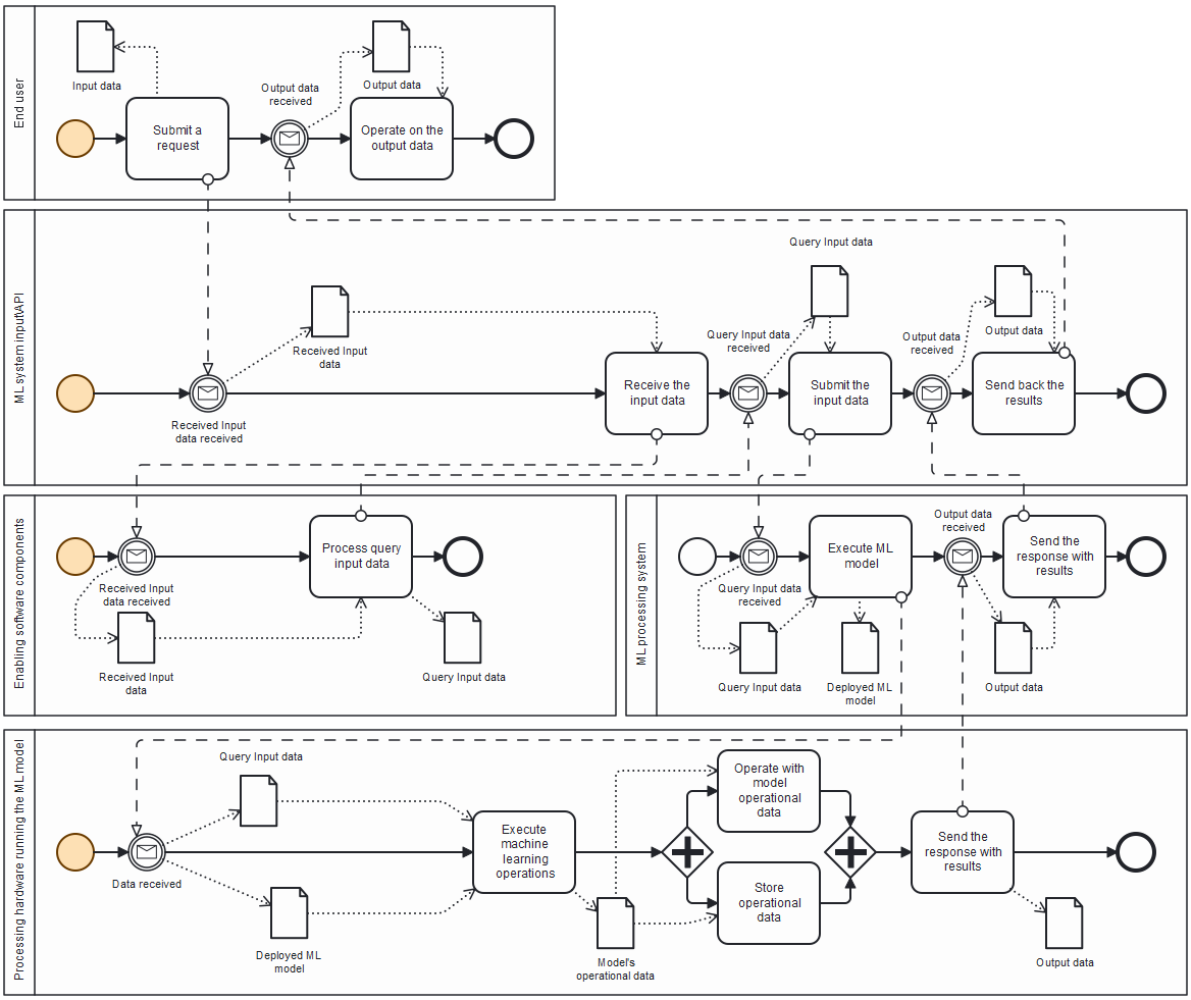

Figure 15. Business process modeling of the inference process for an ML model.

There are 4 main BPMN swim lanes, depicting:

1. "**ML system input/API**" (2nd lane) – the system asset, representing an interface, which receives user input, and transfers it for preprocessing, and for inference to the model.
2. "**Enabling software components**" (left middle lane) – represents the corresponding asset and the processes, which are undertaken with the asset. The components may include software frameworks, libraries, and runtime environments, utilized for model building, training, and development.
3. "**ML processing system**" (right middle lane) - the high-level representation of the LLM system, which interconnects separate system assets, dedicated to the building, training, operating, and maintenance of the machine learning model.
4. "**Processing hardware running the ML model**" (bottom lane) – hardware, specialized accelerator systems, which execute the ML model's inference operations.

An auxiliary top lane represents an end user who jump-starts the training process. The lane does not correspond to any assets within the UML diagram; it is added to depict the starting point of the inference process, with the user submitting input data.

An end user starts the inference process of a machine learning model by submitting a query with input data through a set interface. Through enabling software components, the initial input query is preprocessed and returned to the input system. The input system transfers the preprocessed input data query to the ML processing system, which starts the execution of the deployed model. The ML processing system passes the deployed model and the preprocessed input query to the hardware processing system. The hardware processing system conducts the execution of required calculations with the deployed model against the preprocessed input query. In between execution cycles, operational data is produced and stored. Finally, the inferences (output data) are produced, and they are returned through the system assets back to the end user. The end user conducts further operations on the received inferences and output data. Output data and the tasks related to its retrieval are auxiliary, depicted to form a complete loop; the data and tasks do not correspond to any business assets or method, defined in the main asset model in Figure 4.

4.5.3 Fine-Tuning Process

The current chapter provides a diagram (Figure 16), depicting the fine-tuning process of a machine learning model. There are 3 main BPMN swim lanes, depicting:

1. "**Machine learning training system**" (top lane) – the system asset standing for the collection of tools, services, and processes responsible for the training of a machine learning model.
2. "**ML system input/API**" (middle lane) – the system asset, representing an interface, which receives user input, and transfers it for preprocessing, and for inference to the model.
3. "**Enabling software components**" (bottom lane) - represents the corresponding system asset and the processes, which are undertaken with the asset. The components may include software frameworks, libraries, and runtime environments, utilized for model building, training, and development.

The diagram represents the possible variation of fine-tuning, which takes a preprocessed input query and utilizes it to update and re-train the model. The process starts with the submission of the input data through the input system. The input system exchanges input data with the enabling software components for the input data preprocessing and model refinement. The enabling software components pass the preprocessed input data to the machine learning training system. The training system re-trains, tunes the machine learning model, and returns it to the enabling components for deployment.

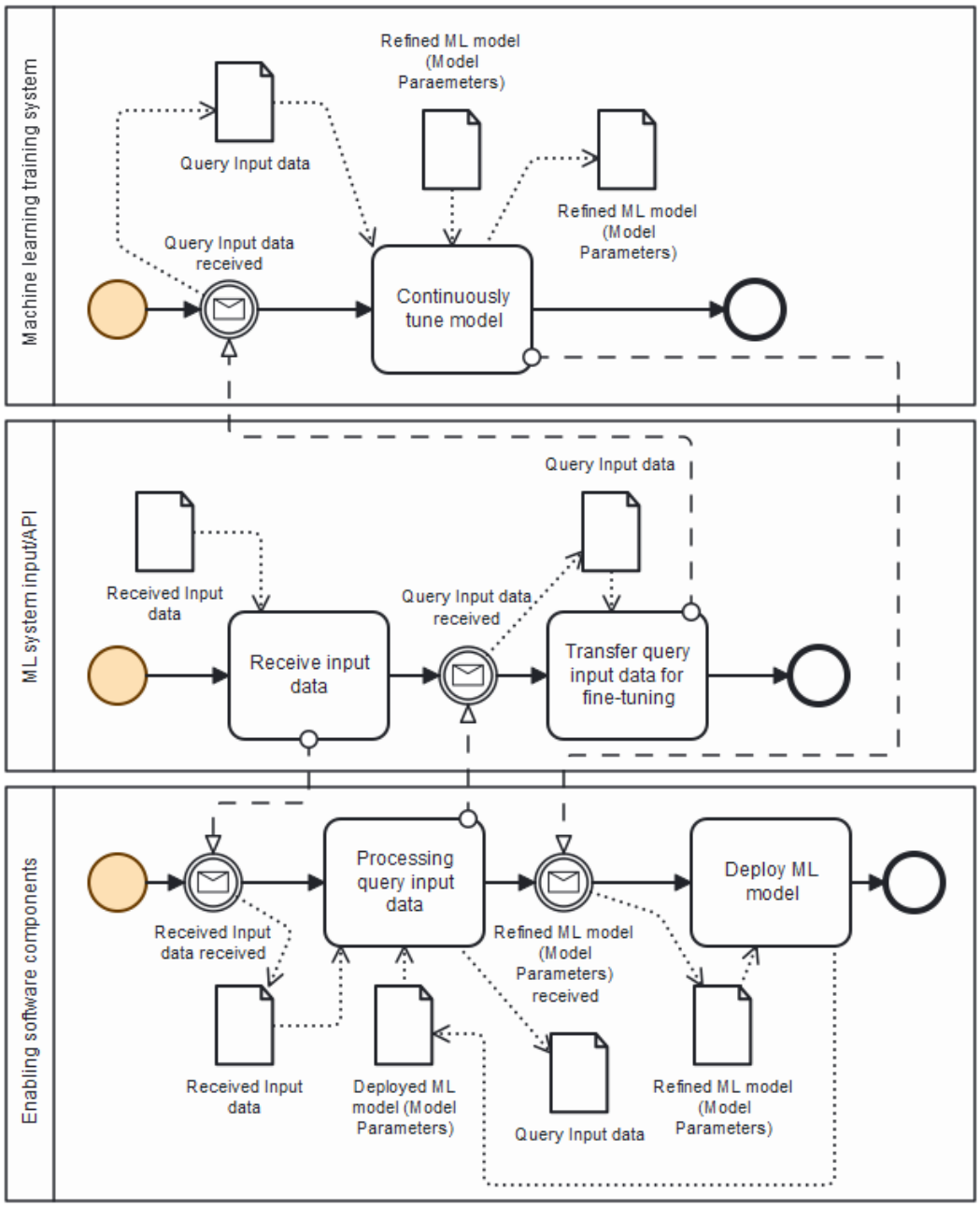

Figure 16. Business process modeling of the fine-tuning process for an ML model.

4.5.4 Security Requirements for System Assets

The current chapter provides a BPMN diagram (Figure 17), depicting the mapping of security requirements to each of the system assets, which took part in the preceding BPMN diagrams. The diagrams indicate that all the security requirements are independent; the implementation of a separate requirement will depend on the needs of a target system. Implementation of security controls, corresponding to the security requirement will mitigate a corresponding threat. A user of an LLM system, after conducting a risk assessment, and eliciting a set of likely applicable threats, could select a subset of security requirements for each system asset. The security requirements selection will enable the user to select a security control from the lists of controls present in Appendix III.

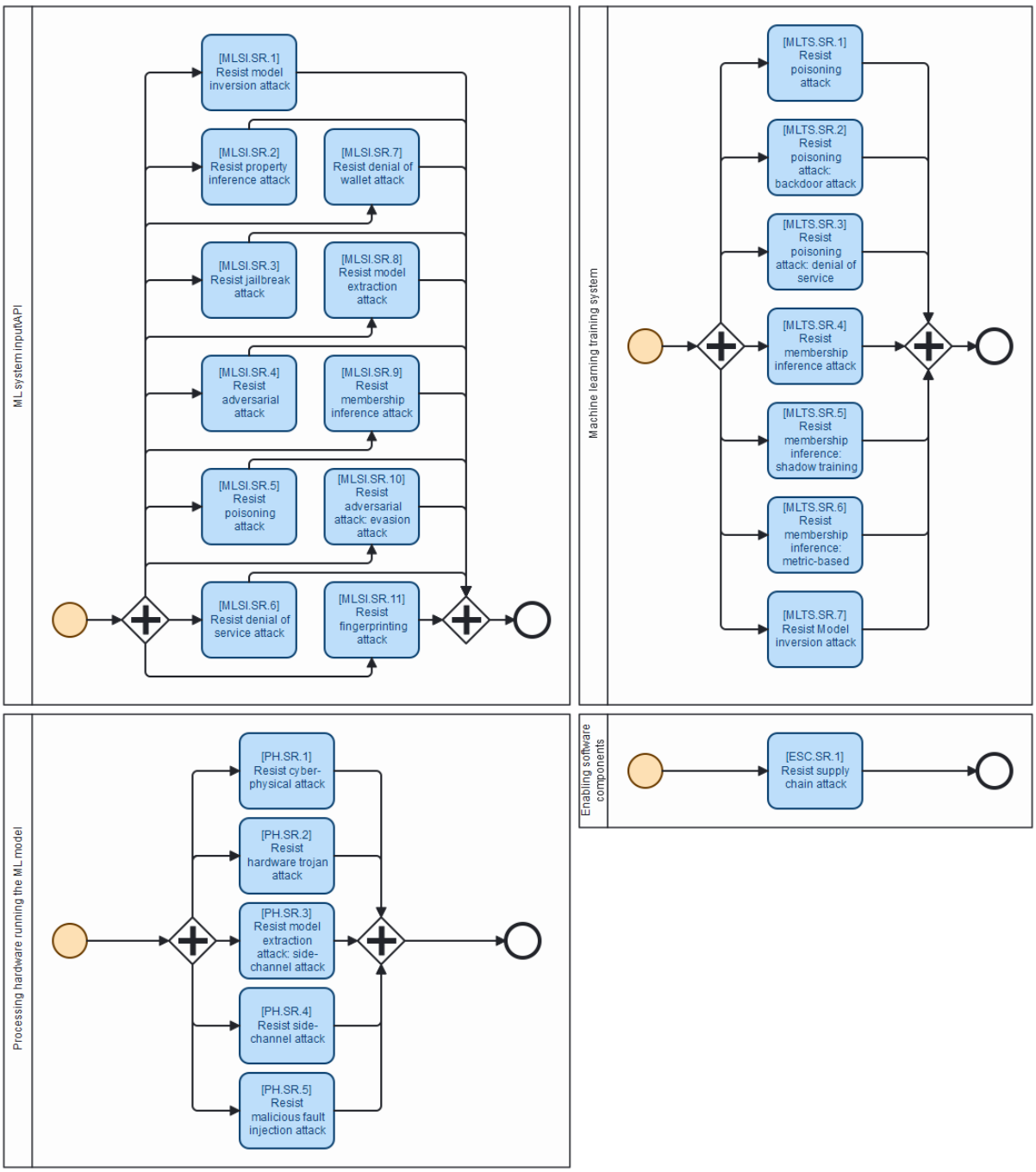

Figure 17. Business process modeling of security requirements for system assets.

4.6 Interactive Web Page

This chapter presents the interactive web page, which was made to allow the audience to interact with the diagrams depicted in the preceding chapters. The web pages provide an overview of all the elicited and synthesized data within this thesis, related to the threat model. The web page integrates UML class and diagrams, BPMN process flows, and textual explanations of the diagram elements to offer a comprehensive, user-friendly interface. In addition, the main page presents guidelines for interacting with the pages to get access to the relevant information, which organizations could apply to their machine learning and large language model systems. The goal of the web page is to assist machine learning engineers,

cybersecurity specialists, and other stakeholders in understanding, analyzing, and mitigating threats targeting machine learning and large language model systems.

The motivation behind the development of the interactive web page is to address the complexity and breadth of the threat-related information presented within the preceding chapters and thesis' appendices. The aim is to break down the information into more comprehensible chunks. The objectives are:

1. **Clarity in asset relationship and the structure of the machine learning system** through the presentation of UML class diagrams. The class diagrams illustrate the asset relationship and the connection between the assets and the threats.
2. **Holistic process understanding** by presentation of the BPMN process diagrams for training, inference, and fine-tuning processes, depicting the flow between the system assets.
3. **Threat mapping** – allows the user to explore which threats exist and which assets are targeted by such threats.
4. **Existing security controls** – linking of security controls and requirements to the affected assets to help organizations derive or refine their defensive strategies.

The following guidelines are provided to the audience to traverse the web pages:

1. Explore the UML class diagrams: hover over separate blocks/classes and click on the element that you are interested in. For example, "Input data" will lead you to the page with information about the business asset and threats that target the business asset.
2. Review the business asset definitions: By selecting a system asset like "ML system input/API", you will get to the page with the definition of the system asset and the involved business assets.
3. Examine threat–asset relationships: By selecting a business asset, a diagram and a list of threats will be presented. By selecting a threat on the diagram, you will get to the page with the threat definitions.
4. Check the BPMN process flows: There are 3 additional BPMN process diagrams, which can be selected from the top navigation bar, depicting separate processes related to the model.
5. Match threats to security requirements: The pages for threats provide the related security requirement and security controls, which can help you mitigate the threat.
6. Combine with your organizational context: Apply the information from the asset and threat pages to your own ML/LLM system's context. Identify which assets exist in your environment, note the relevant threats, and consult the associated security requirements and controls to plan the development or improvement of the present defenses.

The interactive web page is structured into two main components:

1. **UML and BPMN diagrams**
    a. **Asset UML class diagram –** the main diagram displaying the relationship between the system and business assets. A user can click on a class to view additional diagrams, if applicable, and textual information.
    b. **Threat-asset UML class diagrams** – depiction of the threats targeting a particular business asset and system assets. Connection with system assets shows the pathways, which can be exploited by a threat agent.
    c. **BPMN process diagrams** – depiction of 3 machine learning model processes. Each swim lane represents a system asset, the related tasks, and the transfer of business assets between the lanes.
2. **Textual explanations**

a. **Asset definitions** – explains the meaning of the elicited assets, their methods, and interaction with other assets, if applicable.
b. **Threat descriptions** – definition of targeted assets, affected security criteria, required threat agent capabilities, exploitable vulnerabilities, attack methods, and security countermeasures.
c. **Security control information** – enumeration of elicited controls, which can help mitigate the related threat and achieve the related security requirement.

It is expected that the interactive web page will provide value to the following stakeholders:

1. **ML engineers** should be able to rapidly identify applicable threats and security countermeasures and refine workflows with the acquired threat information.
2. **Security analysts** should gain a map of the existing threats and applicable defenses, which should facilitate risk assessment, countermeasures, and compliance implementations.
3. **Researchers and students** benefit from a holistic overview of the applicable threats and countermeasures to the machine learning and large language model system, which could support further research within this field.

A link to the repository with the web pages' codebase can be found in Appendix IV. In addition, the solution is temporarily hosted on the author's personal page (hosted by the university) - https://enos.itcollege.ee/~mikara/.

Below are provided 2 screenshots of the web pages – Figure 18 and Figure 19. The screenshots depict parts of the main and threat information pages.

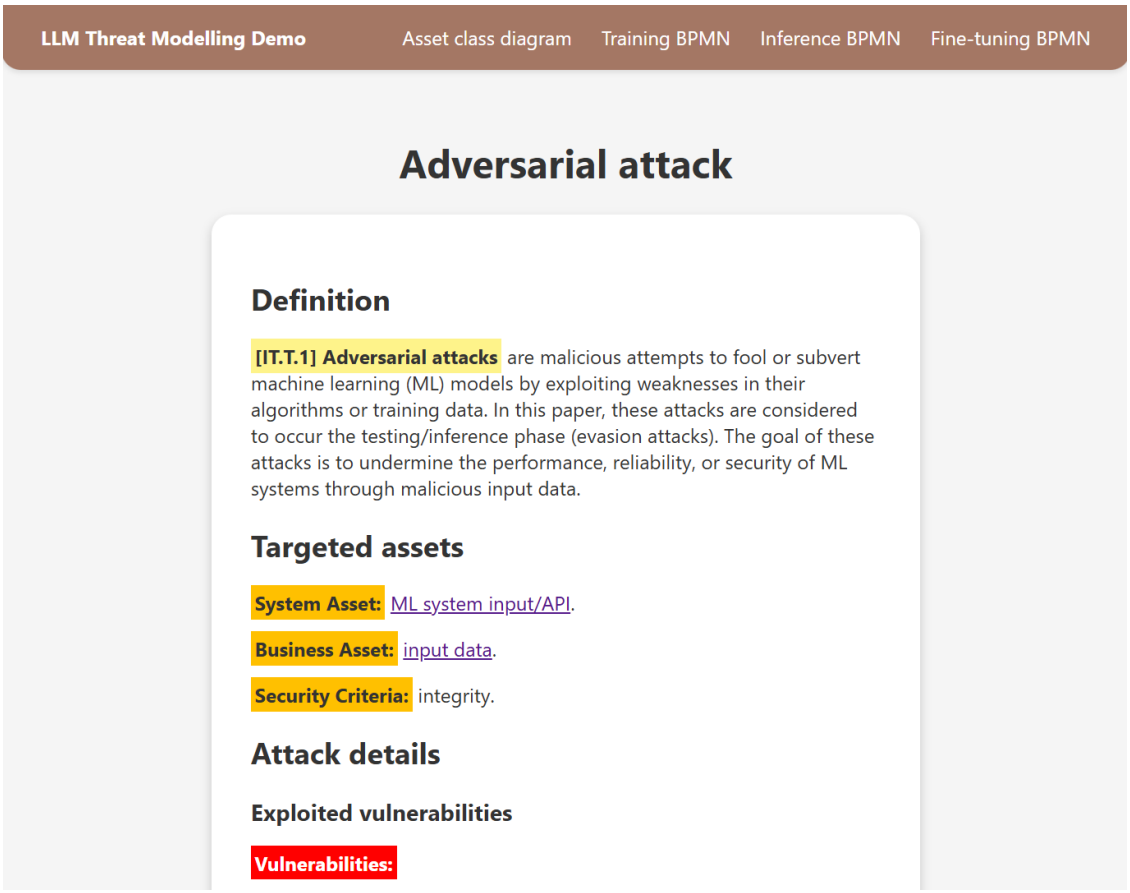

Figure 18. A screenshot depicting a web page with threat information.

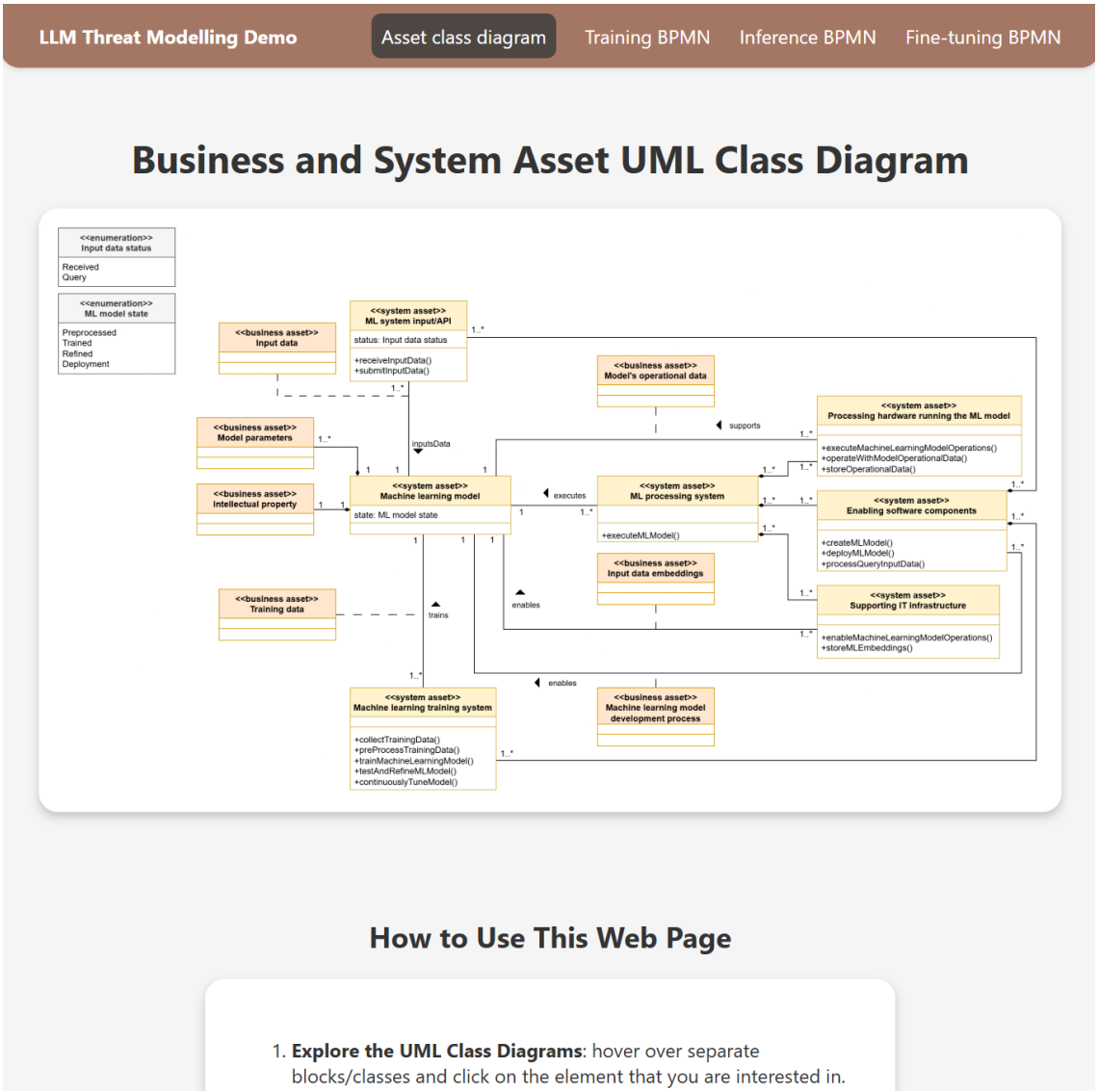

Figure 19. A screenshot of a part of the main web page.

4.7 Summary

Chapter 4 instantiates the gathered evidence from the conducted SLR by designing an asset-centric threat model for large language model-based systems. Firstly, a business-system asset relationship model was established, depicting an LLM-based system's components and interdependencies, thus setting a baseline for the following models. The chapter introduced a detailed definition and meaning behind the elicited assets. In addition, security criteria for business assets were provided. Followed by dedicated models and lookup tables (Tables 3-9), which map threats, the targeted system assets, affected business asset's security criteria, and a security requirement. To provide an operational context, the chapter models three process diagrams, depicting key processes of the LLM model lifecycle. The models are followed by process diagrams linking security requirements to each of the system assets involved in the depicted processes. Finally, an interactive web page is set up, which links the diagrams with the threat and security control definitions, enabling practitioners to navigate from a high-level overview to countermeasures. Based on the ISSRM domain model, these artifacts collectively form the threat model artifact, mapping the elicited threat taxonomy into a framework for LLM risk analysis. The produced artifact fulfills the DSR's stage – artifact.

5. Validation of Threat Model

In the current chapter, two processes of validation are described. First, a feasibility analysis is conducted of the produced threat model against an existing LLM model. Afterwards, an empirical analysis for one threat is conducted to determine the threat's applicability. For both validations, the outcomes and limitations are discussed based on the conducted feasibility analysis and empirical analysis. This chapter fulfills the validation stage of the DSR paradigm.

5.1 Feasibility Analysis of Threat Model

A feasibility analysis, a qualitative, documentation coverage-based validation method, is conducted to test the produced threat model against an existing LLM system. The **goal** of the feasibility analysis is to determine the **completeness** of the produced threat model. If the reference threat model allows to determine system assets within the analyzed system, which can be affected by the elicited risks.

This chapter aims to answer the following **validation question**: How well does the proposed threat model for LLMs (designed in the 4th chapter) represent the composition of an LLM system and allow for eliciting applicable threats? To answer the set question, a high-level **feasibility analysis** is conducted against a selected LLM model. The analysis is conducted in the following way:

1. The data and the evidence are collected from the primary sources – documentation, codebase, and reports of the target model.
2. A qualitative analysis is conducted over the target sources to map the target system to the produced threat model.
3. The applicability of threats from the threat model is determined.
4. The documentation is reviewed for possible official countermeasures.

Analysis of the target model's documentation and codebase allows for mapping the components of the target system to the produced threat model. The completeness of the mapping determines the completeness of the evaluated threat model.

The following **metric** is defined for evaluating the completeness of the threat model based on the use case mapping:

1. A mapping of a target system component to the system asset in the threat model is represented with 3 distinct values:
    a. 1 – The component of the system under analysis fully maps to a system asset in the threat model.
    b. 0.5 – The alignment between the threat model's system asset and the analyzed model's component is partial. This means that the model's component may possess a subset of the properties defined for the mapped system asset, or if the component combines multiple system assets in itself.
    c. 0 – There is no direct mapping between the defined system asset and a component of the target model.
2. The sum of the mapping values is taken.
3. The sum is divided by the count of the system assets, producing the final completeness score.

In formulaic representation, completeness, used for evaluation of the threat model's completeness, is calculated as follows:

$$Completenss = \frac{\sum_{i=1}^{n}(Asset\ specific\ mapping\ score)_i}{\sum_{i=1}^{n}(System\ Asset)_i}$$

This completeness analysis is conducted to uncover gaps in the produced threat model's coverage of LLM system components. An LLM model is selected, and its documentation and codebase are analyzed, mapping the present components to the system assets within the developed threat model. If the mapping is present, it is assumed that the elicited threats may apply to the target system, unless a clear indication of countermeasures is present.

A Mistral Small, the latest (as of April 2025) open-weight model, released by Mistral AI, is chosen for the feasibility analysis due to the following properties:

1. The model is provided under the Apache 2.0 license, thus making the model's codebase open for analysis [47].
2. Mistal AI provides an official inference library, which can be analyzed [48].
3. Mistral AI offers documentation on the usage of the model and presents safety features [49].

Mapping of the following system assets is conducted (defined in Chapter 4.2): "ML system input/API", "Machine learning model", "ML processing system", "Processing hardware running the ML model", "Enabling software components", "Supporting IT infrastructure", "Machine learning training system".

ML system input/API:

The input interface is provided in the example of the chatting process with the model in the "mistral-inference" codebase within the "getting_started.ipynb" notebook [50], an excerpt is presented in Figure 20. The produced class object contains the preprocessed input data, utilized by the inference "generate.py" [51] script.

```
# chat completion request
completion_request =
ChatCompletionRequest(messages=[UserMessage(content="Explain Machine
Learning to me in a nutshell.")])
```

Figure 20. Definition of the input function usage.

The definition of the class "ChatCompletionRequest", processing the input text string, can be found in [52]; the excerpt of the function's contract is depicted in Figure 21.

```
class ChatCompletionRequest(BaseCompletionRequest,
Generic[ChatMessageType]):
    model: Optional[str] = None
    messages: List[ChatMessageType]
    response_format: ResponseFormat =
Field(default_factory=ResponseFormat)
    tools: Optional[List[Tool]] = None
    tool_choice: ToolChoice = ToolChoice.auto
    truncate_for_context_length: bool = False
```

Figure 21. Definition of the class that takes the input.

In addition, open codebases are present, defining the API for chat completion in Python [53] and TypeScript [54] languages. With the definition of the asset from Chapter 4.2, the found components do map to the system asset. Thus, the asset-specific score for the "ML system input/API" asset is determined to be 1.

Machine learning model:

The target model can be acquired in a format of "safetensors" at the Hugging Face model repository [55]. With the definition of the system asset from Chapter 4.2, the found component does match the system asset. The asset-specific score for the "Machine learning model" asset is 1.

Processing hardware running the ML model:

From the official blog post about publication of the model [56], it is indicated that the model can be run on a personal desktop computer. The quote indicating such a capability: "Lightweight: Mistral Small 3.1 can run on a single RTX 4090 or a Mac with 32GB RAM. This makes it a great fit for on-device use cases." [56].

Alternatively, the model can be hosted and executed on a cloud platform of the user's choice [57], such as Azure AI, AWS Bedrock, Google Cloud Vertex AI Model Garden, Snowflake Cortex, IBM watsonx, and Outscale. In addition, Mistral AI offers the ability to run the model through their official deployment platform - La Plateforme [58]. According to the system asset's definition from Chapter 4.2, the asset-specific score for the "Processing hardware running the ML model" asset is 1.

Enabling software components:

Mistral has a set of tools in the "mistral-common" repository [59], which is utilized by the official inference library [51]. The inference library [51] provides the capability of running the model. Alternatively, vLLM, an inference and serving library [60], can be utilized for the execution and handling of the model, as shown in the examples at [55]. With the definition of the system asset from Chapter 4.2, the found component does match the system asset; the score for the asset mapping is 1.

Supporting IT infrastructure:

The computational environment varies in the form of model deployment. From the capability of running the model locally [56], on a personal computer, it can be inferred that the personal computer with the necessary software components plays the role of the defined asset.

If the model is hosted and run on a cloud platform, then the platform of choice is utilized for the provisioning of auxiliary functionality. In the case of the official La Plateforme [58], it is capable of provisioning inference containers, routing, caching, load balancing, monitoring, etc. The score of the mapping of the system asset is 1, considering the asset's definition from Chapter 4.2.

ML processing system:

The system contains the "processing hardware running the ML model", "enabling software components", "supporting IT infrastructure", and system assets as its components, according to the definitions from the 4th chapter.

As the ML processing system, a local setup of necessary libraries and personal computer hardware can be considered, as the model can be run with consumer-grade hardware [56]. Alternatively, the model can be run with the official deployment platform La Plateforme [58],

capable of providing the capabilities of defined system sub-assets, or another cloud provider of choice [57].

The score for the mapping of the asset is determined to be 1, as:

1. Other sub-system assets received a mapping score of 1;
2. The mapping is established with the existing platform, encapsulating the capabilities of the system asset.

Machine learning training system:

No source was found providing coverage of the training system for the target model. Although the official documentation describes the process of fine-tuning – "Fine-tuning is a powerful technique for customizing and optimizing the performance of large language models (LLMs) for specific use cases. [61]". Mistral provides a fine-tuning API with their deployment platform La Plateforme [58], [61]. In addition, an official codebase for local fine-tuning has been published – "mistral-finetune" [62].

According to the designed asset model - Figure 4, the "Machine learning training system" has a method responsible for fine-tuning "continuouslyTuneModel()". As the direct description and documentation of the training process was not found, but a sub-process of fine-tuning was found to be defined, the mapping is considered to be partial. The asset-specific score for the "Machine learning training system" asset is 0,5. The score for the mapping of system assets to components of the target system is presented below:

Table 10. Scores for the mapping of threat model system assets to the model components.

System asset	Mapping score	Rationale
ML system input/API	1	Identified in "mistral-common" codebase (ChatCompletionRequest) [52] and API codebases [53], [54]. Aligns with the concept of an external interface.
Machine learning model	1	The model weights are defined and available in the format of "safetensors" in a Hugging Face repository [55].
Processing hardware running the ML model	1	Can be processed on local desktop hardware [56] or on a cloud platform [57].
Enabling software components	1	The execution of the model requires multiple official libraries (mistral-inference [51], mistral-common [59]) or a 3rd party library – vLLM [60].
Supporting IT infrastructure	1	Local [56] desktop setup provides needed capabilities with the local environment, updated with the necessary software components. In addition, platforms, like Le Plateforme [58], provide the support capabilities for model execution.
ML processing system	1	A local collection of hardware and software libraries maps to the subsystem

		assets. Alternatively, the capabilities of a processing system can be provided by a supported cloud platform [57].
Machine learning training system	0,5	Partial mapping: There is no documentation of the training pipeline. But there are fine-tuning scripts [62], fulfilling one of the functionalities of the system asset.

The total amount of the system assets – 7; the sum of the mapping scores – 6,5. Considering the defined completeness evaluation formula, the **completeness score** is $\frac{6.5}{7} \approx 0{,}93$. Based on the conducted analysis. The mapping of the system assets is presented in the form of the following system asset diagram Figure 22, adapted from Figure 4. Business assets, attributes, and methods were omitted for brevity, as they were not analyzed for the mapping to the target model. The diagram depicts the relationship between the mapped system assets of a locally run Mistral Small model.

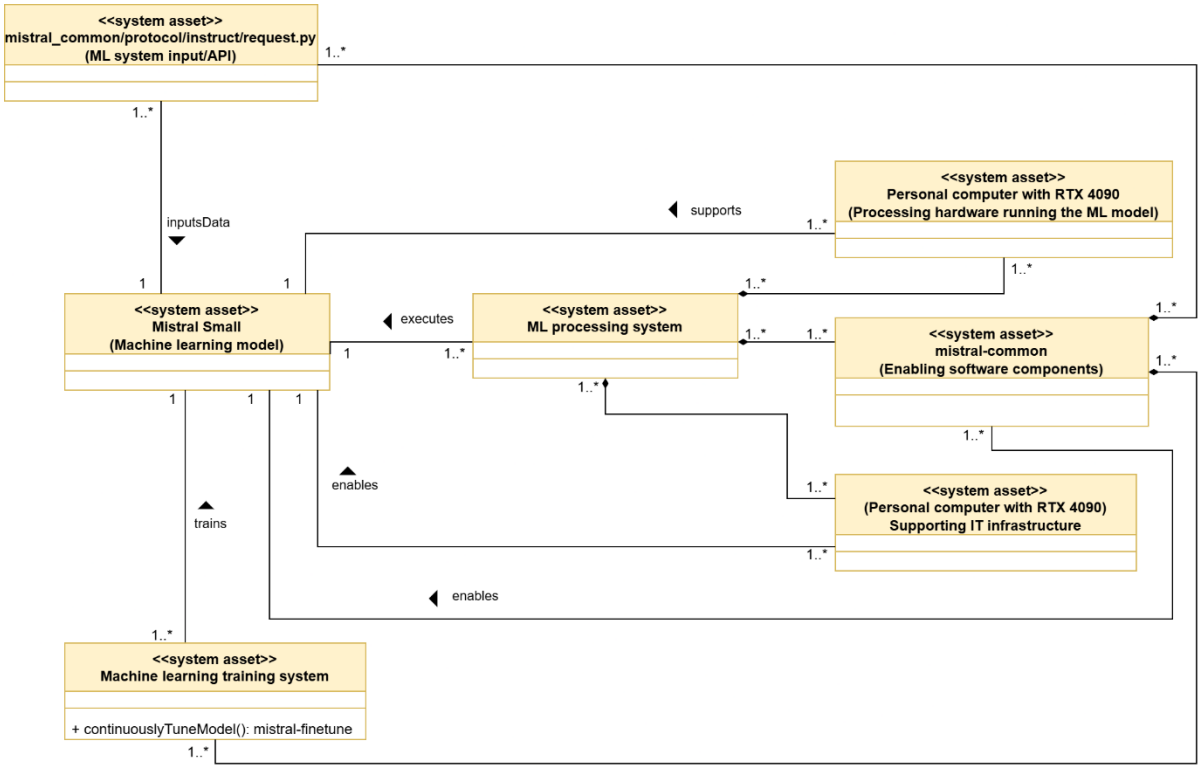

Figure 22. Adapted system asset model to the local Mistral model.

Considering the ability to deploy on a cloud platform. The diagram could be further abstracted, combining the processing "hardware running the ML model", "enabling software components", and "supporting IT infrastructure" system assets under the "ML processing system" system asset. This can be done, due to the cloud platform incorporating the capabilities of the combined system assets, based on the conducted analysis. Taking the official deployment platform "Le Platforme", the initial asset relationship diagram is modified to Figure 23.

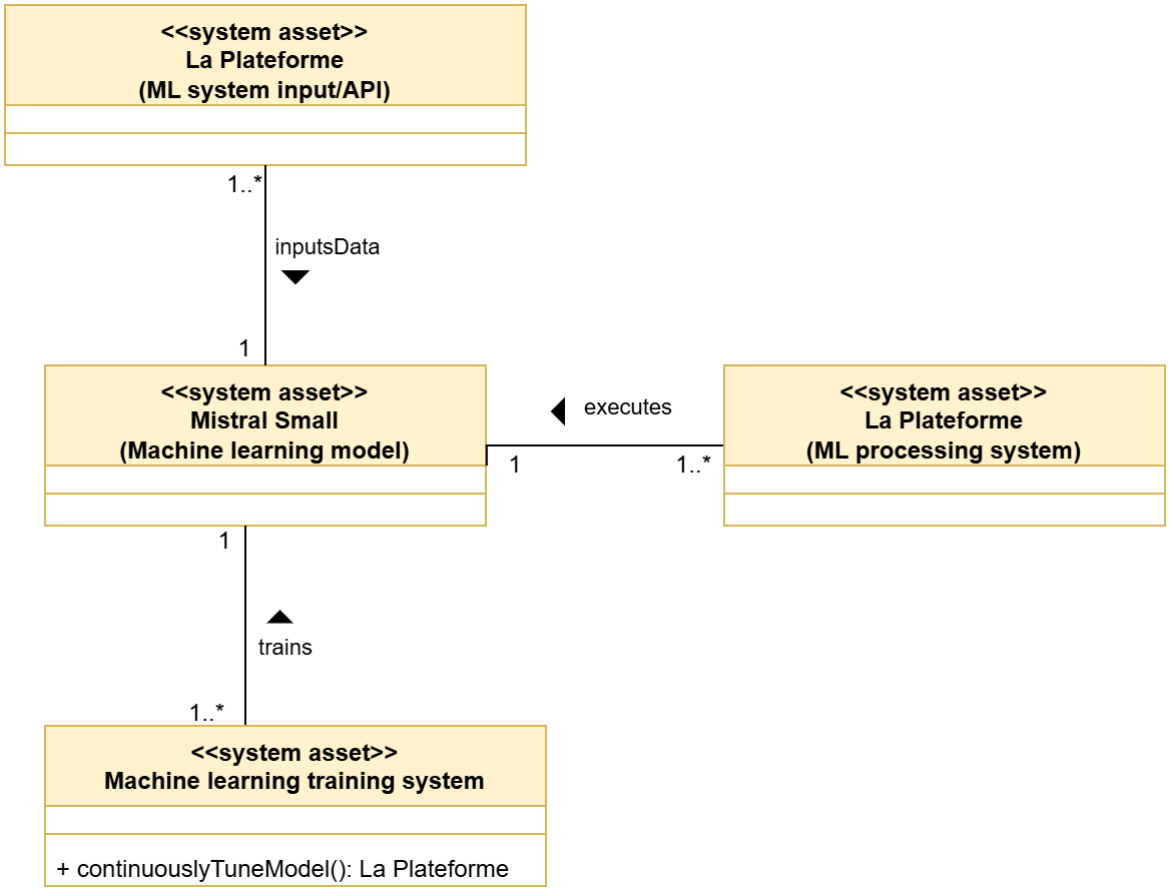

Figure 23. Adapted asset model to the Mistral model deployed with the La Plateforme.

From the conducted mapping, the following system assets have been mapped to a direct component: “ML system input/API”, “Machine learning model”, “Processing hardware running the ML model”, “Enabling software components”, “Supporting IT infrastructure”, “ML processing system”. Thus, the threats targeting the “Machine learning training system” are considered to be inapplicable due to the partial mapping of the system asset to a component of the analyzed model system.

5.2 Consideration of Countermeasures and Applicability of Threats

This sub-chapter considers the countermeasures that were documented for the studied model. Based on the documentation, threats are determined, the risk of which is lowered by documented countermeasures. Based on this review, a subset of the threats is considered to be less applicable to the target LLM model.

Out of all 24 elicited threats, one threat - [IT.T.3] Jailbreak attack, targeting the ML system input/API asset, is less applicable. Its applicability can be hampered by the official implementation of a moderation service, which, if used, may detect the harmful context [63]. This functionality could be used by middleware, parsing the inputs to filter them out before submitting them for processing to the model. In addition, there is an official option within the inference API to integrate a guardrail on submitted prompts. With the optional guardrail, the messages to the model through the API are prepended with a system prompt, suggesting safer behavior for the system.

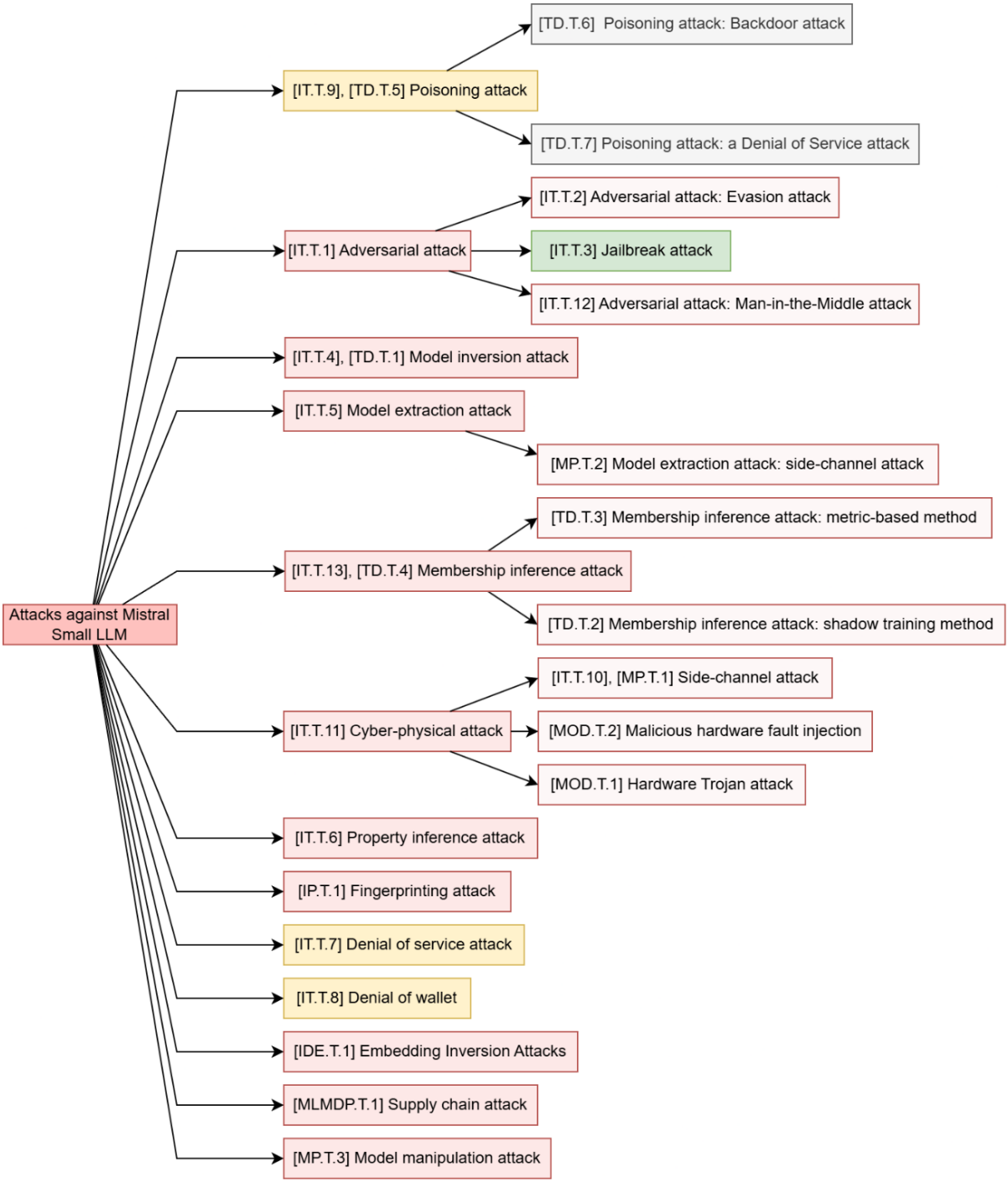

Figure 24. A tree diagram of attack types potentially applicable to Mistral Small.

Moreover, the official platform for running the models from Mistral [63] and support for 3rd party integrations, provides access to observability tools [64]. The observability tools can monitor the following elements: individual LLM calls (input prompt, model, output), application (execution flow), input prompt (prompt template, examples, retrieve context, memory, tools), model (model version, configuration settings, hyperparameters), processed tokens and cost [64]. 3rd party integration with "Weights & Biases" additional tracking of fine-tuning process' metrics [64]. These observability tools may allow the LLM system operator to detect malicious adversarial input and undesirable changes in fine-tuning metrics. From the

elicited security controls, the risk from the following threats may be reduced with the observability tools: [IT.T.9], [TD.T.5] Poisoning Attack, [IT.T.7] Denial of service attack, [IT.T.8] Denial of wallet attack.

The attack tree of threats, elicited from the systematic literature review, "Figure 3," is adapted for the depiction of threats, which may apply to the Mistral Small in Figure 24. The attacks marked in red and lighter red may be applicable, as the targeted asset by the threat was mapped to a component in the analyzed system. The threats in yellow (e.g., [IT.T.9], [TD.T.5] Poisoning Attack) may be less applicable if auxiliary observability systems are implemented and used by the user of the LLM system. The threat in green ([IT.T.3] Jailbreak attack) is the threat against which there are present official countermeasures, thus reducing the likelihood of its realization. Threats marked in grey (e.g., [TD.T.7] Poisoning attack: a Denial-of-Service attack) are considered to be inapplicable, as the complete mapping between the targeted system asset and a component of the analyzed system was not established.

5.3 Test of a Jailbreak Attack Threat

5.3.1 Objective

The purpose of the current threat study is to empirically examine whether the Jailbreak threat (IT.T.3) is exploitable against the target Mistral Small model under the feasibility analysis. In addition, the efficacy of the official guard-railing solution is analyzed to determine to what extent the threat is mitigated.

Empirical exercise provides evidence that a threat from the threat model manifests in a production system. Demonstration of a compromise and quantification of the threat mitigation corroborate the practical feasibility of the attack and proposed mitigation, which can inform the threat model's user for control selection.

The investigation concentrates on one threat due to the limited afforded timetable for hands-on validation. The jailbreak attack is prioritized due to its ease of realization and verification in a black-box setting. As analysis of a pair of an input prompt and the model's output is sufficient to determine if the target model has been compromised. Code, parent dataset, and sets of prompt-output pairs can be found in the repository referred to in Appendix IV.

5.3.2 Experimental Setup

The experiment was conducted against an official latest production instance of Mistral Small as of April 2025 on the official La Plateforme [58] solution. The access to the model is exposed via an inference Mistral AI API [65].

Configuration: The model was queried two separate times with a set of prompts, once without any guardrails, and the second time with the proposed guardrail solution enabled. Other API behavior settings were left untouched.

Adversarial prompt corpus: A corpus of 100 selected known jailbreak prompts was used. The prompts were extracted from the public JailBreakV-28K benchmark dataset [66]. The parent dataset contains malicious prompts of 5 jailbreak methods, covering 16 safety policies.

Metrics: A proportion in percentages is calculated of prompts leading to the production of harmful output against the total applicable prompt count. The proportion of resisted prompts is calculated as well. Formula (an example for prompts leading to harmful output): $\frac{number\ of\ prompts\ leading\ to\ harmful\ output}{total\ number\ of\ applicalbe\ prompts} \times 100\%$.

5.3.3 Approach

The test of the threat followed the following steps:

1. 100 prompts are randomly selected from the parent dataset.
2. 100 prompts are issued to the target model over an API without any safety measures enabled. The outputs from the model are logged in pairs with the corresponding prompt.
3. 100 prompts are issued to the target model over an API with the official safety measure enabled. The outputs from the model are logged in pairs with the corresponding prompt.
4. The pairs of prompts and outputs are manually and qualitatively analyzed.
    a. Pairs of prompts and outputs are filtered out if the prompt employs a multi-modal jailbreak method.
    b. Pairs of prompts and outputs are counted, which have led to the production of harmful output, and which have not. ASR is calculated.

5.3.4 Results

Out of the 100 randomly selected pairs, 27 were filtered out due to they relied on the accompanying image input to the model. The experiment focused on textual prompts. Thus, the total number of applicable prompts – 73. In Table 11 and Table 12, metrics are provided for the number of prompts that lead to the production of harmful output and which were resisted. In both setups, there was one unique prompt that led the model to role-play according to the prompt's directions, but it failed to respond to the main harmful question. Although the output still contained harmful speech.

Table 11. Metrics for prompts queried against a model without guardrails.

Condition	Count	Proportion (%)
Prompts leading to harmful output	57	78%
Prompts, which were resisted by the model	15	21%
Prompt leading to partially harmful output	1	1%

Table 12. Metrics for prompts queried against a model with guardrails.

Condition	Count	Proportion (%)
Prompts leading to harmful output	51	70%
Prompts, which were resisted by the model	21	29%
Prompt leading to partially harmful output	1	1%

5.3.5 Discussion

The experiment confirms that the jailbreak threat applies to the target model, with 78% of applicable prompts in a setup without guardrails. This number was reduced to 70% in a setup with official safety measures. The tested countermeasure produces a measurable (8%) reduction in successful jailbreak prompts. As most of the prompts continued to successfully

compromise the target model, the use of the model should consider additional countermeasures. For example, adversarial training could be done with the official fine-tuning [61] service.

5.4 Threats to Validity

The validity of the results produced by the conducted feasibility analysis and empirical test can be threatened as follows:

1. **Internal validity:**
    a. **History.** The studied model and official countermeasures may be updated after the feasibility study is conducted, potentially leading to different outcomes in repeated studies. In addition, the field of large language model systems continues to evolve. The analysis of the model and the threat model itself may not account for new model system architecture and threats.
    b. **Testing**. In the scope of the empirical test, the model may be trained on the submitted prompts, potentially leading to an updated behavior in the future.
    c. **Selection.** Only one model was analyzed in combination with official software components for model execution. The findings may not be fully generalized to other LLM-based systems, which diverge in their architecture from the covered simple setup.
2. **Construct validity:**
    a. **Mono-method bias.** Only one model was analyzed in combination with official software components for model execution. The findings may not be fully generalized to other LLM-based systems, which diverge in their architecture from the covered simple setup.
    b. **Experimenter expectancies.** The author of the threat model has conducted the feasibility analysis, which could have introduced bias into the judgment of the threat model's alignment with the target system, and thus, its applicability. The documentation did not present the composition of the system, the presence of countermeasures, or the depiction of the training pipeline. Thus, the mapping of system assets may have involved the author's bias.
3. **Conclusion validity:**
    a. **Ambiguity about the direction of causal influence.** Determination of 0,5 and 1 mapping involves the author's judgment, which may involve bias, and thus, present bias may lead to incorrect mapping and conclusions.

5.5 Conclusions

The conducted analysis allows us to answer the posed question: "How well does the proposed threat model for LLMs (designed in Chapter 4) represent the composition of an LLM system and allow for eliciting applicable threats?". In addition, the chapter's outcomes fulfill the DSR's validation stage of the produced artifact.

The analysis has shown that the produced threat model partially aligns with the target model (Mistral Small) and official surrounding components, with a **completeness score of 0,93 out of 1**. Most of the threat model system assets are mapped to the target system's components. The partial mapping of one system asset, representing a training system, stems from the limited coverage of Mistral's training pipeline.

An empirical test was conducted to determine the applicability of the Jailbreak attack (IT.T.3) and the official safety countermeasure. The experiment confirmed the applicability of the jailbreak threat in practice. 78% of the test prompts lead to harmful output without guardrails, and 70% lead to a similar outcome with official safety guardrails. While safety improvement

is measurable, the threat's continued applicability underscores the need for additional countermeasures within the specific tested setup. This empirical test confirms the applicability of the jailbreak threat, defined within the threat model, thus further supporting the applicability of the designed threat model.

Overall, this could mean that the produced threat model depicts real LLM systems at a high level. Thus, the model could be utilized to determine a set of applicable threats, which could be further narrowed down based on the target system's properties and the business' needs. Nevertheless, the feasibility analysis was conducted from the author's perspective, which could have introduced bias into the judgment of the threat model's alignment with the target system. Future studies or supplementary documentation may further refine the threat model's completeness in the coverage of assets and threats.

6. Conclusion

The work set out to develop the first threat model, which is tailored to the large language model systems. The study combined and executed upon the parts of a design-science research paradigm:

1. theory – conducted a systematic literature review of 45 peer-reviewed and grey sources;
2. artifact – design an asset-oriented threat model for LLMs following the ISSRM domain model;
3. validation – conducted a feasibility study of the produced threat model and an empirical test of a jailbreak threat.

A taxonomy of 13 parent attack classes (a total of 24 variants) was established (Chapter 3), which may affect LLM system components, from data poisoning and jailbreak attacks to embedding inversion and supply-chain compromise. Chapter 4 converted the threat insights into a high-level threat model by modeling the business and system assets of an LLM system. The assets' security criteria and impact were determined in case of security criteria compromise. Elicited threats and security requirements were mapped to LLM system assets following the *Information Systems Security Risk-Management Domain Model* (ISSRM). Finally, the threat model's applicability was tested (Chapter 5), and the threat model achieved a 0.93 completeness score out of 1 for mapping an open-source Mistral Small model. In addition, an experiment was conducted to determine the applicability of jailbreak attacks, showing 78% jailbreak prompting success without guardrails and 70% with the official safety measure.

The following additional key findings can be highlighted:

1. **The breadth of the threat surface.** The LLM ecosystem inherits a wide spectrum of ML attacks (22 attack types) while introducing 2 unique attack types for this type of model.
2. **The gap in holistic frameworks.** No prior research was found cataloging applicable attacks against LLM systems.
3. **Jailbreak attack is applicable despite the provided official safety measures.** Experimental evidence showed that the official safety layer for the Mistral Small model on the "La Plateforme" platform only partially mitigated the jailbreak attack, underlining the need for defense-in-depth strategies of defense.

6.1 Limitations

The threats to the thesis' validity can be defined with the following types:

1. **Internal validity:**
    a. **History.** Large language models are continuously studied and modified. The designed threat model may not fit new model system architectures and threats. Newer LLM designs could shift the completeness score of the produced threat model.
    b. **Testing**. In the empirical test, the tested model may be trained on the submitted prompts, leading to an updated behavior in the future and a mismatch with the thesis's empirical results.
    c. **Selection.** Only one open-weight model was studied. This limits the comparative baseline of the thesis's results. In addition, manual literature selection, extraction of information about threats and assets in SRL, and

mapping of threats to assets rely on the author's judgment. Thus, the sole author's judgment may have introduced bias in the design of the threat model.

2. **Construct validity:**
    a. **Mono-method bias.** Feasibility analysis of the design threat model relied on source code and documentation analysis. While an empirical test was made with one threat, no interviews were undertaken to involve an experienced 3rd party.
    b. **Experimenter expectancies.** The author of the thesis has conducted the SLR, threat model design, and feasibility analysis. The author's bias could have been introduced in the extraction of threat information, threat model design, and judgment of the threat model's alignment with the analyzed system.
    c. **Completeness metric.** The sole 0 to 1 scoring may oversimplify mapping analysis.

6.2 Answer to Research Question

Returning to the **main research question** of the thesis: "How to model threats against large language model systems to identify and categorize security threats, thereby enabling structured risk analysis and selection of countermeasures?". **Answer:** The threats against LLM systems can and were modeled by enumerating known risks, LLM system assets, and controls through a rigorous literature review following ISSRM domain model. Each elicited threat was mapped to the targeted asset. Cybersecurity and AI practitioners receive an actionable, asset-centric threat model that supports qualitative risk assessment. The feasibility and empirical validations confirm that the resulting threat model captures real-world LLM architectures on a high level.

6.3 Novelty

The conducted systematic literature review (Chapter 3.6) determined the absence of security-oriented frameworks and threat modeling for LLM-based systems. This thesis has addressed this identified gap by adapting the ISSRM (Chapter 2.2.2) for bespoke threat modeling of identified threats against an LLM system, tuned to the system's asset composition. Security analysts and ML engineers can now use the produced threat model to enumerate potentially applicable threats in the system design phase and trace each of the risks to proposed countermeasures.

6.4 Future Work

The study could be continued with the following goals. The presented goals could further innovate on the outcomes produced in this thesis.

1. **Quantitative risk scoring.** The produced qualitative mappings could be augmented with probabilistic likelihood and impact estimators to further support cost-benefit risk mitigation analysis.
2. **Multi-model generalization.** The feasibility study could be conducted across a wider set of open and closed models. In addition, multi-agent systems could be analyzed as well. Such studies would further determine the validity of the model and lead to its updates and alignment with new developments in the industry.
3. **Empirical verification** of the applicability of remaining untested threats.
4. **Empirical verification** of the proposed countermeasures against the elicited threats.
5. **Defense automation tooling.** The threat model could be used for the development of a DevSecOps plugin, which analyzes the LLM systems's lifecycle in the pipeline, notifying of potential threats and proposing possible countermeasures. This solution could be linked to real-time threat intelligence, thus informing users of emergent attack vectors.

References

[1] K. Huang, Y. Wang, B. Goertzel, Y. Li, S. Wright, and J. Ponnapalli, Eds., *Generative AI Security: Theories and Practices*. in Future of Business and Finance. Cham: Springer Nature Switzerland, 2024. doi: 10.1007/978-3-031-54252-7.

[2] X. Qi, K. Huang, A. Panda, P. Henderson, M. Wang, and P. Mittal, "Visual Adversarial Examples Jailbreak Aligned Large Language Models," presented at the Proceedings of the AAAI Conference on Artificial Intelligence, 2024, pp. 21527–21536. doi: 10.1609/aaai.v38i19.30150.

[3] E. Quiring, D. Arp, and K. Rieck, "Forgotten Siblings: Unifying Attacks on Machine Learning and Digital Watermarking," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, Apr. 2018, pp. 488–502. doi: 10.1109/EuroSP.2018.00041.

[4] L. Muñoz-González and E. C. Lupu, "The security of machine learning systems," *Intelligent Systems Reference Library*, vol. 151, pp. 47–79, 2019, doi: 10.1007/978-3-319-98842-9_3.

[5] Ö. F. Tuna and F. E. Kadan, "Security of AI-Driven Beam Selection for Distributed MIMO in an Adversarial Setting," *IEEE Access*, vol. 12, pp. 42028–42041, 2024, doi: 10.1109/ACCESS.2024.3378263.

[6] "Machine Learning for Security and the Internet of Things: The Good, the Bad, and the Ugly | IEEE Journals & Magazine | IEEE Xplore." Accessed: Feb. 07, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/8879591

[7] A. Raja, L. Njilla, and J. Yuan, "Adversarial Attacks and Defenses Toward AI-Assisted UAV Infrastructure Inspection," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 23379–23389, Dec. 2022, doi: 10.1109/JIOT.2022.3206276.

[8] N. Pitropakis, E. Panaousis, T. Giannetsos, E. Anastasiadis, and G. Loukas, "A taxonomy and survey of attacks against machine learning," *Computer Science Review*, vol. 34, p. 100199, Nov. 2019, doi: 10.1016/j.cosrev.2019.100199.

[9] Y. Liu *et al.*, "ML-DOCTOR: Holistic Risk Assessment of Inference Attacks Against Machine Learning Models," presented at the Proceedings of the 31st USENIX Security Symposium, Security 2022, 2022, pp. 4525–4542.

[10] Owaspllmp. Admin, "OWASP Top 10 for LLM Applications 2025," OWASP Top 10 for LLM & Generative AI Security. Accessed: Feb. 11, 2025. [Online]. Available: https://genai.owasp.org/resource/owasp-top-10-for-llm-applications-2025/

[11] R. Matulevičius, *Fundamentals of Secure System Modelling*. Cham: Springer International Publishing, 2017. doi: 10.1007/978-3-319-61717-6.

[12] B. A. Kitchenham, "Procedures for Performing Systematic Reviews," 2004. [Online]. Available: https://api.semanticscholar.org/CorpusID:54019416

[13] A. Habbal, M. K. Ali, and M. A. Abuzaraida, "Artificial Intelligence Trust, Risk and Security Management (AI TRiSM): Frameworks, applications, challenges and future research directions," *Expert Systems with Applications*, vol. 240, p. 122442, Apr. 2024, doi: 10.1016/j.eswa.2023.122442.

[14] P. Breda, R. Markova, A. F. Abdin, N. P. Mantı, A. Carlo, and D. Jha, "An extended review on cyber vulnerabilities of AI technologies in space applications: Technological challenges and international governance of AI," *Journal of Space Safety Engineering*, vol. 10, no. 4, pp. 447–458, Dec. 2023, doi: 10.1016/j.jsse.2023.08.003.

[15] "Article 3: Definitions | EU Artificial Intelligence Act." Accessed: Apr. 27, 2025. [Online]. Available: https://artificialintelligenceact.eu/article/3/

[16] "Regulation - EU - 2024/1689 - EN - EUR-Lex." Accessed: Apr. 27, 2025. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng

[17] T. F. Blauth, O. J. Gstrein, and A. Zwitter, “Artificial Intelligence Crime: An Overview of Malicious Use and Abuse of AI,” *IEEE Access*, vol. 10, pp. 77110–77122, 2022, doi: 10.1109/ACCESS.2022.3191790.
[18] G. Apruzzese, H. S. Anderson, S. Dambra, D. Freeman, F. Pierazzi, and K. Roundy, “‘Real Attackers Don’t Compute Gradients’: Bridging the Gap Between Adversarial ML Research and Practice,” in *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, Feb. 2023, pp. 339–364. doi: 10.1109/SaTML54575.2023.00031.
[19] Andrew Glassner, “Deep Learning: A Visual Approach.” Accessed: Mar. 30, 2025. [Online]. Available: https://nostarch.com/deep-learning-visual-approach
[20] “Membership Inference Attacks Against Machine Learning Models | IEEE Conference Publication | IEEE Xplore.” Accessed: Feb. 07, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/7958568
[21] S. Zhou, C. Liu, D. Ye, T. Zhu, W. Zhou, and P. S. Yu, “Adversarial Attacks and Defenses in Deep Learning: From a Perspective of Cybersecurity,” *ACM Comput. Surv.*, vol. 55, no. 8, p. 163:1-163:39, Dec. 2022, doi: 10.1145/3547330.
[22] A. Kara, N. Koprucu, and M. E. Gursoy, “Beta Poisoning Attacks Against Machine Learning Models: Extensions, Limitations and Defenses,” in *2022 IEEE 4th International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA)*, Dec. 2022, pp. 178–187. doi: 10.1109/TPS-ISA56441.2022.00031.
[23] H. Dong, J. Dong, S. Wan, S. Yuan, and Z. Guan, “Transferable adversarial distribution learning: Query-efficient adversarial attack against large language models,” *Computers & Security*, vol. 135, p. 103482, Dec. 2023, doi: 10.1016/j.cose.2023.103482.
[24] “Threat Modeling: Designing for Security.” Accessed: Apr. 27, 2025. [Online]. Available: https://shostack.org/books/threat-modeling-book.html
[25] C. Song and A. Raghunathan, “Information Leakage in Embedding Models,” Aug. 19, 2020, *arXiv*: arXiv:2004.00053. doi: 10.48550/arXiv.2004.00053.
[26] H. Li, M. Xu, and Y. Song, “Sentence Embedding Leaks More Information than You Expect: Generative Embedding Inversion Attack to Recover the Whole Sentence,” May 04, 2023, *arXiv*: arXiv:2305.03010. doi: 10.48550/arXiv.2305.03010.
[27] “Server-Based Manipulation Attacks Against Machine Learning Models | Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy.” Accessed: Feb. 07, 2025. [Online]. Available: https://dl.acm.org/doi/10.1145/3176258.3176321
[28] X. Wang, J. Li, X. Kuang, Y.-A. Tan, and J. Li, “The security of machine learning in an adversarial setting: A survey,” *Journal of Parallel and Distributed Computing*, vol. 130, pp. 12–23, 2019, doi: 10.1016/j.jpdc.2019.03.003.
[29] F. A. Yerlikaya and Ş. Bahtiyar, “Data poisoning attacks against machine learning algorithms,” *Expert Systems with Applications*, vol. 208, p. 118101, Dec. 2022, doi: 10.1016/j.eswa.2022.118101.
[30] X. Chen, Y. Ma, and S. Lu, “Use Procedural Noise to Achieve Backdoor Attack,” *IEEE Access*, vol. 9, pp. 127204–127216, 2021, doi: 10.1109/ACCESS.2021.3110239.
[31] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, “The security of machine learning,” *Machine Learning*, vol. 81, no. 2, pp. 121–148, 2010, doi: 10.1007/s10994-010-5188-5.
[32] N. Nazari *et al.*, “Adversarial Attacks Against Machine Learning-Based Resource Provisioning Systems,” *IEEE Micro*, vol. 43, no. 5, pp. 35–44, Sep. 2023, doi: 10.1109/MM.2023.3267481.

[33] “Man-in-the-Middle Attacks Against Machine Learning Classifiers Via Malicious Generative Models | IEEE Journals & Magazine | IEEE Xplore.” Accessed: Feb. 07, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/9183938
[34] Z. Ye, W. Luo, M. L. Naseem, X. Yang, Y. Shi, and Y. Jia, “C2FMI: Corse-to-Fine Black-Box Model Inversion Attack,” *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 3, pp. 1437–1450, May 2024, doi: 10.1109/TDSC.2023.3285071.
[35] B. Rao, J. Zhang, D. Wu, C. Zhu, X. Sun, and B. Chen, “Privacy Inference Attack and Defense in Centralized and Federated Learning: A Comprehensive Survey,” *IEEE Transactions on Artificial Intelligence*, pp. 1–22, 2024, doi: 10.1109/TAI.2024.3363670.
[36] Q. L. Roux, E. Bourbao, Y. Teglia, and K. Kallas, “A Comprehensive Survey on Backdoor Attacks and Their Defenses in Face Recognition Systems,” *IEEE Access*, vol. 12, pp. 47433–47468, 2024, doi: 10.1109/ACCESS.2024.3382584.
[37] V. Battis and A. Penner, “Transformer-based Extraction of Deep Image Models,” in *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, Jun. 2022, pp. 320–336. doi: 10.1109/EuroSP53844.2022.00028.
[38] C. Park, Y. Kim, J.-G. Park, D. Hong, and C. Seo, “Evaluating Differentially Private Generative Adversarial Networks Over Membership Inference Attack,” *IEEE Access*, vol. 9, pp. 167412–167425, 2021, doi: 10.1109/ACCESS.2021.3137278.
[39] “Gradient-Leaks: Enabling Black-Box Membership Inference Attacks Against Machine Learning Models | IEEE Journals & Magazine | IEEE Xplore.” Accessed: Feb. 07, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10285881
[40] “Two Sides of the Same Coin: Boons and Banes of Machine Learning in Hardware Security | IEEE Journals & Magazine | IEEE Xplore.” Accessed: Feb. 07, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/9442769
[41] “Fingerprinting Classifiers With Benign Inputs | IEEE Journals & Magazine | IEEE Xplore.” Accessed: Feb. 07, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10201933
[42] “On the Security of Machine Learning in Malware C&C Detection: A Survey: ACM Computing Surveys: Vol 49, No 3.” Accessed: Feb. 07, 2025. [Online]. Available: https://dl.acm.org/doi/10.1145/3003816
[43] 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)*, Jul. 2021, pp. 147–154. doi: 10.1109/CSR51186.2021.9527917.
[44] “Dynamic Backdoor Attacks Against Machine Learning Models | IEEE Conference Publication | IEEE Xplore.” Accessed: Feb. 07, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/9797338
[45] J. Wang and P. Srikantha, “Stealthy Black-Box Attacks on Deep Learning Non-Intrusive Load Monitoring Models,” *IEEE Transactions on Smart Grid*, vol. 12, no. 4, pp. 3479–3492, Jul. 2021, doi: 10.1109/TSG.2021.3062722.
[46] “Exploring the Vulnerabilities of Machine Learning and Quantum Machine Learning to Adversarial Attacks Using a Malware Dataset: A Comparative Analysis | IEEE Conference Publication | IEEE Xplore.” Accessed: Feb. 07, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10234365
[47] “Model weights | Mistral AI Large Language Models.” Accessed: Apr. 07, 2025. [Online]. Available: https://docs.mistral.ai/getting-started/models/weights/
[48] *mistralai/mistral-inference*. (Apr. 06, 2025). Jupyter Notebook. Mistral AI. Accessed: Apr. 07, 2025. [Online]. Available: https://github.com/mistralai/mistral-inference

[49] “Bienvenue to Mistral AI Documentation | Mistral AI Large Language Models.” Accessed: Apr. 07, 2025. [Online]. Available: https://docs.mistral.ai/
[50] “mistral-inference/tutorials/getting_started.ipynb at main · mistralai/mistral-inference,” GitHub. Accessed: Apr. 07, 2025. [Online]. Available: https://github.com/mistralai/mistral-inference/blob/main/tutorials/getting_started.ipynb
[51] “mistral-inference/src/mistral_inference/generate.py at main · mistralai/mistral-inference,” GitHub. Accessed: Apr. 07, 2025. [Online]. Available: https://github.com/mistralai/mistral-inference/blob/main/src/mistral_inference/generate.py
[52] “mistral-common/src/mistral_common/protocol/instruct/request.py at main · mistralai/mistral-common · GitHub.” Accessed: Apr. 14, 2025. [Online]. Available: https://github.com/mistralai/mistral-common/blob/main/src/mistral_common/protocol/instruct/request.py
[53] *mistralai/client-python*. (Apr. 06, 2025). Python. Mistral AI. Accessed: Apr. 07, 2025. [Online]. Available: https://github.com/mistralai/client-python
[54] “GitHub - mistralai/client-ts: TS Client library for Mistral AI platform.” Accessed: Apr. 07, 2025. [Online]. Available: https://github.com/mistralai/client-ts
[55] “mistralai/Mistral-Small-3.1-24B-Base-2503 · Hugging Face.” Accessed: Apr. 07, 2025. [Online]. Available: https://huggingface.co/mistralai/Mistral-Small-3.1-24B-Base-2503
[56] “Mistral Small 3.1 | Mistral AI.” Accessed: Apr. 07, 2025. [Online]. Available: https://mistral.ai/news/mistral-small-3-1
[57] “Cloud | Mistral AI Large Language Models.” Accessed: Apr. 07, 2025. [Online]. Available: https://docs.mistral.ai/deployment/cloud/overview/
[58] “La Plateforme - frontier LLMs | Mistral AI.” Accessed: Apr. 07, 2025. [Online]. Available: https://mistral.ai/products/la-plateforme
[59] “mistral-common/ at main · mistralai/mistral-common · GitHub.” Accessed: Apr. 14, 2025. [Online]. Available: https://github.com/mistralai/mistral-common/tree/main
[60] “Welcome to vLLM — vLLM.” Accessed: Apr. 14, 2025. [Online]. Available: https://docs.vllm.ai/en/latest/index.html
[61] “Fine-tuning | Mistral AI Large Language Models.” Accessed: Apr. 07, 2025. [Online]. Available: https://docs.mistral.ai/guides/finetuning/
[62] *mistralai/mistral-finetune*. (Apr. 06, 2025). Python. Mistral AI. Accessed: Apr. 07, 2025. [Online]. Available: https://github.com/mistralai/mistral-finetune
[63] “Moderation | Mistral AI Large Language Models.” Accessed: Apr. 07, 2025. [Online]. Available: https://docs.mistral.ai/capabilities/guardrailing/
[64] “Observability | Mistral AI Large Language Models.” Accessed: Apr. 07, 2025. [Online]. Available: https://docs.mistral.ai/guides/observability/
[65] “Mistral AI API | Mistral AI Large Language Models.” Accessed: Apr. 27, 2025. [Online]. Available: https://docs.mistral.ai/api/
[66] “JailbreakV-28K/JailBreakV-28k · Datasets at Hugging Face.” Accessed: Apr. 27, 2025. [Online]. Available: https://huggingface.co/datasets/JailbreakV-28K/JailBreakV-28k
[67] Y. Feng, B. Ma, J. Zhang, S. Zhao, Y. Xia, and D. Tao, “FIBA: Frequency-Injection based Backdoor Attack in Medical Image Analysis,” in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2022, pp. 20844–20853. doi: 10.1109/CVPR52688.2022.02021.
[68] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, “Characterizing and Evaluating Adversarial Examples for Offline Handwritten Signature Verification,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 8, pp. 2153–2166, Aug. 2019, doi: 10.1109/TIFS.2019.2894031.

[69] “A Wolf in Sheep’s Clothing: Query-Free Evasion Attacks Against Machine Learning-Based Malware Detectors with Generative Adversarial Networks | IEEE Conference Publication | IEEE Xplore.” Accessed: Feb. 07, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10190713
[70] Y. Qin and C. Yue, “Fuzzing-based hard-label black-box attacks against machine learning models,” *Computers & Security*, vol. 117, p. 102694, Jun. 2022, doi: 10.1016/j.cose.2022.102694.
[71] “SecureDroid | Proceedings of the 33rd Annual Computer Security Applications Conference.” Accessed: Feb. 07, 2025. [Online]. Available: https://dl.acm.org/doi/10.1145/3134600.3134636

Appendices

I. Glossary

AI	Artificial Intelligence
API	Application Programming Interface
BPMN	Business Process Model and Notation
CIA	Confidentiality, Integrity, Availability
CPU	Central Processing Unit
CPS	Cyber-Physical Systems
DoS	Denial of Service
DoW	Denial of Wallet
DL	Deep Learning
DNN	Deep Neural Network
DP	Differential Privacy
DSR	Design Science Research
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
HT	Hardware Trojan
IP	Intellectual Property
IS assets	Information System assets
ISSRM	Information Systems Security Risk Management
IT	Information Technology
LLM	Large Language Model
ML	Machine Learning
RAG	Retrieval-Augmented Generation
RAM	Random Access Memory
REST	Representational State Transfer
SLR	Systematic Literature Review
STRIDE	Spoofing, Tampering, Repudiation, Information disclosure, Denial of service Elevation of privilege
UAV	Unmanned Aerial Vehicle
UML	Unified Modeling Language

II. Systematic Literature Review Extracted Data

The extracted data from the systematic literature review can be found in the form of an Excel file, on the "threats" sheet in the following repository: https://github.com/CybRookie/llm_threat_modeling_research_artifacts. A copy of the table's subset is provided below. Entries marked with an asterisk - *, mean that the content was not directly stated within the contents of the reviewed paper, but was derived by the author of this thesis from the other content and the present context. Fields with the entry of "None" mean that no suitable content was found within the reviewed paper, according to the criteria, as well as it was not possible to derive an entry from the present context.

Table 13. A table of the extracted data from the literature review.

Attack name(-s)	System Asset	Business Asset	Security Criteria	Vulnerability	Threat Agent	Attack Method	Impact and Harm	Security Requirements	Security Control
(Poisoning attack): Backdoor attack [67]	Machine learning training system*	Training data	Integrity*, Availability*	1. The training dataset is susceptible to unauthorized modifications.* 2. The model is incapable to distinguish original and poisoned samples, when subtle frequency-domain modifications are introduced.	Attacker: white-box scenario*	Frequency space signals of the trigger and benign images are obtained with the Fast Fourier Transform (FFT). Trigger data is injected into the amplitude spectrum of the original, benign image from the trigger image's amplitude spectrum.	Negates the integrity and availability of the targeted machine learning model. This leads to misclassification of malicious input.*	The machine learning system must be resistant to backdoor attacks.*	None.
Poisoning attack* [46]	Machine learning training system*	Input data	Integrity*	1. The training dataset is susceptible to unauthorized modifications.* 2. The target model's performance can be skewed through training on malicious samples.*	Attacker: white-box scenario*	Addition of the malicious training samples into the dataset and target model retraining on the poisoned dataset. The malicious samples are produced by random generation of noise and its addition to the original samples.	Negates the integrity of the targeted machine learning model. This leads to misclassification of malicious input.*	The machine learning system must be resistant to poisoning attacks.*	None. The analyzed Quantum Neural Network (QNN) was more resistant to the attack than the conventional neural networks.
Poisoning attack [22]	Machine learning training system*	Training data	Integrity*, Availability*	1. The training dataset is susceptible to unauthorized modifications.* 2. The target model's performance can be skewed through training on malicious samples.*	Attacker: white-box scenario* (access to the training dataset)	Poisoning sample that maximizes class likelihood is crafted with a heuristic-based poisoning method that uses kernel density estimation (KDE) and linear combinations of existing samples. The poisoning sample is added to the target training dataset.	Negates the integrity and availability of the targeted machine learning model. This leads to misclassification of malicious input.*	The machine learning system must be resistant to poisoning attacks.*	A discriminator-based defense approach, using a GAN's discriminator to identify and filter out poisoning samples, preserving model integrity.
Adversarial Attack [32]	ML system input/API*	Input data	Integrity*	There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model.*	Attacker: black-box scenario	Initially the target model is reverse engineered, a shadow model is created. Adversarial samples are generated with the use of gradient-based methods (e.g., FGSM-like approach) and created shadow model. The adversarial samples are submitted to the target model to cause malfunction in its classification.	Negates the integrity of the targeted machine learning model. This may lead to misclassification of benign and malicious inputs.*	The machine learning system must be resistant to adversarial attacks.*	1. Adversarial training: using adversarial examples in training. 2. Data randomization, modification of the input data. 3. Data compression of the input. 4. Addition of a masking layer to control dominant weights, thus reducing model's sensitivity. 5. Regularization of model based off of model's outputs and inputs during training. 6. Feature squeezing, utilize multiple models to determine if the sample is adversarial. 7.Utilise generative adversarial networks to train a model. 8.Rectification, addition of a "pre-input" layer, acting as a perturbation detector.
Adversarial Attack [7]	ML system input/API*	Input data	Integrity*	There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model.*	Adversary: White and Black box scenarios	A submission of an adversarial sample. A set of suitable perturbations is determined by solving a constrained optimization problem. Afterwards, the set of perturbations is reduced based on imposed spatial and physical constraints. The suitable perturbation is applied to the target setting, resulting in an adversarial sample.	Negates the integrity of the targeted machine learning model.* In the context of the infrastructure inspection, this could lead to severe infrastructure safety concerns.	The machine learning system must be resistant to adversarial attacks.*	Adversarial training: inclusion of adversarial samples into the training dataset, thus training to minimize the loss from the inclusion of adversarial samples.
Model inversion attack [34]	ML system input/API*	Input data*	Confidentiality*	It is possible to recover features of the training data based of predictions retrieved from the target machine learning model.*	Adversary: black - box scenario; white-box scenario for feature extractor	1. A custom feature extractor is trained upon auxiliary data. 2. The public, auxiliary data is fed to the feature extractor and the target model to produce feature-prediction probability pairs. 3. An inverse model is trained on the feature-prediction probability data, mapping prediction data to the feature space. 5. The inverse model produces a feature based on the provided label data. 4. Output of a GAN model (trained on public, auxiliary data) is produced based on the inverse model's generated features. 5. The produced sample by GAN is fed to the target model to acquire new predictions. 6. Features are extracted from the generated image and compared to the features previously produced by the inverse model, accounting for the new predictions. 7. Thus, the features are updated iteratively with differential evolution (DE) optimization algorithm. 8. A new image is generated with the GAN model based on the updated features.	Negates the confidentiality of the targeted machine learning model. This may lead to leakage of sensitive data. Exposure of the sensitive private data may lead to legal repercussions.*	The machine learning system must be resistant to model inversion attacks.*	1. Degradation of precision accuracy. This may result in slight degradation of the attack's performance. 2. The prediction results can be rounded down or replaced with null data based on the threshold. This results in degradation of the attack's performance.
Model extraction attack [37]	ML system input/API*	Input data*	Confidentiality*	It is possible to replicate target model's properties within a new model based on the target machine learning model's output.*	Adversary: black-box scenario	The target model is queried with surrogate data, based on the public data, creating a dataset of surrogate data-target model output data pairs. A publicly available pre-trained Data-efficient Transformer (DeiT) model is fine-tuned based on the previously produced dataset.	Negates the confidentiality of the targeted machine learning model. This may lead to the loss of intellectual property.*	The machine learning system must be resistant to model extraction attacks.*	1. Prediction poisoning, addition of bounded noise to the model's output for maximization of the annular deviation (MAD) between the original gradients and poisoned ones, while maintaining the rank of the most confident prediction, to prevent accuracy loss. 2. Reverse Sigmoid, addition of an activation layer, which controls the amount of perturbation added to the target model's posterior probabilities, to prevent loss of the model's accuracy .
Adversarial attack [45]	ML system input/API	Input data	Integrity*	1. There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model.* There exists inherent ambiguity between the decision boundaries of a machine learning model and true decision boundaries. 2. A system flaw, that can be accessed and exploited externally.	Adversary: black-box scenario	The target model is queried limited amount of times. The retrieved outputs are utilized to train a substitute model (feed-forward neural network (FFNN)), while augmenting data with a custom augmentation algorithm and further retraining the model. Adversarial sample re-crafted from the trained substitute model. Adversarial sample generation accounts for the cost gradients of the substitute model, the degree of	Negates the integrity of the targeted machine learning model. This may lead to the monetary losses and cascading outages.*	The machine learning system must be resistant to adversarial attacks.*	None.

						perturbations is limited. The produced adversarial samples are submitted to the target model.			
(Poisoning attack*): Backdoor attack [30]	Machine learning training system	Training data	Integrity*	1. The training dataset is susceptible to unauthorized modifications.* 2. The model 's classification boundary is prone to compromise, if trained on malicious data samples.*	Attacker: white-box scenario*	Procedural noise is crafted using algorithms, such as Perlin noise, Gabor noise, Worley noise. A clean model is trained on original images to get the attention images. The attention images are fused with noise, and then added to the original image. The target model is trained on the poisoned data, turning into a backdoored model.	Negates the integrity of the targeted machine learning model. This may lead to collateral damage due to incorrect operation of the machine learning system.*	The machine learning system must be resistant to backdoor attacks.*	Activate clustering: a detection method splitting feature representation into a poisoned and clean clusters. The last hidden layer can detect differences among cluster's high-level features. Perlin and Gabor noise avoid detection, while Worley noise gets detected.
Adversarial attack [68]	ML system input/API*	Input data	Integrity*, Availability*	There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model.*	Adversary: Grey box scenario (limited knowledge scenario, no access to the training dataset)	An adversarial sample can be produced through one of the four algorithms, two gradient-based - Carlini & Wagner, Fast Gradient Method, and two gradient-free - Decision-based attack, Simulated Annealing.	Negates the integrity and availability of the targeted machine learning model. This leads to misclassification of benign and malicious inputs.*	The machine learning system must be resistant to adversarial attacks.*	1. Ensemble Adversarial Learning provided robustness against Fast Gradient Method. Carlini attack remains successful. 2.Madry defense involves a saddle point optimization problem, optimizing for the worst case. Increases the amount of noise needed to turn a sample into an adversarial one. Improved model's robustness against Fast Gradient Method and increased the amount of noise that is needed for Carlini method to succeed.
Membership inference attack [38]	ML system input/API of the GAN's discriminator model*	Input data	Confidentiality*	It is possible to learn additional information about the training data sample from the target model's output.	Attacker: white-box scenario (attacker has access to the discriminator of the target GAN model)	GAN's discriminator model is used to deduce whether a particular data instance was part of the training dataset of a target model. This is achieved by exploiting differences in the model's response to known (trained on) versus unknown input data.	Negates the confidentiality of the targeted machine learning model. This leads to privacy leakage and potential legal repercussions.*	The machine learning system must be resistant to membership inference attacks.*	The defensive mechanism utilized in this study is Differential Privacy (DP), applied to Generative Adversarial Networks (GANs) to protect against privacy invasion attacks, specifically membership inference attacks. Differential privacy introduces controlled noise into the model training process to obscure the presence or absence of any single data point in the training set, thus aiming to protect individual data privacy.
Membership inference attack [38]	ML system input/API of the GAN's generator model*	Input data	Confidentiality*	It is possible to learn additional information about the training data sample from the target model's output.	Attacker: partial white-box scenario (attacker has access to the generative model of the target GAN and latent code)	Synthesis of generated data through optimization against the latent code. The distance between the generated data and target data is measured to determine membership.	Negates the confidentiality of the targeted machine learning model. This leads to privacy leakage and potential legal repercussions.*	The machine learning system must be resistant to membership inference attacks.*	Differential privacy: calibrated noise is added to the gradients during training, applied gradient clipping to bound the sensitivity of the training algorithm.
Malicious hardware fault injection: fault sneaking attack, bit-flip attack (examples: rowhammer attack, ram-jam attack, laser beaming, clock glitch injection) [40]	Processing hardware running the ML model	Model's operational data	Integrity*	The state of machine learning model's operational data can be influenced through hardware state manipulation with fault injection.*	Attacker: black-box and white-box scenarios*	A malicious fault is introduced into the operations of hardware, processing target machine learning model's operations.	Negates the integrity of the targeted machine learning model. This may lead to misclassification of benign and malicious inputs.*	The machine learning system must be resistant to malicious hardware fault injection attacks.*	MAC operations, low-precision data representations, bound-constrained dynamic range compression: limit error propagation and aggregation; quantizing data to lower bit precision reduces the proportion of vulnerable parameters. Activation output clipping. Binarization method: mimic bit-flip noise on the weights, thus increasing robustness against bit-flip attacks. Piece-wise clustering method: adds fixed single bit-width constraint during the training, this increasing robustness against bit-flip attacks. Weight reconstruction: averages errors over a grain of weights with their quantization and clipping, thus increasing robustness against bit-flip attacks. Defensive quantization: constrains the Lipschitz constant during training to limit mapping sensitivity. Hardware with Triple Modular Redundancy (TMR): three copies of the functional circuits are present, majority vote determines correction and masking of faults in copied; imposes higher energy and resource overhead. DNN accelerator: tolerant to SRAM read faults from voltage variations. Word masking and bit masking: round faulty bits to zero; a whole word reset to zero or flipped bits are reset to zero respectively. TE-Drop: an error-tolerant design for the MAC units, for example utilizing Razor flip flops module for active fault detection; the detected error is dropped. Hardening of selective memory cells. Application of modular redundancy on sensitive weights. Possible direction - explainable AI, if both inductive and deductive reasonings are incorporated together, this could reduce frequency of logical fallacies.
Hardware Trojan attack [40]	Processing hardware running the ML model	Model's operational data	Confidentiality*, Integrity*, Availability*	The state of machine learning model's operational data can be influenced through a hardware trojan inserted into the integrated circuits.*	Attacker: black-box scenario (relative to the machine learning model)*	A hardware trojan is embedded into an integrated circuit. The embedded hardware trojan modifies the chain of operations based of a trigger, thus modifying the expected chain of operations during machine learning operation.	Negates the confidentiality, integrity and availability of the targeted machine learning model. This may lead to misclassification of benign and malicious inputs, degraded performance or private data leakage.*	The machine learning system must be resistant to malicious hardware trojan attacks.*	MAC (Multiply-and-Accumulate) operations, low-precision data representations, bound-constrained dynamic range compression: limit error propagation and aggregation; quantizing data to lower bit precision reduces the proportion of vulnerable parameters. Activation output clipping. Binarization method: mimic bit-flip noise on the weights, thus increasing robustness against bit-flip attacks. Piece-wise clustering method: adds fixed single bit-width constraint during the training, this increasing robustness against bit-flip attacks. Weight reconstruction: averages errors over a grain of weights with their quantization and clipping, thus increasing robustness against bit-flip attacks. Defensive quantization: constrains the Lipschitz constant during training to limit mapping sensitivity. Hardware with Triple Modular Redundancy (TMR): three copies of the functional circuits are present, majority vote determines correction and masking of faults in copied; imposes higher energy an resource overhead. DNN (Deep Neural Networks) accelerator: tolerant to SRAM read faults from voltage variations. Word masking and bit masking: round faulty bits to zero; a whole word reset to zero or flipped bits are rest to zero respectively. TE-Drop: an error-tolerant design for the MAC units, for example utilizing Razor flip flops module for active fault detection; the detected error is dropped. Hardening of selective memory cells. Application of modular redundancy on sensitive weights. Hardware root-of-trust: safeguarding DNN IP cores and private data. An

									obfuscation framework employing key-dependent backpropagation algorithm to lock some neurons of the model; on-chip key can recover the correct functionality of the model. Trusted Inference Engine (TIE): Pseudo Random Number Generators (PRNG) and PUF (Physically Unclonable Function) are utilized to decrypt the encrypted machine learning model, stored on off-chip memory. A DNN accelerator with a memory encryption engine, encrypting data in DRAM; also utilizing Integrity Verification (IV) engine for detection of unauthorized operations on the data from the external memory; comes with low overhead (this defense does not account for hardware side-channels). Possible direction - explainable AI, if both inductive and deductive reasonings are incorporated together, this could reduce frequency of logical fallacies.
Side-channel attack (utilized for model extraction) [40]	Processing hardware running the ML model	Machine learning model components (i.e. layers, type of activation function, layer connections, parameters)	Confidentiality*	The state of machine learning model's operational component data can be inferred through measurable attributes of the machine learning, such as power consumption, EM emission, memory access pattern.*	Attacker: black-box scenario (relative to the machine learning model)*	Measurable attributes of hardware, processing target machine learning model's operations, are observed. The observed and measured metrics are analyzed to infer the machine learning model's structure.	Negates the confidentiality of the targeted machine learning model. This may lead to the loss of the intellectual property.*	The machine learning system must be resistant to malicious side-channel attacks.*	Internal operation shuffling: order of execution is mixed to modify the scheduled operation time. Computation masking: taints sensitive operations with random values, to eliminate dependencies between the private data and the side-channel attributes. Augmenting masking: masks adder trees and ReLU (Rectified Linear Unit). BoMaNet masking: uses gate-level Boolean masking for splitting secrets, thus reducing relation between secret related computations and side-channel attributes. Data quantization: mitigates weight matrices leakage based of off cache access patterns. Cache partitioning: distinct portions of the last-level cache are allocated to different applications to eliminate cache interference between the attacker and the victim. Reduction of hardware profiler's precision: reduces side-channel leakage from context-switching penalties. Oblivious RAM (ORAM): reduces side-channel information leakage based of off memory patterns and timing, shuffles and re-encrypts the data to conceal access pattern. Memory-Trace Obliviousness (MTO): reduces side-channel information leakage based of off memory patterns an timing. Creation of fake memory access with TIE (Trusted Inference Engine). Randomization of the width of the coalescing unit and merge of transactions: mitigates GPU memory timing side-channel attacks. GPUGuard: detects spy programs through a decision tree method, thus mitigating side-channel attacks. Possible direction - explainable AI, if both inductive and deductive reasonings are incorporated together, this could reduce frequency of logical fallacies.
Side-channel attack (utilized for model extraction, leaking inference data) [40]	Processing hardware running the ML model	Input data	Confidentiality*	The state of machine learning model's operational component data can be inferred through measurable attributes of the machine learning, such as power consumption, EM emission, memory access pattern.*	Attacker: black-box scenario (relative to the machine learning model)*	Measurable attributes of hardware, processing target machine learning model's operations, are observed. The observed and measured metrics are analyzed to infer the input, submitted to the target machine learning model for processing.	Negates the confidentiality of the targeted machine learning model. This may lead to the private data leakage and possible legal repercussions.*	The machine learning system must be resistant to malicious side-channel attacks.*	Internal operation shuffling: order of execution is mixed to modify the scheduled operation time. Computation masking: taints sensitive operations with random values, to eliminate dependencies between the private data and the side-channel attributes. Augmenting masking: masks adder trees and ReLU (Rectified Linear Unit). BoMaNet masking: uses gate-level Boolean masking for splitting secrets, thus reducing relation between secret related computations and side-channel attributes. Data quantization: mitigates weight matrices leakage based of off cache access patterns. Cache partitioning: distinct portions of the last-level cache are allocated to different applications to eliminate cache interference between the attacker and the victim. Reduction of hardware profiler's precision: reduces side-channel leakage from context-switching penalties. Oblivious RAM (ORAM): reduces side-channel information leakage based of off memory patterns and timing, shuffles and re-encrypts the data to conceal access pattern. Memory-Trace Obliviousness (MTO): reduces side-channel information leakage based of off memory patterns an timing. Creation of fake memory access with TIE (Trusted Inference Engine). Randomization of the width of the coalescing unit and merge of transactions: mitigates GPU memory timing side-channel attacks. GPUGuard: detects spy programs through a decision tree method, thus mitigating side-channel attacks. Possible direction - explainable AI, if both inductive and deductive reasonings are incorporated together, this could reduce frequency of logical fallacies.
Membership inference attack: shadow training method [35]	ML system input/API*	Training data*	Confidentiality*	It is possible to learn additional information about the training data sample from the target model's output.* Model tends to memorize its training data in case of over-parametrization.	Attacker: black-box and white-box scenarios	Multiple shadow models are built imitating the target model's structure and training process. These shadow models are trained on datasets where the membership status is known. The outputs (e.g., confidence scores) are collected from these shadow models for both member and non-member data. An attack model is then trained to distinguish between members and non-members based on these outputs. The attacker queries the target model with data of unknown membership and uses the attack model to infer membership status against the target model, covering the same domain.	Negates the confidentiality of the targeted machine learning model. This may lead to the private data leakage and possible legal repercussions.*	The machine learning system must be resistant to malicious membership inference attacks.*	Differential privacy: addition of noise to deviate the outputs from the original. Secure multi-party computation: joint computations are conducted within confidential environment. Homomorphic encryption: calculations are conducted through confidential means, allowing operations on encrypted data without revealing the original data. Adversarial machine learning: incorporation of data about adversarial techniques into the model's training process. Vulnerability detection: risk assessment method for machine learning mode, evaluation of the models pre-release.
Membership inference attack: metric-based method [35]	ML system input/API*	Training data*	Confidentiality*	Model tends to memorize its training data in case of over-parametrization. It is possible to learn additional information about the training data sample from the target model's output.*	Attacker: black-box and white-box scenarios	Observer target machine learning model's behavior on training data versus unseen data. Analysis of metrics such as confidence scores, loss values, or entropy, patterns indicative of membership are identified. Statistical thresholds or anomalies in the metrics are utilized to determine the membership of the target sample. Membership is inferred when the metrics for a	Negates the confidentiality of the targeted machine learning model. This may lead to the private data leakage and possible legal repercussions.*	The machine learning system must be resistant to malicious membership inference attacks.*	Differential privacy: addition of noise to deviate the outputs from the original. Secure multi-party computation: joint computations are conducted within confidential environment. Homomorphic encryption: calculations are conducted through confidential means, allowing operations on encrypted data without revealing the original data. Adversarial machine

						given input differ significantly from those expected for non-members.			learning: incorporation of data about adversarial techniques into the model's training process. Vulnerability detection: risk assessment method for machine learning mode, evaluation of the models pre-release.
Model inversion attack (may involve single-sample label inference attack, single-sample reconstruction attack, multi-sample label distribution estimation attack, and multi-sample reconstruction attack) [35]	ML system input/API*, model's parameters	Training data	Confidentiality*	Model tends to memorize its training data in case of over-parametrization. It is possible to learn additional information about the training data sample from the target model's output.*	Attacker: black-box and white-box scenarios	One of the methods: outputs of different model versions are recorded; Multi-Layer Perceptron is utilized to analyze the difference between the version outputs, producing information about the target training data. Other possible techniques: LSB encoding; Correlated value encoding; Sign encoding, Using Model Overfitting, Neuron sorting; Set-based representation, Training-based strategy values, Recognition related neuron, Training attack classifier, Training Shadow GAN, Poisoning attack, Analysis of the confidence score , Training a meta-classifier, Design a multi-task GAN, Regularized Maximum Likelihood Estimation; Inverse-Network, Deep Leakage from Gradients, Numerical reconstruction Matching virtual and shared gradients, Equality solving; Path Restriction, Direct/passive/active label inference attack.	Negates the confidentiality of the targeted machine learning model. This may lead to the private data leakage and possible legal repercussions.*	The machine learning system must be resistant to malicious model inversion attacks.*	Differential privacy: addition of noise to deviate the outputs from the original. Secure multi-party computation: joint computations are conducted within confidential environment. Homomorphic encryption: calculations are conducted through confidential means, allowing operations on encrypted data without revealing the original data. Adversarial machine learning: incorporation of data about adversarial techniques into the model's training process. Vulnerability detection: risk assessment method for machine learning mode, evaluation of the models pre-release.
Property inference attack [35]	ML system input/API*	Input data*	Confidentiality*	Model tends to memorize its training data in case of over-parametrization. It is possible to learn additional information about the training data sample from the target model's output.*	Attacker: black-box and white-box scenarios	Target model's output and parameters are analyzed to determine properties of the utilized training dataset.	Negates the confidentiality of the targeted machine learning model. This may lead to the private data leakage and possible legal repercussions.*	The machine learning system must be resistant to malicious property inference attacks.*	Differential privacy: addition of noise to deviate the outputs from the original. Secure multi-party computation: joint computations are conducted within confidential environment. Homomorphic encryption: calculations are conducted through confidential means, allowing operations on encrypted data without revealing the original data. Adversarial machine learning: incorporation of data about adversarial techniques into the model's training process. Vulnerability detection: risk assessment method for machine learning mode, evaluation of the models pre-release.
Model extraction attack [35]	ML system input/API*	Input data*	Confidentiality	It is possible to possible to reconstruct target model's functionality from the target model's output.*	Attacker: white-box, black-box and gray-box scenarios	One of the possible methods: A set amount of requests is produced against the target model, producing a set of responses. A surrogate model is trained based on request data-response data pairs. Other possible techniques: Linear least square approach, Malicious samples query, Build universal thief datasets , Query synthesis active learning, Autoregressive generation, Fine-tuned encoder - Algebraic attack, Direct Extraction - Recreate Projection, Head Fuzzy gray correlation.	Negates the confidentiality of the targeted machine learning model. This may lead to the intellectual property theft.*	The machine learning system must be resistant to model extraction attacks.*	Differential privacy: addition of noise to deviate the outputs from the original. Secure multi-party computation: joint computations are conducted within confidential environment. Homomorphic encryption: calculations are conducted through confidential means, allowing operations on encrypted data without revealing the original data. Adversarial machine learning: incorporation of data about adversarial techniques into the model's training process. Watermarking techniques: embedding of watermarks into model's parameters, or algorithmic analysis of the model due to over-parametrization, or entanglement between watermark and training data features. Vulnerability detection: risk assessment method for machine learning mode, evaluation of the models pre-release.
Model extraction attack: side-channel attack [35]	Processing hardware running the ML model	Model's parameters	Confidentiality*	The state of machine learning model's operational component data can be inferred through measurable attributes of the machine learning, such as power consumption, EM emission, memory access pattern.*	Attacker: gray-box scenario	A malicious fault is introduced into the operations of hardware, processing target machine learning model's operations.*	Negates the confidentiality of the targeted machine learning model. This may lead to the intellectual property theft.*	The machine learning system must be resistant to model extraction attacks.*	Differential privacy: addition of noise to deviate the outputs from the original. Secure multi-party computation: joint computations are conducted within confidential environment. Homomorphic encryption: calculations are conducted through confidential means, allowing operations on encrypted data without revealing the original data. Adversarial machine learning: incorporation of data about adversarial techniques into the model's training process. Watermarking techniques: embedding of watermarks into model's parameters, or algorithmic analysis of the model due to over-parametrization, or entanglement between watermark and training data features. Vulnerability detection: risk assessment method for machine learning mode, evaluation of the models pre-release.
Adversarial attack*: Evasion attack [3]	ML system input/API*	Input data*	Integrity	There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model.*	Attacker: black-box and white-box scenarios	Iterative probing, gradient estimation, or using a surrogate model to craft adversarial examples. Characteristics of a sample are manipulated to turn it into an adversarial sample. Adversarial sample are submitted to the model to cause misclassification.	Negates the integrity of the targeted machine learning model. This may lead to misclassification of benign and malicious inputs.*	The machine learning system must be resistant to adversarial attacks.*	1. Randomization - introduction of unpredictability into model's responses. Example approach: randomly generate and train multiple classifiers using different subsets of the feature space. The final output is aggregated from the predictions of all classifiers. 2. Complexity - complexity of model's decision function is increased, for example the boundary can be made non-linear or be fractalized. 3. Stateful analysis - query analysis, maintenance of query history and its analysis with meta-detectors.
Model extraction [3]	ML system input/API*	Input data*	Confidentiality	It is possible to replicate target model's properties within a new model based on the target machine learning model's output.*	Attacker: black-box and gray-box scenarios	The target model is probed, the received output is utilized to build a clone model. The cloned model can be utilized for further attacks, such as an evasion attack.	Negates the confidentiality of the targeted machine learning model. This may lead to the loss of intellectual property.*	The machine learning system must be resistant to model extraction attacks.*	1. Randomization - introduction of unpredictability into model's responses. Example approach: randomly generate and train multiple classifiers using different subsets of the feature space. The final output is aggregated from the predictions of all classifiers. 2. Complexity - complexity of model's decision function is increased, for example the boundary can be made non-linear or be fractalized. 3. Stateful analysis - query analysis, maintenance of query history and its analysis with meta-detectors.
Fingerprinting attack [41]	ML system input/API*	Intellectual property (IP)*	Confidentiality*	The model's unique decision boundaries and output patterns serve as a "fingerprint"; benign inputs reveal characteristic outputs.	Attacker: black-box scenario	A set of benign queries are sent to the target set of models. The responses from models are compared to the known response from the targeted model for statistical similarity.	Negates the confidentiality of the targeted machine learning model.*	The machine learning system must be resistant to fingerprinting attacks.*	1. Randomized smoothing - addition of noise to the input, output classes are aggregated. 2. Pruning of the last layer.
Adversarial attack: perturbation [5]	ML system input/API*	Input data	Integrity*	There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model.*	Attacker: black-box scenario	Requests are made to the target model and the model's responses are collected. Collected samples can be used to create perturbations for adversarial attacks with universal adversarial perturbation (UAP) method. Alternatively, a surrogate model is trained on responses, it is used for estimating target responses; Basic Iterative Method (BIM) is used to produce	Negates the integrity availability of the targeted machine learning model. This leads to misclassification of malicious input or reduced performance.	The machine learning system must be resistant to adversarial attacks.*	Scrambling: an operation for radio unit (RU) ordering during training and inference stages to mitigate adversarial attacks by obfuscating AI model input relationships, significantly reducing attack effectiveness. This approach is not possible in other domain, such as image, text or audio, where semantics of the data would be lost.

						perturbations. Perturbations are used for creation adversarial samples.			
Adversarial attack: perturbation [5]	ML system input/API*	Input data	Integrity*	There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model.*	Attacker: white-box scenario	With the knowledge of the target model, adversarial examples can be directly produced with Basic Iterative Method (BIM).	Negates the integrity of the targeted machine learning model. This leads to misclassification of malicious input or reduced performance.	The machine learning system must be resistant to adversarial attacks.*	Scrambling: an operation for radio unit (RU) ordering during training and inference stages to mitigate adversarial attacks by obfuscating AI model input relationships, significantly reducing attack effectiveness. This approach is not possible in other domain, such as image, text or audio, where semantics of the data would be lost.
Membership inference attack [39]	ML system input/API*	Input data	Confidentiality*	1. There is a difference in gradient behavior between training (member) and non-training (non-member) records, which reflects membership information due to overfitting or gradient convergence during training. 2. The more output classes the model has, the more data is leaked.	Attacker: black-box scenario	The target model is requested with perturbed features. With the target model's outputs toward requests a local linear regression model is trained. An autoencoder is used to extract membership features from the approximated gradients for training of the local attack model, which would be used to classify, if the target sample is a member of the target model's training dataset or not.	Negates the confidentiality of the targeted machine learning model. This may lead to the private data leakage and possible legal repercussions.*	The machine learning system must be resistant to membership inference attacks.*	None.
Adversarial attack: evasion attack [69]	ML system input/API*	Input data*	Integrity*, Availability*	There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model.*	Attacker: white-box and black-box scenarios	The new approach uses GANs to generate adversarial samples that resemble benign files in feature space without needing access to the model's queries or internal structure. The GAN consists of a generator network that creates realistic malware features by transforming malicious features into benign-looking distributions, and a critic network that assesses these features' benignness. This method performs well in evading detection by state-of-the-art ML detectors, including VirusTotal, without requiring queries.	Negates the integrity of the targeted machine learning model. This leads to misclassification of malicious input.*	The machine learning system must be resistant to adversarial attacks.*	None.
Adversarial attack: Man-in-the-Middle attack [33]	Machine learning model	Input data	Integrity*	There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model.*	Attacker: white-box and black-box scenarios	1. Train a Variational Autoencoder (VAE) with a malicious decoder (MVD). 2. MVD is fine-tuned to produce adversarial samples. 3. Integrate the VAE with the trained MVD into the chain either between raw input data source and the classifier, or swapping out the existing decoder with MVD from the present VAE. 4. As data passes through, the MVD transforms encoded representation of the input data into an adversarial example.	Negates the integrity of the targeted machine learning model. This leads to misclassification of malicious input.*	The machine learning system must be resistant to adversarial attacks.*	1. Randomization - a technique, randomizing model's input. 2. Adversarial training: the model is trained against adversarial samples, to increase it robustness.
Poisoning attack*: Backdoor attack: random backdoor [44]	Machine learning training system*	Training data	Integrity*	The training dataset is susceptible to unauthorized modifications.* Training data directly influences the performance of the machine learning mode.*	Attacker: grey-box scenario* (the attacker is able to modify the training set)	A random trigger is sampled from uniform distribution. The random trigger is added at random locations within original samples with a target label.	Negates the integrity of the targeted machine learning model. This leads to misclassification of malicious input.*	The machine learning system must be resistant to backdoor attacks.*	Autoencoder-based reconstruction: an autoencoder, trained on clean data, encodes the training data (compresses it) and then decodes it back into training data (decompresses it).
Poisoning attack*: Backdoor attack: Backdoor Generating Network (BaN) [44]	Machine learning training system*	Training data	Integrity*	The training dataset is susceptible to unauthorized modifications.* Training data directly influences the performance of the machine learning mode.*	Attacker: grey-box scenario* (the attacker is able to modify the training set)	A Backdoor Generating Network (BaN) is created: BaN is a generative model (GAN based) that algorithmically creates backdoor triggers instead of relying on fixed patterns or random sampling. BaN is jointly trained with the target model (acts as a discriminator), which optimizes the trigger patterns. The BaN produces a trigger, it is provided as an input to the target model. Received loss from the backdoored sample is used with losses from benign samples to retrain the BaN. When BaN is ready, it produces a trigger that is added to the target sample in particular location.	Negates the integrity of the targeted machine learning model. This leads to misclassification of malicious input.*	The machine learning system must be resistant to backdoor attacks.*	Autoencoder-based reconstruction: an autoencoder, trained on clean data, encodes the training data (compresses it) and then decodes it back into training data (decompresses it).
Poisoning attack*: Backdoor attack: Backdoor Generating Network (c-BaN) [44]	Machine learning training system*	Training data	Integrity*	The training dataset is susceptible to unauthorized modifications.* Training data directly influences the performance of the machine learning mode.*	Attacker: grey-box scenario* (the attacker is able to modify the training set)	Backdoor Generating Network (c-BaN): a modified BaN for generation of label-specific triggers. The c-BaN model uses both the target label and a noise vector as inputs, creating triggers that can correspond to any target label, and that can be positioned at any location within the input space. This enables shared locations for triggers among different target labels, enhancing the stealthiness of the backdoor. The c-BaN is used to generate a backdoor trigger that is added to a target sample, which is to be submitted to the target model.	Negates the integrity of the targeted machine learning model. This leads to misclassification of malicious input.*	The machine learning system must be resistant to backdoor attacks.*	Autoencoder-based reconstruction: an autoencoder, trained on clean data, encodes the training data (compresses it) and then decodes it back into training data (decompresses it).
Membership Inference Attack [20]	ML system input/API*	Training data	Confidentiality*	1. Machine learning model's behavior (response or output) adjusts based on, if it operates with the data, it was trained on, or with unknown data. 2. Overfitted models tend to memorize training data, causing further discrepancy in output against the input data that is unknown and , which is a part of the training set. 3. Structure of the model to stronger memorization of training data, leading to leak information about its training dataset.	Attacker: black-box scenario	Shadow models are trained on datasets similar to the target model's dataset to replicate its behavior. An attack model is trained on the shadow models' outputs to distinguish between training and non-training data based on prediction behavior. The attack model is utilized if the target sample is a part of the targeted model's training dataset.	Negates the confidentiality of the targeted machine learning model. This may lead to the private data leakage and possible legal repercussions.*	The machine learning system must be resistant to membership inference attacks.*	1. Regularization - a technique that reduces the overfitted nature of a model. It generalizes the model and reduces information leakages about the training dataset. 2. Differential privacy: adds mathematically bounded noise to the training process or to the final model parameters, limiting the influence of any single data record on the model's outcomes.
Adversarial attack [28]	ML system input/API*	Input data*	Availability	There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model.	Adversary: White/Black box scenario	A submission of an adversarial input. The adversarial input can be generated with an algorithm by perturbating an original benign sample. Covered algorithms: L-BFGS method, Fast gradient sign method (FGSM), Universal adversarial perturbations (UAP), UPSET and ANGR method, C&W attack method.	Negates availability of the target machine learning system. With this type of risk, there can be collateral damage to the state of the systems that incorporated the targeted machine learning model.*	The machine learning system must be resistant to adversarial input.*	1. Implement a security evaluation mechanism: a reactive defense updates the model based on the new attacks; a proactive defense considers possible security deficiencies before deploying the model. 2. Defense mechanism during the training phase: enhance generalization capability of the machine learning model. Possible methods: Bagging (bootstrap aggregating algorithm), RSM (random subspace method) by Biggio et al., the ANTIDOTE algorithm by Rubinstein et al. 3. Defense mechanism on the prediction/test phase. Modify the machine learning to make it more resistant to adversarial samples through the following methods: adversarial training, incorporate adversarial data into the training data, this is a non-adaptive approach; conduct data compression (image specific), high compression rate may lead to the loss in classification accuracy; apply a foveation method to an image region (image specific), effectiveness against more powerful attack has not been validated; modify gradients of the input data, loss/activation function with a gradient masking method, trains the model by penalizing the input variation degree; transfer knowledge of the model to a

									new model through a defensive distillation method; DeepCloak method adds a new trained layer before the network decision layer, removing the prominent features by masking dominant weights, this method does not require model retraining. Append an external model: GAN-based method conducts GAN training of the target model; feature squeezing method (image specific) modifies properties of an image and compares the image with classification result, if there is a significant difference, the images is considered to be adversarial; universal perturbation method (image specific) implements a perturbation rectifying network (PRN) before the input layer, the network is separately trained to rectify the input images before feeding them into the target model.
Poisoning attack [28]	Machine learning training system*	Training data	Integrity, Availability	The training dataset is susceptible to unauthorized modifications.*	Adversary: White box scenario	Unauthorized malicious modification of the training dataset. Labels can be obscured within the training dataset to achieve the poisoning attack with the following specific attacks: random label flipping (RLF) attack (randomly modifies labels within the training data subset), nearest-prior label flipping (NPLF) attack (distorts labels based near the decision boundary), farthest-prior label flipping (FPLF) attack (distorts labels that are far from the decision boundary), farthest-rotation label flipping (FRLF) attack, adversarial label flipping (ALF) attacks (attempts to maximum the classification error through distorted examples).	Negates the integrity and availability of the target machine learning system. This may lead to collateral damage due to incorrect operation of the machine learning system.*	The machine learning system must be resistant to poisoning attacks.*	1. Implement a security evaluation mechanism: a reactive defense updates the model based on the new attacks; a proactive defense considers possible security deficiencies before deploying the model. 2. Defense mechanism during the training phase: data sanitization, a possible method is to attach a new sample to the existing dataset, train the model on the new set and compare the results to the previous models, if error rates significantly differ - discard the sample. This method has heavy computational cost.
Inversion attack [28]	ML system input/API*	Input data*	Confidentiality*	It is possible to extract confidential data from the trained machine learning system, trained on confidential data.	None.	None.	Negates the confidentiality of the targeted machine learning model. This may lead to legal repercussions.*	The machine learning system must be resistant to inversion attacks.*	1. Differential-privacy-based method, obscures the input by adding noise to the original data model. Possible methods: randomized aggregative privacy-preserving ordinal response (RAPPOR) method, PATE (Private Aggregation of Teacher Ensembles). The PATE method trains a teacher model on disjoint subsets of data and then a student model is trained on the teacher's output. 2. Homomorphic-encryption-based method, allows for substantial amount of important data to be securely transmitted in a cloud environment. Alternatively, a CryptoNets model was developed, which encrypts the model parameters.
Jailbreak attack [2]	ML system input/API*	Input data*	Integrity*	It is possible to bypass built-in output restrictions of a VLM by providing a malicious input data in alternative medium, as adversarial image or textual data.	Adversary: White and Black box scenarios	1. A submission of an adversarial image input alongside a textual prompt, requesting a malicious output. Adversarial image data is generated based on examples of malicious content through Projected Gradient Descent (PGD), this image data is later paired with malicious textual prompt. 2. A submission of an adversarial textual input alongside a textual prompt, requesting a malicious output. A discrete optimization algorithm from Shin et al., an improved version of the hotflip attacks can be utilized for adversarial text generation.	Negates the integrity of the targeted machine learning model. This may lead to legal repercussions.*	The machine learning system must be resistant to jailbreak attacks.*	1. Conduct adversarial training, the cost is prohibitive. In addition, the bounds on the perturbation can be much wider that generally assumed. 2. Conduct robustness certification, although the cost is prohibitive. 3. The input can be pre-processed. A possible method is DiffPure, which introduces noise into the input images and then diffuses it back into learned data manifold, thus restoring a clean image. This method cannot be applied to an offline, local model in adversary's possession. 4. Utilize common harmfulness detection APIs like Perspective API and Moderation API. The API's accuracy is limited, they may cause bias, harm and reduce helpfulness of a model. These API's are not applicable to offline models in adversary's possession.
Inference attack: membership inference [9]	Machine learning model*, ML system input/API*	Training data, Input data	Confidentiality*	It is possible to infer membership of samples within the target machine learning model.	Adversary: White and Black box scenarios	1. An attacker trains a shadow model on the part of the shadow dataset, acquired from the same distribution as the target model's training data. The model is trained to determine if the requested sample is a part of the shadow model's training dataset or not. Finally, the attacker queries the shadow model with the entire shadow dataset, to determine, if it is a part of the originally targeted model. 2. Based on the produced prediction and the shadow training data set, an attack model is trained, a binary membership classifier. (If the attacker has a part of the original training dataset, the attack model can be trained directly.) 3. The attacker then queries the target model with a target sample to determine if the sample is a part of its dataset. 4. If the attacker has access to the targeted model, then the target model's gradients can be incorporated into the training of the attack model. The attack model is then used to determine if the target sample is a part of the targeted model, based on the posteriors and predicted label for the supplied target sample.	Negates the confidentiality of the targeted machine learning model. This may lead to legal repercussions.*	The machine learning system must be resistant to inference attacks.*	1. Utilize Differential Privacy, Differentially-Private Stochastic Gradient Descend (DP-SGD). This method adds Gaussian noise to gradient during the target model's training. The method is capable of mitigating inference attacks without significantly deteriorating model's utility. 2. Knowledge Distillation (KD) method is capable of reducing membership inference attack risks. KD method transfers knowledge from the larger target model to the smaller distilled model. The distilled model can be more resource efficient that the original model.
Inference attack: model inversion [9]	ML system input/API*	Input data*	Confidentiality*	It is possible to reconstruct the training data samples from the target machine learning model.	Adversary: White box scenario	1. An attacker with white box access feeds a noise sample to the targeted model to receive its posteriors. Afterwards, through back-propagation over model's parameters the input is optimized, thus producing a representative sample of a class. 2. With a shadow dataset a generative adversarial network (GAN) can be trained. The GAN model is provided with optimized inputs, with an aim of generating samples that reach high posteriors on the target model.	Negates the confidentiality of the targeted machine learning model. This may lead to legal repercussions.*	The machine learning system must be resistant to inference attacks.*	1. Utilize Differential Privacy, Differentially-Private Stochastic Gradient Descend (DP-SGD). This method adds Gaussian noise to gradient during the target model's training. The method is capable of mitigating inference attacks without significantly deteriorating model's utility. 2. Knowledge Distillation (KD) method is capable of reducing membership inference attack risks. KD method transfers knowledge from the larger target model to the smaller distilled model. The distilled model can be more resource efficient that the original model.
Inference attack: attribute inference [9]	ML system input/API*	Input data*	Confidentiality*	It is possible to learn additional information about the training data sample from the target model's output.	Adversary: White box scenario	The attacker utilizes embeddings of the target sample from the target model to train a classifier, which will predict the sample's target attributes.	Negates the confidentiality of the targeted machine learning model. This may lead to legal repercussions.*	The machine learning system must be resistant to inference attacks.*	1. Utilize Differential Privacy, Differentially-Private Stochastic Gradient Descend (DP-SGD). This method adds Gaussian noise to gradient during the target model's training. The method is capable of mitigating inference attacks without significantly deteriorating model's utility. 2. Knowledge Distillation (KD) method is capable of reducing membership inference attack risks. KD method transfers knowledge from the larger target model to the smaller distilled model. The distilled model can be more resource efficient that the original model.

Inference attack: model stealing [9]	ML system input/API*	Input data*	Confidentiality*	It is possible to replicate target model's properties within a new model.	Adversary: Black box scenario	The attacker utilizes partial training dataset or shadow set to query the target model to receive its posteriors. The retrieved posteriors and the utilized samples are used to train the new adversary's model.	Negates the confidentiality of the targeted machine learning model. This may lead to the loss of intellectual property.*	The machine learning system must be resistant to inference attacks.*	1. Utilize Differential Privacy, Differentially-Private Stochastic Gradient Descend (DP-SGD). This method adds Gaussian noise to gradient during the target model's training. The method is capable of mitigating inference attacks without significantly deteriorating model's utility. 2. Knowledge Distillation (KD) method is capable of reducing membership inference attack risks. KD method transfers knowledge from the larger target model to the smaller distilled model. The distilled model can be more resource efficient that the original model.
Poisoning attack*: causative integrity attack [31]	Machine learning training system*	Training data	Integrity	The training dataset is susceptible to unauthorized modifications.*	Adversary: white-box scenario*	The adversary carefully inserts generated malicious samples into the training dataset. A possible method is to introduce spurious features into the training dataset to mislead the retrained classifier, and afterwards provide malicious input to the model lacking the spurious features, thus bypassing the defenses, utilized in a "Red herring attack".	Negates the integrity of the targeted machine learning model. This leads to misclassification of malicious input.*	The machine learning system must be resistant to poisoning attacks.*	1. Reject On Negative Impact (RONI), assess the empirical effect of each training sample and remove the sample that have significant negative impact on the classification accuracy. This method may have a high computational cost. 2. Utilize robust statistics, the robustness could be measured with the influence functions and breakdown point. Aim is to utilize a procedure with a high breakdown point and bounded influence function. 3. Combine multiple classifiers, which may provide different security properties to produce a composite prediction.
Poisoning attack*: causative availability attack [31]	Machine learning training system*	Training data	Availability	The training dataset is susceptible to unauthorized modifications.*	Adversary: white-box scenario*	The adversary introduces spurious features to the positive training samples, this is a "correlated outlier attack". The adversary mimics the legitimate traffic as malicious and submits it to the target classifier, the crafted malicious traffic is used for target model training, this is an "allergy attack".	Negates the availability of the targeted machine learning model. This leads to misclassification of benign, legitimate input.*	The machine learning system must be resistant to poisoning attacks.*	1. Reject On Negative Impact (RONI), assess the empirical effect of each training sample and remove the sample that have significant negative impact on the classification accuracy. This method may have a high computational cost. 2. Utilize robust statistics, the robustness could be measured with the influence functions and breakdown point. Aim is to utilize a procedure with a high breakdown point and bounded influence function. 3. Combine multiple classifiers, which may provide different security properties to produce a composite prediction.
Adversarial attack*: exploratory integrity attack [31]	ML system input/API*	Input data	Integrity	There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model.*	Adversary: white-box scenario*	The adversary crafts malicious input that is misclassified by the target model. The malicious input can be encrypted to be made statistically identical to normal input, this is a "polymorphic blending attack". The malicious input can integrate features of a benign input, this is a "good word attack".	Negates the integrity of the targeted machine learning model. This leads to misclassification of malicious input.*	The machine learning system must be resistant to adversarial attacks.*	1. Limit access to information about the training procedure and training data. Can be difficult to keep the training data secret. 2. Harden classifiers through higher order patterns, such as n-grams or randomized feature selection. 3. Create an adversary-aware classifier by adjusting the likelihood function to anticipate the attacker's changes. 4. Introduce randomness in the classification process, this may decrease the utility of the responses. 4. Limit the feedback that is provided to the attacker or provide intentionally misleading responses. This may reduce the utility of the responses.
Denial of service attack*: exploratory availability attack [31]	ML system input/API*	Input data	Availability	The target system hosting the machine learning system can be overwhelmed and made unavailable with computationally expensive requests.	Adversary: white-box scenario*	The adversary overwhelms the system by creating many computationally expensive input requests.	Negates the availability of the targeted machine learning model. This leads to potential system failures, service unavailability, increase of processing times, reduction in quality of service.*	The machine learning system must be resistant to denial of service attacks.*	Limit the feedback that is provided to the attacker or provide intentionally misleading responses. This may reduce the utility of the responses and computational cost of the request.*
Poisoning attack: error-generic, a Denial of Service attack [4]	Machine learning training system*	Training data	Availability	The training dataset is susceptible to unauthorized modifications.* The public data that is utilized for training may contain malicious samples.	Adversary: White and Black box scenario*	The attacker modifies the target dataset features and labels to maximize the loss function for targeted samples.	Negates the availability of the targeted machine learning model. This leads to misclassification of benign, legitimate input.*	The machine learning system must be resistant to poisoning attacks.*	1. Reject On Negative Impact (RONI) defense detect and discards samples within the training dataset that have negative impact on the classifier's accuracy. This techniques is computationally very expensive. The method may be susceptible to overfitting, reducing its performance, when operated on small training dataset, compared to the amount of features. 2. Detect and remove outliers, pre-filter the training dataset. 3. Combine outlier detection with optimization techniques to correlate classifier predictions with labels. This method requires prior knowledge on the fraction of the poisoned samples. 4. Utilize a small, curated, and verified subset of trusted data points to train outlier detectors for each class. This method requires curation of trusted data. 5. Relabel potentially malicious data points based on their k-Nearest Neighbors in the feature space. This method is inefficient if malicious sample are close to genuine data. 6.Setup an influence function that estimates the influence of each training sample on the model's predictions.
Poisoning attack: error-specific [4]	Machine learning training system*	Training data	Integrity, Availability	1. The training dataset is susceptible to unauthorized modifications.* 2. The target model's performance can be skewed through training on malicious samples.*	Adversary: White and Black box scenario*	The attacker mislabels targeted samples within the training dataset to minimize the loss function on the relabeled samples.	Negates the integrity and availability of the targeted machine learning model. This leads to misclassification of benign and malicious inputs.*	The machine learning system must be resistant to poisoning attacks.*	1. Reject On Negative Impact (RONI) defense detect and discards samples within the training dataset that have negative impact on the classifier's accuracy. This techniques is computationally very expensive. The method may be susceptible to overfitting, reducing its performance, when operated on small training dataset, compared to the amount of features. 2. Detect and remove outliers, pre-filter the training dataset. 3. Combine outlier detection with optimization techniques to correlate classifier predictions with labels. This method requires prior knowledge on the fraction of the poisoned samples. 4. Utilize a small, curated, and verified subset of trusted data points to train outlier detectors for each class. This method requires curation of trusted data. 5. Relabel potentially malicious data points based on their k-Nearest Neighbors in the feature space. This method is inefficient if malicious sample are close to genuine data. 6.Setup an influence function that estimates the influence of each training sample on the model's predictions.
Evasion attack/Adversarial attack*: error-generic attack [4]	ML system input/API*	Input data	Integrity*	1. The machine learning model may produce unexpected results, if the input incorporates features that are not supported by the training	Adversary: White and Black box scenario*	The attacker produces adversarial samples that are close to the incorrect classes. The adversarial samples are provided as input to the machine learning system with an aim of causing	Negates the integrity of the targeted machine learning model. This leads	The machine learning system must be resistant to	1. Adversarial re-training, the model is retrained on the training dataset that includes adversarial samples. The cost of this method may be prohibitive. 2. Gradient

				dataset's feature space. 2. The machine learning model can produce erroneous output, when the input incorporates features that exploit imperfect decision boundary produced by the learning algorithm. This may happen due to utilization of a limited training dataset or utilization of a learning algorithm with limited capacity.		misclassification. Gradient descent strategy can be used to solve the optimization problem to produce an adversarial sample. Fast Gradient Sign Method can be used as an alternative more computationally efficient method.	to misclassification of provided inputs.*	evasion (adversarial) attacks.*	masking/Defensive distillation method produces a distilled model with a smoothed out decision surface. Research has shown that this method may not effective against evasion attacks, and the produce model may be as vulnerable as the original model. 3. Dimensionality reduction, Principal Component Analysis (PCA) can be used to reduce components (feature space) of the classifier. This method may reduce performance of the algorithm in exchange for robustness.
Evasion attack/Adversarial attack*: error-specific attack [4]	ML system input/API*	Input data	Integrity*	1. The machine learning model may produce unexpected results, if the input incorporates features that are not supported by the training dataset's feature space. 2. The machine learning model can produce erroneous output, when the input incorporates features that exploit imperfect decision boundary produced by the learning algorithm. This may happen due to utilization of a limited training dataset or utilization of a learning algorithm with limited capacity.	Adversary: White and Black box scenario*	The attacker produces adversarial samples that are close to the target class. The adversarial samples are provided as input to the machine learning system to cause specific misclassification. Gradient descent strategy can be used to solve the optimization problem to produce an adversarial sample. Fast Gradient Sign Method can be used as an alternative more computationally efficient method.	Negates the integrity of the targeted machine learning model. This leads to misclassification of provided inputs.*	The machine learning system must be resistant to evasion (adversarial) attacks.*	1. Adversarial re-training, the model is retrained on the training dataset that includes adversarial samples. The cost of this method may be prohibitive. 2. Gradient masking/Defensive distillation method produces a distilled model with a smoothed out decision surface. Research has shown that this method may not effective against evasion attacks, and the produce model may be as vulnerable as the original model. 3. Dimensionality reduction, Principal Component Analysis (PCA) can be used to reduce components (feature space) of the classifier. This method may reduce performance of the algorithm in exchange for robustness.
Adversarial attack* [14]	ML system input/API*	Input data	Integrity*	There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model.*	None.	Addition of perturbations to the input data to be processed by the target AI system.	Negates the integrity of the targeted machine learning model.*	The machine learning system must be resistant to adversarial attacks.*	None.
Model extraction* [14]	ML system input/API*	Input data*	Confidentiality*	It is possible to replicate target model's properties within a new model based on the target machine learning model's output.*	None.	None.	Negates the confidentiality of the targeted machine learning model. This may lead to the loss of intellectual property.*	The machine learning system must be resistant to model extraction attacks.*	None.
Cyber-physical attack* [14]	Processing hardware running the ML model*	Input data	Availability	A machine learning system is susceptible to vulnerabilities on software and hardware, on which it depends.	None.	Exploit vulnerability in the software or hardware is exploited, on which the target machine learning system depends.	Negates the availability of the targeted machine learning model.*	The machine learning system must be resistant to malicious cyber-physical attacks.*	None.
None. [14]	None.	None.	None.	The machine learning system does transparently and correctly explain the produced decisions and taken actions.	None.	Lack of machine learning explainability can be used to hide malicious intervention into machine learning system's operations.	None.	The machine learning system's actions and decisions must be transparently and correctly explained.*	None.
Jailbreak attack* (Adversarial attack) [23]	ML system input/API*	Input data	Integrity*	1. There exists inherent ambiguity between the decision boundaries of a machine learning model and true decision boundaries.* 2. Lack of or insufficiency in defensive measures.* 3. Vulnerability of deep neural networks to small, almost imperceptible perturbations to benign examples. 4. Transferability of adversarial examples from surrogate to target models.	Adversary: Black box scenario	1. A surrogate LLM is fine-tuned. 2. Optimize adversarial distribution using Gumbel-Softmax. 3. Apply constraint model (CLM) for perplexity/semantic regularization. 4. Use geometric loss to balance objectives. 5. Sample adversarial examples with semantic filtering. 6. Submit the generated adversarial data as an input to the target LLM system.	Negates the integrity of the targeted machine learning model. This leads to reduction in model's accuracy.*	The machine learning system's actions and decisions must be resistant to jailbreak attacks.*	None.
Adversarial attack: Fuzzing attack [70]	ML system input/API*	Input data	Integrity*	There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model. There exists inherent ambiguity between the decision boundaries of a machine learning model and true decision boundaries.*	Adversary: Black box scenario	A sample is iteratively modified with a guidance image to produce an adversarial sample of the target class. Each step takes a seed, mutates the image, evaluates the result. Final adversarial sample is submitted as an input to the target machine learning system.	Negates the integrity of the targeted machine learning model. This leads to reduction in model's accuracy.*	The machine learning system's actions and decisions must be resistant to adversarial attacks.*	1. Query number restriction. 2. Leverage the proposed fuzzing attack framework to improve the robustness of the defense mechanisms against bulk-generated adversarial examples. 3. Adversarial training on examples generated by such attacks to strengthen model robustness. 4. Certified defense, achieving a constant model prediction within a specific bound; a base classifier is trained with noise to average model's output over noisy samples.
Poisoning attack (random label flipping) [29]	Machine learning training system*	Training data	Integrity*, Availability*	1. The training dataset is susceptible to unauthorized modifications.* 2. The target model's performance can be skewed through training on malicious samples.* 3. Lack of the training data integrity checks.* 4. Lack of sanitization of publicly acquired training data.*	Adversary: White box scenario	Label values of the half of the target training dataset is iteratively flipped.	Negates the integrity and availability of the targeted machine learning model. This leads to reduction in model's accuracy.*	The machine learning system's actions and decisions must be resistant to poisoning attacks.*	None.
Poisoning attack (distance-based table flipping) [29]	Machine learning training system*	Training data	Integrity*, Availability*	1. The training dataset is susceptible to unauthorized modifications.* 2. The target model's performance can be skewed through training on malicious samples.* 3. Lack of the training data integrity checks.* 4. Lack of sanitization of publicly acquired training data.*	Adversary: White box scenario	Label values of samples from the data set a flipped based on the distance from the classifier's decision hyperplane.	Negates the integrity and availability of the targeted machine learning model. This leads to reduction in model's accuracy.*	The machine learning system's actions and decisions must be resistant to poisoning attacks.*	None.
Adversarial attack* (Evasion attack) [8]	ML system input/API*	Input data*	Integrity*	There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model. There exists inherent ambiguity between the decision boundaries of a machine learning model and true decision boundaries.*	Adversary: White and Grey box scenarios	Adversarial samples are iteratively created by modifying it through a gradient-based algorithm to reduce the classifier's discriminant function value and push the sample across the decision boundary. The produced adversarial samples are submitted to the machine learning algorithm for processing.	Negates the integrity of the targeted machine learning model. This leads to reduction in model's accuracy and may result in malicious output.*	The machine learning system's actions and decisions must be resistant to adversarial attacks.*	1. Utilization of regularization terms that promote enclosure of the legitimate class. 2. Adversarial training., training including malicious samples.
Poisoning attack [8]	Machine learning training system*	Training data	Availability	1. The training dataset is susceptible to unauthorized modifications.* 2. The target model's performance can be skewed through training on malicious samples.* 3. Lack of the training data integrity checks.* 4. Lack of sanitization of publicly acquired training data.*	Adversary: White and Grey box scenarios	An attacker chooses an initial guess for each poisoned sample. The chosen guess is used to retrain the model. The difference in model's performance is used to update the guessed poisoned sample through computation of (sub)gradient-ascent algorithm. The final set of produced poisoned samples is injected into the target model' data set.	Negates the availability of the targeted machine learning model. This leads to reduction in model's accuracy.*	The machine learning system's actions and decisions must be resistant to poisoning attacks.*	None.

Poisoning attack [8]	Machine learning training system*	Training data	Availability	1. The training dataset is susceptible to unauthorized modifications.* 2. The target model's performance can be skewed through training on malicious samples.* 3. Lack of the training data integrity checks.* 4. Lack of sanitization of publicly acquired training data.*	Adversary: White box scenario	Labels of the samples within the training set of the targeted ML model are mixed. The target ML model trains on the modified training data set.	Negates the availability of the targeted machine learning model. This leads to reduction in model's accuracy.*	The machine learning system's actions and decisions must be resistant to poisoning attacks.*	None.
Adversarial attack* (Evasion attack) [8]	ML system input/API*	Input data*	Integrity*	1. There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model. There exists inherent ambiguity between the decision boundaries of a machine learning model and true decision boundaries.* 2. Better fitting (overfitting) make models more vulnerable to the deviations introduced by the adversaries. There is an inverse relationship between single model fitting accuracy and robustness to adversarial evasion. 3. Machine learning model depends on the features, which can be mimicked and manipulated by the adversary.	Adversary: White box scenario	Malicious machine learning models are trained on the features, which are analyzed by the targeted ML model and target ML model's performance against malicious model's behavior. Malicious machine learning models are utilized to produce adversarial samples. The produced adversarial samples are submitted to the target model for analysis.	Negates the integrity of the targeted machine learning model. This leads to reduction in model's accuracy.*	The machine learning system's actions and decisions must be resistant to adversarial attacks.*	None.
Adversarial attack* (Evasion attack) [8]	ML system input/API*	Input data*	Integrity*	1. Reliance of the model on non-predictive (not valuable) features. 2. There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model. There exists inherent ambiguity between the decision boundaries of a machine learning model and true decision boundaries.*	Adversary: Black box scenario	Influential features are algorithmically identified through feature attribution methods. The non-essential features selected and are modified via gradient-guided optimization by computing the gradient of target model's classification function. The malicious input is iteratively modified until it gets misclassified by the target model. The produced malicious input data is submitted to the target model for analysis.	Negates the integrity of the targeted machine learning model. This leads to reduction in model's accuracy.*	The machine learning system's actions and decisions must be resistant to adversarial attacks.*	1. Explainable AI design, utilization of methods like LIME and LASSO to learn the decision boundary around specific input points. This could be helpful in design of more robust ML models.
Adversarial attack* (Evasion attack) [8]	ML system input/API*	Input data*	Integrity*	There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model. There exists inherent ambiguity between the decision boundaries of a machine learning model and true decision boundaries.*	Adversary: Black, Grey and White box scenarios	An adversarial sample is produced, accounting for the weaknesses in target ML classification capabilities, to evade its proper classification. The produced adversarial sample is submitted to the target ML model for analysis.	Negates the integrity of the targeted machine learning model. This leads to reduction in model's accuracy.*	The machine learning system's actions and decisions must be resistant to adversarial attacks.*	None.
Poisoning attack [8]	Machine learning training system*	Training data	Integrity*	1. The target model's performance can be skewed through training on malicious samples.* 2. Lack of the training data integrity checks.* 3. Lack of sanitization of publicly acquired training data.*	Adversary: Black, Grey and White box scenarios	Malicious samples are derived and are injected into the target model's training data set. The target ML model is trained on the modified training dataset.	Negates the integrity of the targeted machine learning model. This leads to reduction in model's accuracy.*	The machine learning system's actions and decisions must be resistant to poisoning attacks.*	None.
Adversarial attack [71]	ML system input/API*	Input data*	Integrity*	There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model. There exists inherent ambiguity between the decision boundaries of a machine learning model and true decision boundaries.*	Adversary: Black box scenario*	With Jaccard similarity a malicious samples is iterated upon to arrive at its modified version, closes to a benign sample, which gets misclassified by the target ML model. The final produced malicious samples is submitted to the target ML model for analysis.	Negates the integrity of the targeted machine learning model. This leads to reduction in model's accuracy.*	The machine learning system's actions and decisions must be resistant to adversarial attacks.*	1. Robust feature selection method - a feature selection method, considering the feature importance for classification, the costs of manipulating of each feature, probabilistically selecting features inversely proportional to their attack vulnerability. 2. Ensemble learning - combination of multiple classifiers trained on different feature subsets; the classifiers are designed so that all the features are integrated and that classifiers differentiate from each other.
Adversarial attack [71]	ML system input/API*	Input data*	Integrity*	There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model. There exists inherent ambiguity between the decision boundaries of a machine learning model and true decision boundaries.*	Adversary: Black box scenario*	Features of the malicious sample are randomly modified. The produced malicious sample is submitted to the target ML model for analysis.	Negates the integrity of the targeted machine learning model. This leads to reduction in model's accuracy.*	The machine learning system's actions and decisions must be resistant to adversarial attacks.*	1. Robust feature selection method - a feature selection method, considering the feature importance for classification, the costs of manipulating of each feature, probabilistically selecting features inversely proportional to their attack vulnerability. 2. Ensemble learning - combination of multiple classifiers trained on different feature subsets; the classifiers are designed so that all the features are integrated and that classifiers differentiate from each other.
Adversarial attack [71]	ML system input/API*	Input data*	Integrity*	There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model. There exists inherent ambiguity between the decision boundaries of a machine learning model and true decision boundaries.*	Adversary: Black box scenario*	A feature within the malicious sample is iteratively selected and greedily updated with an aim of increasing classification errors. Features are ranked through information gain. The features are further bi-directionally selected. The feature are added and eliminated from the target malicious sample. The produced malicious sample is submitted to the target ML model for analysis.	Negates the integrity of the targeted machine learning model. This leads to reduction in model's accuracy.*	The machine learning system's actions and decisions must be resistant to adversarial attacks.*	1. Robust feature selection method - a feature selection method, considering the feature importance for classification, the costs of manipulating of each feature, probabilistically selecting features inversely proportional to their attack vulnerability. 2. Ensemble learning - combination of multiple classifiers trained on different feature subsets; the classifiers are designed so that all the features are integrated and that classifiers differentiate from each other.
Model manipulation (Evasion attack) [27]	Machine learning model	Model's parameters	Integrity*	1. Lack of model integrity verification.* 2. Unauthorized access to the model.	Adversary: White box scenarios	A loss function is optimized combining cross-entropy loss to misclassify target samples and weight regularization to minimize parameter deviations from the original model, ensuring that selected malicious samples are classified as benign. The final modifications are applied to the target ML model, and malicious samples are submitted.	Negates the integrity of the targeted machine learning model. This leads to reduction in model's accuracy.*	The machine learning system's actions and decisions must be resistant to model manipulation attacks.*	None.
Model manipulation (Evasion attack) [27]	Machine learning model	Model's parameters	Integrity*	1. Lack of model integrity verification.* 2. Unauthorized access to the model.	Adversary: White box scenarios	Using backpropagation, the final layers are optimized to force the target model into assigning incorrect labels to the chosen malicious samples while minimizing changes to overall model accuracy. Only the fully connected layers are modified through gradient descent with an added constraint, while initial convolutional layers are frozen, preserving the model's ability to generalize while misclassifying specific target malicious samples. The final modifications are applied to the target ML model, and malicious samples are submitted.	Negates the integrity of the targeted machine learning model. This leads to reduction in model's accuracy.*	The machine learning system's actions and decisions must be resistant to model manipulation attacks.*	None.
Jailbreak attack (Prompt Injection) [10]	ML system input/API*	Input data	Integrity*, Confidentiality	1. There exists inherent ambiguity between the decision boundaries of a machine learning model and true decision boundaries.* 2. Lack of or insufficiency in defensive measures.* 3. Increased vulnerability in multi-modal setup. 4.	Adversary: Black box scenario*	Input of a malicious prompt, bypassing present safeguards.	Negates the integrity of the targeted machine learning model. This leads to the production of the unintended output by the model. In addition, a successful attack may lead to: disclosure of sensitive information;	The machine learning system's actions and decisions must be resistant to jailbreak attacks.*	1. Model behavior constraints: definition of system prompts, restricting model's capabilities, role, accepted output, actions and topics. 2. Deterministic validation of the output. 3. Filter input and output: assess the context, consider semantic filtering; sanitize the input data. 4. Restrict LLM application's privileges. 5. Human-in-the-loop control and validation of

				Model's API misconfiguration. 5. Stochastic nature of the machine learning model.			leak of AI system architecture or system prompt information; access to the unauthorized actions; manipulation of the decision-making process.		privileged operations caused by the LLM application. 6. Separation of the external input data and limitation of its influence. 7.Conduct penetration testing. 8. Restrict model's access to sensitive and external data sources. 9. Federated learning, reduces the need for centralized data collection and by extension reduces data exposure risk. 9. Differential privacy - addition of noise to the output, reduces the data exposure risk. 10. Education of users on the risks of sensitive data input. 11. Define clear and transparent policy about data retention, usage, removal. 12. Conceal system prompt and configuration. 13. Proper and secure configuration of the model's input API, to restrict feedback on error and configuration details. 14. Homomorphic encryption - to ensure secure data analysis and training process.
Supply chain attack [10]	Enabling software components*	Machine learning model development process*	Integrity, Availability*	1. Deployment platform vulnerabilities. 2. Traditional software vulnerabilities: code and dependencies flaws. 3. Corrupted, vulnerable 3rd party pre-trained models. 4. Usage of unmaintained software, models. 5. Vulnerable LoRa adapters, a malicious adjustment of the pre-trained model. 6. Development platform vulnerabilities.	Adversary: Black box scenario*	Infection or impersonation of a software components, public machine learning models, development environment or data resource, on which the target machine learning model system depends on.	Negates the integrity and availability of the targeted machine learning model. This attack may lead to the loss of integrity of training data, deployment platform, biased output, security breach.	The machine learning system's actions and decisions must be resistant to supply chain attacks.*	1. Vet data sources, suppliers, terms and conditions, privacy policies. 2. Regular review and audit suppliers' security and terms. 3. Vulnerability scanning. 4. Vulnerable software patching or a virtual patching . 5. Remove unused dependencies and unnecessary features. 6. Monitor dependencies for their state, versions and vulnerabilities. 7. Source components and components from official sources. 8. Conduct red teaming, penetration testing, integrity checking against 3rd party models. 9. Maintain an inventory of components, a Software Bill of Materials (SBOM). 10. Utilize code signing for externally supplied code. 11. Monitor and audit collaborative and development environments. Example: HuggingFace SF_Convertbot Scanner. 12. Utilize integrity checks and vendor attestation APIs against apps and models.
Poisoning attack [10]	Machine learning training system*	Training data	Integrity, Availability*	1. Gathering of public data without validation and sanitization.* 2. Unauthorized access to the training data. 3. Usage of unverified and unsanitized training data.	Adversary: White and Black box scenario*	1. Poisoning of the general training data. 2. Poisoning the fine-tuning process. 3. Embedding process poisoning (conversion of text into numerical vectors). 4. Poisoning of public sources through temporary content modification or domain re-acquisition.	Negates the integrity and availability of the targeted machine learning model.* This may lead to the model's reduced performance, undesirable output, impaired capability. This in turn may lead to the exploitation, corruption, underperformance of the downstream systems.	The machine learning system's actions and decisions must be resistant to poisoning attacks.*	1. Track training data origins and transformations, verify data legitimacy. 2. Vet data vendors. 3. Monitor and validate model's output after conducting training. 4. Filter the training data for adversarial samples. 5. Limit model's exposure to the unverified training data. 6.Narrow down the scope and purpose of the models and the utilized training datasets. 7. Version control and track changes within the training datasets. 8. Conduct red teaming and penetration testing to ensure the desired output of the model. 9. Federated learning - reduce the influence of training data perturbations. 10. Implementation of Retrieval-Augmented Generation (RAG): retrieval of additional information and its integration in form additional context.
Embedding Inversion Attacks [10]	Supporting IT infrastructure*	Input data embeddings	Confidentiality	1. Embedding vectors retains enough information to enable reconstruction of significant parts of the original text. 2. The leakage or unauthorized access to the embeddings.	Adversary: White and Black box scenario	Inversion of embeddings, leading to recovery of source information. Utilize gradient-based (white-box) or learning-based (black-box) methods to invert the target embeddings. The produced mappings will reveal partially reveal the original input to the model.	Negates the confidentiality of previously provided input to the machine learning system, by extension model's confidentiality is compromised in addition. This may lead to legal repercussions.	The machine learning system's actions and decisions must be resistant to embedding inversion attacks.*	1. Implement permission and access control to the embedding store. 2. Monitor and log the data retrieval activities. 3. Audit and validate integrity of the data stores. 4. Operate with data retrieved only from the trusted sources. 5. Monitor an devaluate the influence of RAG on the model's performance.
Denial of service attack [10]	ML system input/API*	Input data	Availability	1. There is no restrictions on the amount and frequency of inference requests. 2. Operation of the machine learning has high computational demands.	Adversary: Black box scenario*	Continuously conduct a high frequency of computationally demanding inference requests.	Negates the availability of the targeted machine learning model. This leads to potential system failures, service unavailability, increase of processing times, reduction in quality of service.*	The machine learning system's actions and decisions must be resistant to denial of service attacks.*	1. Input validation, validate the size. 2. Dynamically allocate necessary resources for model's performance. 3. Rate limit the amount of requests from a particular user. 4. Throttle and timeout resource-intensive on-going model operations. 5. Monitor the resource usage for possible anomalies. 6. Design graceful degradation, achieve partial functionality under heavy load. 7.Regulate and load-balance queued actions.
Model extraction attack [10]	ML system input/API*	Input data*	Confidentiality*	1. There is no restrictions on the amount and frequency of inference requests. 2. Operation of the machine learning has high computational demands and a high cost.	Adversary: Black box scenario*	Querying of model's input API with malicious input samples for collection of data, sufficient for model replication.	Negates the confidentiality of the targeted machine learning model.* This may lead to the intellectual property theft.	The machine learning system's actions and decisions must be resistant to inversion attacks.*	1. Rate limit the amount of requests from a particular user. 2. Watermark the model's outputs to detect their unauthorized usage. 3. Adversarial training, training the model for detection and resistance against the extraction attempts.
Denial of wallet [10]	ML system input/API*	Input data*	Availability	High cost of model services' operation.	Adversary: Black box scenario*	Initiation of a high volume of requests, operations.	Negates the availability of the targeted machine learning model.* This may lead to the unsustainable financial burdens.	The machine learning system's actions and decisions must be resistant to denial of wallet attacks.*	1. Input validation, validate the size. 2. Dynamically allocate necessary resources for model's performance. 3. Rate limit the amount of requests from a particular user. 4. Throttle and timeout resource-intensive on-going model operations. 5. Monitor the resource usage for possible anomalies. 6. Design graceful degradation, achieve partial functionality under heavy load. 7.Regulate and load-balance queued actions.

III. Combination of the SLR Extracted Data

The aggregated form of the extracted data from the systematic literature review can be found in the form of an Excel file, on the “aggregated_threats” sheet within the following repository: https://github.com/CybRookie/llm_threat_modeling_research_artifacts. The content from the sheet is presented below in an alternative, textual form.

i. [IT.T.9], [TD.T.5] Poisoning Attack

Attack definition: Poisoning attacks involve an adversary compromising a machine learning (ML) model by manipulating the training data [4], [27], [28]. The attacker injects malicious data into the training dataset or alters the original training data [29]. The high-level goal is to maximize the generalization error in the classification process or reduce the system’s performance [4], [8]. These attacks occur during the training process, aiming to shift the decision boundaries of classifiers [29].

System Asset: machine learning training system, ML system input/API.

Business Asset: input data, training data.

Security Criteria: integrity, availability of input data and training data.

Vulnerability:

1. The training dataset is susceptible to unauthorized modifications.
2. The target model's performance can be skewed through training on malicious samples.
3. Training data directly influences the performance of the machine learning mode.
4. *Gathering of public data without validation and sanitization* [26].
5. Lack of training data integrity checks.
6. *Unauthorized access to the training data* [10].
7. Lack of sanitization of publicly acquired training data.
8. *Usage of unverified and unsanitized training data* [10].

Threat Agent: white-box scenario.

Attack Method:

1. The adversary introduces spurious features to the positive training samples, this is a "correlated outlier attack". The adversary mimics the legitimate traffic as malicious and submits it to the target classifier, the crafted malicious traffic is used for target model training, this is an "allergy attack" [31].
2. Addition of the malicious training samples into the dataset and target model retraining on the poisoned dataset. The malicious samples are produced by random generation of noise and its addition to the original sample [46].
3. Unauthorized malicious modification of the training dataset. Labels can be obscured within the training dataset to achieve the poisoning attack with the following specific attacks: random label flipping (RLF) attack (randomly modifies labels within the training data subset), nearest-prior label flipping (NPLF) attack (distorts labels based near the decision boundary), farthest-prior label flipping (FPLF) attack (distorts labels that are far from the decision boundary), farthest-rotation label flipping (FRLF) attack, adversarial label flipping (ALF) attacks (attempts to maximum the classification error through distorted examples) [28].
4. The attacker mislabels targeted samples within the training dataset to minimize the loss function on the relabeled samples [4].

5. Malicious samples are derived and are injected into the target model's training data set. The target ML model is trained on the modified training dataset [8].
6. Poisoning of the general (public data) training data[10].
7. Poisoning the fine-tuning process [10].
8. Embedding process poisoning (conversion of text into numerical vectors) [10].
9. Poisoning of public sources through temporary content modification or domain re-acquisition [10].
10. The adversary carefully generated malicious samples into the training dataset. A possible method is to introduce spurious features in the training dataset to mislead the classifier and provide malicious input to the model lacking the spurious features, thus bypassing the defenses, utilized in a "Red herring attack" [31].
11. Label values of the half of the target training dataset is iteratively flipped [29].
12. Label values of samples from the data set flipped based on the distance from the classifier's decision hyperplane [29].
13. An attacker chooses an initial guess for each poisoned sample. The chosen guess is used to retrain the model. The difference in model's performance is used to update the guessed poisoned sample through computation of (sub)gradient-ascent algorithm. The final set of produced poisoned samples is injected into the target model' data set [8].
14. Labels of the samples within the training set of the targeted ML model are mixed. The target ML model trains on the modified training data set [8].

Impact and Harm: Negates the integrity and availability of the targeted machine learning model. This leads to misclassification of malicious input.

Security Requirements: The machine learning system must be resistant to poisoning attacks.

Security Control:

1. Quantum Neural Network (QNN) is more resistant to the attack than the conventional neural networks [46].
2. Utilize robust statistics, the robustness could be measured with the influence functions and breakdown point. Aim is to utilize a procedure with a high breakdown point and bounded influence function [31].
3. Implement a security evaluation mechanism: a reactive defense updates the model based on the new attacks; a proactive defense considers possible security deficiencies before deploying the model [28].
4. Defense mechanism during the training phase: data sanitization, a possible method is to attach a new sample to the existing dataset, train the model on the new set and compare the results to the previous models, if error rates significantly differ - discard the sample. This method has heavy computational costs [28].
5. Combine multiple classifiers, which may provide different security properties to produce a composite prediction [31].
6. Reject On Negative Impact (RONI) defense detect and discards samples within the training dataset that have a negative impact on the classifier's accuracy. This technique is computationally very expensive. The method may be susceptible to overfitting, reducing its performance, when operated on a small training dataset, compared to the amount of features [4].
7. Combine outlier detection with optimization techniques to correlate classifier predictions with labels. This method requires prior knowledge on the fraction of the poisoned samples [4].
8. Utilize a small, curated, and verified subset of trusted data points to train outlier detectors for each class. This method requires curation of trusted data [4].

9. Vet data sources, suppliers, terms and conditions, privacy policies [10].
10. Regular review and audit suppliers' security and terms [10].
11. Vulnerability scanning [10].
12. Vulnerable software patching or a virtual patching [10].
13. Remove unused dependencies and unnecessary features [10].
14. Monitor dependencies for their state, versions and vulnerabilities [10].
15. Source components and components from official sources [10].
16. Conduct red teaming, penetration testing, integrity checking against 3rd party models [10].
17. Maintain an inventory of components, a Software Bill of Materials (SBOM) [10].
18. Utilize code signing for externally supplied code [10].
19. Monitor and audit collaborative and development environments [10].
20. Utilize integrity checks and vendor attestation APIs against apps and models [10].
21. Relabel potentially malicious data points based on their k-Nearest Neighbors in the feature space. This method is inefficient if malicious sample are close to genuine data [4].
22. Setup an influence function that estimates the influence of each training sample on the model's predictions [4].
23. Detect and remove outliers, pre-filter the training dataset [4].
24. Reject On Negative Impact (RONI), assess the empirical effect of each training sample and remove the sample that have significant negative impact on the classification accuracy. This method may have a high computational cost [31].

a. [TD.T.6] Poisoning Attack: Backdoor Attack

Attack definition: A backdoor attack is a specific type of poisoning attack where adversaries modify the labels of training samples and inject these mislabeled data with backdoor triggers into the training dataset [21]. The goal is to force the trained model to assign a desired target label to new samples containing the trigger [21].

System Asset: machine learning training system, ML system input/API.

Business Asset: training data.

Security Criteria: integrity of training data.

Vulnerability:

1. The training dataset is susceptible to unauthorized modifications.
2. The model 's classification boundary is prone to compromise, if trained on malicious data samples.
3. Training data directly influences the performance of the machine learning mode.
4. The model is incapable of distinguishing original and poisoned samples, when subtle frequency-domain modifications are introduced.
5. The target model's performance can be skewed through training on malicious samples.

Threat Agent: white-box and gray-box scenarios.

Attack Method:

1. Frequency space signals of the trigger and benign images are obtained with the Fast Fourier Transform (FFT). Trigger data is injected into the amplitude spectrum of the original, benign image from the trigger image's amplitude spectrum [67].

2. Procedural noise is crafted using algorithms, such as Perlin noise, Gabor noise, Worley noise. A clean model is trained on original images to get the attention images. The attention images are fused with noise and then added to the original image. The target model is trained on the poisoned data, turning into a backdoored model [30].
3. A random trigger is sampled from uniform distribution. The random trigger is added at random locations within original samples with a target label [44].
4. Backdoor Generating Network (c-BaN): a modified BaN for generation of label-specific triggers. The c-BaN model uses both the target label and a noise vector as inputs, creating triggers that can correspond to any target label, and that can be positioned at any location within the input space. This enables shared locations for triggers among different target labels, enhancing the stealthiness of the backdoor. The c-BaN is used to generate a backdoor trigger that is added to a target sample, which is to be submitted to the target model [44].
5. A Backdoor Generating Network (BaN) is created: BaN is a generative model (GAN based) that algorithmically creates backdoor triggers instead of relying on fixed patterns or random sampling. BaN is jointly trained with the target model (acts as a discriminator), which optimizes the trigger patterns. The BaN produces a trigger, it is provided as an input to the target model. Received loss from the backdoored sample is used with losses from benign samples to retrain the BaN. When BaN is ready, it produces a trigger that is added to the target sample in particular location [44].

Impact and Harm: Negates the integrity of the targeted machine learning model. This leads to misclassification of malicious input.

Security Requirements: The machine learning system must be resistant to backdoor attacks.

Security Control:

1. Activate clustering: a detection method splitting feature representation into a poisoned and clean cluster. The last hidden layer can detect differences among cluster's high-level features. Perlin and Gabor noise avoid detection, while Worley noise gets detected [30].
2. Autoencoder-based reconstruction: an autoencoder, trained on clean data, encodes the training data (compresses it) and then decodes it back into training data (decompresses it) [44].

b. [TD.T.7] Poisoning Attack: a Denial of Service Attack

Attack definition: A poisoning DoS attack is a type of adversarial attack where an attacker manipulates the training data of a machine learning (ML) model with the explicit goal of disrupting the system's availability [4], [21], [31]. This involves injecting malicious data or altering existing data into the training set to degrade the model's performance and cause a denial of service [4], [6], [21], [27], [29].

System Asset: machine learning training system.

Business Asset: training data.

Security Criteria: availability of training data.

Vulnerability:

1. The training dataset is susceptible to unauthorized modifications.
2. The public data that is utilized for training may contain malicious samples.

Threat Agent: white-box and black-box scenarios.

Attack Method: The attacker modifies the target dataset features and labels to maximize the loss function for targeted samples [4].

Impact and Harm: Negates the availability of the targeted machine learning model. This leads to misclassification of malicious input.

Security Requirements: The machine learning system must be resistant to poisoning attacks.

Security Control:

1. Reject On Negative Impact (RONI) defense detect and discards samples within the training dataset that have a negative impact on the classifier's accuracy. This technique is computationally very expensive. The method may be susceptible to overfitting, reducing its performance, when operated on small training dataset, compared to the amount of features [4].
2. Combine outlier detection with optimization techniques to correlate classifier predictions with labels. This method requires prior knowledge on the fraction of the poisoned samples [4].
3. Utilize a small, curated, and verified subset of trusted data points to train outlier detectors for each class. This method requires curation of trusted data [4].
4. Relabel potentially malicious data points based on their k-Nearest Neighbors in the feature space. This method is inefficient if malicious samples are close to genuine data [4].
5. Setup an influence function that estimates the influence of each training sample on the model's predictions [4].
6. Detect and remove outliers, pre-filter the training dataset [4].

ii. [IT.T.1] Adversarial Attack

Attack definition: Adversarial attacks are malicious attempts to fool or subvert machine learning (ML) models by exploiting weaknesses in their algorithms or training data [6], [13], [32]. In this paper, these attacks are considered to occur the testing/inference phase (evasion attacks) [4], [21]. The goal of these attacks is to undermine the performance, reliability, or security of ML systems through malicious input data [6], [21].

System Asset: ML system input/API.

Business Asset: input data.

Security Criteria: integrity of input data.

Vulnerability:

1. There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model.
2. There exists inherent ambiguity between the decision boundaries of a machine learning model and true decision boundaries.
3. A system flaw, that can be accessed and exploited externally.
4. There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model. There exists inherent ambiguity between the decision boundaries of a machine learning model and true decision boundaries.

Threat Agent: white-box and black-box scenarios.

Attack Method:

1. Initially the target model is reverse engineered, a shadow model is created. Adversarial samples are generated with the use of gradient-based methods (e.g., FGSM-like approach) and created shadow models. The adversarial samples are submitted to the target model to cause malfunction in its classification [32].
2. A submission of an adversarial sample. A set of suitable perturbations is determined by solving a constrained optimization problem. Afterwards, the set of perturbation is reduced based on imposed spatial and physical constraints. The suitable perturbation is applied to the target setting, resulting in an adversarial sample [7].
3. The target model is queried limited number of times. The retrieved outputs are utilized to train a substitute model (feed-forward neural network (FFNN)), while augmenting data with a custom augmentation algorithm and further retraining the model. Adversarial sample re-crafted from the trained substitute model. Adversarial sample generation accounts for the cost gradients of the substitute model, the degree of perturbations is limited. The produced adversarial samples are submitted to the target model [45].
4. An adversarial sample can be produced through one of the four algorithms, two gradient-based - Carlini & Wagner, Fast Gradient Method, and two gradient-free - Decision-based attack, Simulated Annealing [68].
5. A submission of an adversarial input. The adversarial input can be generated with an algorithm by perturbating an original benign sample. Covered algorithms: L-BFGS method, Fast gradient sign method (FGSM), Universal adversarial perturbations (UAP), UPSET and ANGR method, C&W attack method [28].
6. The adversary crafts malicious input that is misclassified by the target model. The malicious input can be encrypted to be made statistically identical to normal input, this is a "polymorphic blending attack". The malicious input can integrate features of a benign input, this is a "good word attack" [31].
7. With Jaccard similarity a malicious samples is iterated upon to arrive at its modified version, closes to a benign sample, which gets misclassified by the target ML model. The final produced malicious samples is submitted to the target ML model for analysis [71].
8. Features of the malicious sample are randomly modified. The produced malicious sample is submitted to the target ML model for analysis [71].
9. A sample is iteratively modified with a guidance image to produce an adversarial sample of the target class. Each step takes a seed, mutates the image, evaluates the result. Final adversarial sample is submitted as an input to the target machine learning system [70].
10. A feature within the malicious sample is iteratively selected and greedily updated with an aim of increasing classification errors. Features are ranked through information gain. The features are further bi-directionally selected. The features are added and eliminated from the target malicious sample. The produced malicious sample is submitted to the target ML model for analysis [71].
11. Addition of perturbations to the input data to be processed by the target AI system [14].
12. Requests are made to the target model and the model's responses are collected. Collected samples can be used to create perturbations for adversarial attacks with universal adversarial perturbation (UAP) method. Alternatively, a surrogate model is trained on responses, it is used for estimating target response; Basic Iterative Method

(BIM) is used to produce perturbations. Perturbations are used for creation adversarial samples [5].

13. With the knowledge of the target model, adversarial examples can be directly produced with Basic Iterative Method (BIM) [5].

Impact and Harm: Negates the integrity of the targeted machine learning model. This leads to misclassification of benign and malicious inputs.

Security Requirements: The machine learning system must be resistant to adversarial attacks.

Security Control:

1. Adversarial training: using adversarial examples in training [32].
2. Data randomization, modification of the input data [32].
3. Data compression of the input [32].
4. Addition of a masking layer to control dominant weights, thus reducing model's sensitivity [32].
5. Regularization of model based off of models; outputs and inputs during training [32].
6. Feature squeezing, utilize multiple models to determine if the sample is adversarial [32].
7. Utilize generative adversarial networks to train a model [32].
8. Adversarial training: inclusion of adversarial samples into the training dataset, thus training to minimize the loss from the inclusion of adversarial samples [7].
9. Rectification, additional of new "pre-input" layer, acting as a perturbations detector [32].
10. Ensemble Adversarial Learning provided robustness against Fast Gradient Method. Carlini attack remains successful [68].
11. Implement a security evaluation mechanism: a reactive defense updates the model based on the new attacks; a proactive defense considers possible security deficiencies before deploying the model [28].
12. Defense mechanism during the training phase: enhance generalization capability of the machine learning model. Possible methods: Bagging (bootstrap aggregating algorithm), RSM (random subspace method) by Biggio et al., the ANTIDOTE algorithm by Rubinstein et al [28].
13. Defense mechanism on the prediction/test phase. Modify the machine learning to make it more resistant to adversarial samples through the following methods: adversarial training, incorporate adversarial data into the training data, this is a non-adaptive approach; conduct data compression (image specific), high compression rate may lead to the loss in classification accuracy; apply a foveation method to an image region (image specific), effectiveness against more powerful attack has not been validated; modify gradients of the input data, loss/activation function with a gradient masking method, trains the model by penalizing the input variation degree; transfer knowledge of the model to a new model through a defensive distillation method; DeepCloak method adds a new trained layer before the network decision layer, removing the prominent features by masking dominant weights, this method does not require model retraining. Append an external model: GAN-based method conducts GAN training of the target model; feature squeezing method (image specific) modifies properties of an image and compares the image with classification result, if there is a significant difference, the images is considered to be adversarial; universal perturbation method (image specific) implants a perturbation rectifying network (PRN) before the input layer, the network is separately trained to rectify the input images before feeding them into the target model [28].

14. Limit access to information about the training procedure and training data. Can be difficult to keep the training data secret [31].
15. Harden classifiers through higher order patterns, such as n-grams or randomized feature selection [31].
16. Create an adversary-aware classifier by adjusting the likelihood function to anticipate the attacker's changes [31].
17. Introduce randomness in the classification process, this may decrease the utility of the responses [31].
18. Limit the feedback that is provided to the attacker or provide intentionally misleading responses. This may reduce the utility of the responses [31].
19. Query number restriction [70].
20. Leverage the proposed fuzzing attack framework to improve the robustness of the defence mechanisms against bulk-generated adversarial examples [70].
21. Adversarial training on examples generated by such attacks to strengthen model robustness [70].
22. Robust feature selection method - a feature selection method, considering the feature importance for classification, the costs of manipulating of each feature, probabilistically selecting features inversely proportional to their attack vulnerability [71].
23. Ensemble learning - combination of multiple classifiers trained on different feature subsets; the classifiers are designed so that all the features are integrated and that classifiers differentiate from each other [71].
24. Certified defense, achieving a constant model prediction within a specific bound; a base classifier is trained with noise to average model's output over noisy samples [70].
25. Scrambling: an operation for radio unit (RU) ordering during training and inference stages to mitigate adversarial attacks by obfuscating AI model input relationships, significantly reducing attack effectiveness. This approach is not possible int other domain, such as image, text or audio, where semantics of the data would be lost [5].

a. [IT.T.2] Adversarial Attack: Evasion Attack

Attack definition: An evasion attack is a type of adversarial attack where an adversary manipulates input data at the test or inference stage to cause a trained machine learning (ML) model to misclassify it, thus evading correct detection or classification [6], [8], [21], [27]. The core idea is to exploit vulnerabilities or blind spots in the model without altering the training data or the model's parameters [4].

System Asset: ML system input/API.

Business Asset: input data.

Security Criteria: integrity, availability of input data.

Vulnerability:

1. There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model.
2. The machine learning model may produce unexpected results, if the input incorporates features that are not supported by the training dataset's feature space.
3. There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model. There exists inherent ambiguity between the decision boundaries of a machine learning model and true decision boundaries.

4. Better fitting (overfitting) make models more vulnerable to the deviations introduced by the adversaries. There is an inverse relationship between single model fitting accuracy and robustness to adversarial evasion.
5. Machine learning model depends on the features, which can be mimicked and manipulated by the adversary.
6. Reliance of the model on non-predictive (not valuable) features.
7. The machine learning model can produce erroneous output, when the input incorporates features that exploit imperfect decision boundary produced by the learning algorithm. This may happen due to utilization of a limited training dataset or utilization of a learning algorithm with limited capacity.

Threat Agent: black-box scenario.

Attack Method:

1. Iterative probing, gradient estimation, or using a surrogate model to craft adversarial examples [3].
2. The new approach uses GANs to generate adversarial samples that resemble benign files in feature space without needing access to the model's queries or internal structure. The GAN consists of a generator network that creates realistic malware features by transforming malicious features into benign-looking distributions, and a critic network that assesses these features' benignness. This method performs well in evading detection by state-of-the-art ML detectors, including VirusTotal, without requiring queries [69].
3. The attacker produces adversarial samples that are close to the incorrect classes. Adversarial samples are provided as input to the machine learning system with an aim of causing misclassification. Gradient descent strategy can be used to solve the optimization problem to produce an adversarial sample. Fast Gradient Sign Method can be used as an alternative more computationally efficient method [4].
4. Adversarial samples are iteratively created by modifying them through a gradient-based algorithm to reduce the classifier's discriminant function value and push the sample across the decision boundary. The produced adversarial samples are submitted to the machine learning algorithm for processing [8].
5. Malicious machine learning models are trained on the features, which are analyzed by the targeted ML model and target ML model's performance against malicious model's behavior. Malicious machine learning models are utilized to produce adversarial samples. The produced adversarial samples are submitted to the target model for analysis [8].
6. Influential features are algorithmically identified through feature attribution methods. The non-essential features selected and are modified via gradient-guided optimization by computing the gradient of target model's classification function. The malicious input is iteratively modified until it gets misclassified by the target model. The produced malicious input data is submitted to the target model for analysis [8].
7. An adversarial sample is produced, accounting for the weaknesses in target ML classification capabilities, to evade its proper classification. The produced adversarial sample is submitted to the target ML model for analysis [8].
8. The attacker produces adversarial samples that are close to the target class. Adversarial samples are provided as input to the machine learning system to cause specific misclassification. Gradient descent strategy can be used to solve the optimization problem to produce an adversarial sample. Fast Gradient Sign Method can be used as an alternative more computationally efficient method [4].

Impact and Harm: Negates the integrity and/or availability of the targeted machine learning model. This leads to misclassification of benign and malicious inputs.

Security Requirements: The machine learning system must be resistant to adversarial attacks.

Security Control:

1. Randomization - introduction of unpredictability into model's response. Example approach: randomly generate and train multiple classifiers using different subsets of the feature space. The final output is aggregated from the predictions of all classifiers [3].
2. Complexity - complexity of model's decision function is increased, for example the boundary can be made non-linear or be fractalized [3].
3. Adversarial re-training, the model is retrained on the training dataset that includes adversarial samples. The cost of this method may be prohibitive [4].
4. Utilization of regularization terms that promote enclosure of the legitimate class [8].
5. Explainable AI design, utilization of methods like LIME and LASSO to learn the decision boundary around specific input points. This could be helpful in design of more robust ML models [8].
6. Gradient masking/Defensive distillation method produces a distilled model with a smoothed-out decision surface. Research has shown that this method may not be effective against evasion attacks, and the produce model may be as vulnerable as the original model [4].
7. Dimensionality reduction, Principal Component Analysis (PCA) can be used to reduce components of the classifier. This method may reduce performance of the algorithm in exchange for robustness [4].
8. Stateful analysis - query analysis, maintenance of query history and its analysis with meta-detectors [3].

b. [IT.T.12] Adversarial Attack: Man-in-the-Middle Attack

Attack definition: A Man-in-the-Middle (MitM) attack in the context of machine learning is a type of adversarial attack where an attacker stealthily intercepts and alters the communication between two parties (e.g., a data source and a machine learning classifier) to deliver malicious payloads or manipulate the data, with the aim of compromising the integrity or availability of the ML system [33].

System Asset: Machine learning model.

Business Asset: input data.

Security Criteria: integrity of input data.

Vulnerability: There exist special inputs that are close to correctly classified samples but are completely misclassified by a machine learning model.

Threat Agent: white-box and black-box scenarios.

Attack Method [33]: 1. Train a Variational Autoencoder (VAE) with a malicious decoder (MVD). 2. MVD is fine-tuned to produce adversarial samples. 3. Integrate the VAE with the trained MVD into the chain either between raw input data source and the classifier, or swapping out the existing decoder with MVD from the present VAE. 4. As data passes through, the MVD transforms encoded representation of the input data into an adversarial example.

Impact and Harm: Negates the integrity of the targeted machine learning model. This leads to misclassification of benign and malicious inputs.

Security Requirements: The machine learning system must be resistant to adversarial attacks.

Security Control:

1. Randomization - a technique, randomizing model's input [33].
2. Adversarial training: the model is trained against adversarial samples, to increase it robustness [33].

c. [IT.T.3] Jailbreak Attack

Attack definition: A jailbreak attack is a type of security attack that exploits vulnerabilities within a constrained system (such as an aligned LLM) to bypass imposed restrictions and achieve privilege escalation [2]. In the context of LLMs, jailbreaking refers to the practice of circumventing or overriding alignment guardrails that are designed to govern the scope of content the model can produce [2].

System Asset: ML system input/API.

Business Asset: input data.

Security Criteria: confidentiality, integrity of input data.

Vulnerability:

1. It is possible to bypass built-in output restrictions of a VLM by providing an adversarial image or textual data.
2. There exists inherent ambiguity between the decision boundaries of a machine learning model and true decision boundaries.
3. Lack of or insufficiency in defensive measures.
4. Vulnerability of deep neural networks to small, almost imperceptible perturbations to benign examples.
5. Transferability of adversarial examples from surrogate to target models.
6. Increased vulnerability in multi-modal setup.
7. Model's API misconfiguration.
8. Stochastic nature of the machine learning model.

Threat Agent: white-box and black-box scenarios.

Attack Method:

1. A submission of an adversarial image input alongside a textual prompt, requesting a malicious output. Adversarial image data is generated based on examples of malicious content through Projected Gradient Descent (PGD), this image data is later paired with malicious textual prompt [2].
2. A submission of an adversarial textual input alongside a textual prompt, requesting a malicious output. A discrete optimization algorithm from Shin et al., an improved version of the hotflip attacks can be utilized for adversarial text generation. [2]
3. [23] source defines the following attack method: 1. A surrogate LLM is fine-tuned. 2. Optimize adversarial distribution using Gumbel-Softmax. 3. Apply constraint model (CLM) for perplexity/semantic regularization. 4. Use geometric loss to balance objectives. 5. Sample adversarial examples with semantic filtering. 6. Submit the generated adversarial data as an input to the target LLM system.
4. Input of a malicious prompt, bypassing present safeguards [10].

Impact and Harm: Negates the integrity of the targeted machine learning model. This may lead to legal repercussions.

Security Requirements: The machine learning system must be resistant to jailbreak attacks.

Security Control:

1. Conduct adversarial training, the cost is prohibitive. In addition, the bounds on the perturbation can be much wider that generally assumed [2].
2. Utilize common harmfulness detection APIs like Perspective API and Moderation API. The API's accuracy is limited, they may cause bias, harm and reduce the helpfulness of a model. These API's are not applicable to offline models in adversary's possession [2].
3. Conduct robustness certification, although the cost is prohibitive [2].
4. The input can be pre-processed. A possible method is DiffPure, which introduces noise into the input images and then diffuses it back into learned data manifold, thus restoring a clean image. This method cannot be applied to an offline, local model in adversary's possession [2].
5. Model behavior constraints: definition of system prompts, restricting model's capabilities, role, accepted output, actions and topics [10].
6. Deterministic validation of the output [10].
7. Filter input and output: assess the context, consider semantic filtering; sanitize the input data [10].
8. Restrict LLM application's privileges [10].
9. Human-in-the-loop control and validation of privileged operations caused by the LLM application [10].
10. Separation of the external input data and limitation of its influence [10].
11. Conduct penetration testing [10].
12. Restrict model's access to sensitive and external data sources [10].
13. Federated learning, reduces the need for centralized data collection and by extension reduces data exposure risk [10].
14. Differential privacy - addition of noise to the output, reduces the data exposure risk [10].
15. Education of users on the risks of sensitive data input [10].
16. Define clear and transparent policy about data retention, usage, removal [10].
17. Conceal system prompt and configuration [10].
18. Proper and secure configuration of the model's input API, to restrict feedback on error and configuration details [10].
19. Homomorphic encryption - to ensure secure data analysis and training process [10].

iii. [IT.T.4], [TD.T.1] Model Inversion Attack

Attack definition: A model inversion attack is a type of privacy attack where an adversary aims to reconstruct training samples from a machine learning model by exploiting the model's outputs [9]. The goal is to infer specific features or attributes of the hidden input data used to train the model [32]. This type of attack allows an adversary to directly learn information about the training dataset [9].

System Asset: ML system input/API.

Business Asset: input data, training data.

Security Criteria: confidentiality of input data and training data.

Vulnerability:

1. Model tends to memorize its training data in case of over-parametrization. It is possible to learn additional information about the training data sample from the target model's output.
2. It is possible to recover the features of the training data based of predictions retrieved from the target machine learning model.
3. Model tends to memorize its training data in case of over-parametrization.
4. It is possible to extract confidential data from the trained machine learning system, trained on confidential data.
5. It is possible to learn additional information about the training data sample from the target model's output.

Threat Agent: white-box and black-box scenarios.

Attack Method:

1. One of the methods: outputs of different model versions are recorded; Multi-Layer Perceptron is utilized to analyze the difference between the version outputs, producing information about the target training data. Other possible techniques: LSB encoding; Correlated value encoding; Sign encoding, Using Model Overfitting, Neuron sorting; Set-based representation, Training-based strategy values, Recognition related neuron, Training attack classifier, Training Shadow GAN, Poisoning attack, Analysis of the confidence score , Training a meta-classifier, Design a multi-task GAN, Regularized Maximum Likelihood Estimation; Inverse-Network, Deep Leakage from Gradients, Numerical reconstruction Matching virtual and shared gradients, Equality solving; Path Restriction, Direct/passive/active label inference attack [35].
2. [34] source defines the following attack method: 1. A custom feature extractor is trained upon auxiliary data. 2. The public, auxiliary data is fed to the feature extractor and the target model to produce feature-prediction probability pairs. 3. An inverse model is trained on the feature-prediction probability data, mapping prediction data to the feature space. 5. The inverse model produces a feature based on the provided label data. 4. Output of a GAN model (trained on public, auxiliary data) is produced based on the inverse model's generated features. 5. The produced sample by GAN is fed to the target model to acquire new predictions. 6. Features are extracted from the generated image and compared to the features previously produced by the inverse model, accounting for the new predictions. 7. Thus, the features are updated iteratively with differential evolution (DE) optimization algorithm. 8. A new image is generated with the GAN model based on the updated features.

Impact and Harm: Negates the confidentiality of the targeted machine learning model. This may lead to leakage of sensitive data. Exposure of sensitive private data may lead to legal repercussions.

Security Requirements: The machine learning system must be resistant to model inversion attacks.

Security Control:

1. Degradation of precision accuracy. This may result in slight degradation of the attack's performance [34].
2. The prediction results can be rounded down or replaced with null data based on the threshold. This results in degradation of the attack's performance [34].
3. Differential privacy: addition of noise to deviate the outputs from the original [35].

4. Secure multi-party computation: joint computations are conducted within confidential environment [35].
5. Homomorphic encryption: calculations are conducted through confidential means, allowing operations on encrypted data without revealing the original data [35].
6. Adversarial machine learning: incorporation of data about adversarial techniques into the model's training process [35].
7. Differential-privacy-based method obscures the input by adding noise to the original data model. Possible methods: randomized aggregative privacy-preserving ordinal response (RAPPOR) method, PATE (Private Aggregation of Teacher Ensembles). The PATE method trains a teacher model on disjoint subsets of data and then a student model is trained on the teacher's output [28].
8. Homomorphic-encryption-based method allows for substantial amount of important data to be securely transmitted in a cloud environment. Alternatively, a CryptoNets model was developed, which encrypts the model parameters [28].
9. Vulnerability detection: risk assessment method for machine learning mode, evolution of the model's pre-release [35].

iv. [IT.T.5] Model Extraction Attack

Attack definition: A model extraction attack (also known as model stealing) is a type of security attack where an adversary aims to replicate the functionality of a target machine learning model without having direct access to its internal parameters or training data [3], [18], [35], [36], [37]. The attacker interacts with the target model, typically through a prediction API, to gather information and train a substitute model that mimics the behavior of the original [3], [35], [37]. This allows the adversary to gain insights into the training data and potentially launch further attacks, such as evasion or membership inference attacks [3], [37].

System Asset: ML system input/API.

Business Asset: input data.

Security Criteria: confidentiality of input data.

Vulnerability:

1. It is possible to replicate target model's properties within a new model based on the target machine learning model's output.
2. It is possible to possible to reconstruct target model's functionality from the target model's output.
3. *There are no restrictions on the amount and frequency of inference requests* [10].
4. *Operation of the machine learning has high computational demands and a high cost* [10].
5. It is possible to replicate target model's properties within a new model based on the target machine learning model's output.

Threat Agent: white-box, gray-box and black-box scenarios.

Attack Method:

1. The target model is queried with surrogate data, based on the public data, creating a dataset of surrogate data-target model output data pairs. A publicly available pre-trained Data-efficient Transformer (DeiT) model is fine-tuned based on the previously produced dataset [37].

2. One of the possible methods: A set number of requests is produced against the target model, producing a set of responses. A surrogate model is trained based on request data-response data pairs. Other possible techniques: Linear least square approach, Malicious samples query, Build universal thief datasets, Query synthesis active learning, Autoregressive generation, Fine-tuned encoder - Algebraic attack, Direct Extraction - Recreate Projection, Head Fuzzy gray correlation [35].
3. *Querying of model's input API with malicious input samples for collection of data, sufficient for model replication* [10].
4. The target model is probed, the received output is utilized to build a clone model. The cloned model can be utilized for further attacks, such as an evasion attack [3].
5. The attacker utilizes a partial training dataset or shadow set to query the target model to receive its posteriors. The retrieved posteriors and the utilized samples are used to train the new adversary's model [9].

Impact and Harm: Negates the confidentiality of the targeted machine learning model. This may lead to the loss of intellectual property.

Security Requirements: The machine learning system must be resistant to model extraction attacks.

Security Control:

1. Prediction poisoning, addition of bounded noise to the model's output for maximization of the annular deviation (MAD) between the original gradients and poisoned ones, while maintaining the rank of the most confident prediction, to prevent accuracy loss [37].
2. Reverse Sigmoid, addition of an activation layer, which controls the amount of perturbation added to the target model's posterior probabilities, to prevent loss of the model's accuracy [37].
3. Differential privacy: addition of noise to deviate the outputs from the original [35].
4. Secure multi-party computation: joint computations are conducted within confidential environment [35].
5. Homomorphic encryption: calculations are conducted through confidential means, allowing operations on encrypted data without revealing the original data [35].
6. Adversarial machine learning: incorporation of data about adversarial techniques into the model's training process [35].
7. Watermarking techniques: embedding of watermarks into model's parameters, or algorithmic analysis of the model due to over-parametrization, or entanglement between watermark and training data features [35].
8. *Rate limit the amount of requests from a particular user* [10].
9. *Watermark the model's outputs to detect their unauthorized usage* [10].
10. Adversarial training, training the model for detection and resistance against the extraction attempts [10].
11. Vulnerability detection: risk assessment method for machine learning mode, evolution of the models pre-release [35].
12. Randomization - introduction of unpredictability into model's response. Example approach: randomly generate and train multiple classifiers using different subsets of the feature space. The final output is aggregated from the predictions of all classifiers [3].
13. Complexity - complexity of model's decision function is increased, for example the boundary can be made non-linear or be fractalized [3].
14. Stateful analysis - query analysis, maintenance of query history and its analysis with meta-detectors [3].

15. Utilize Differential Privacy, Differentially-Private Stochastic Gradient Descend (DP-SGD). This method adds Gaussian noise to gradient during the target model's training. The method is capable of mitigating inference attacks without significantly deteriorating model's utility [9].
16. Knowledge Distillation (KD) method can reduce membership inference attack risks. KD method transfers knowledge from the larger target model to the smaller distilled model. The distilled model can be more resource efficient that the original model [9].

a. [MP.T.2] Model Extraction Attack: Side-Channel Attack

Attack definition: Model extraction attack, utilizing side-channel attack methods, leverages hardware implementation vulnerabilities to determine target machine learning model's architecture and parameter values [35].

System Asset: Processing hardware running the ML model.

Business Asset: model's parameters.

Security Criteria: confidentiality of model's parameters.

Vulnerability: The state of machine learning model's operational component data can be inferred through measurable attributes of the machine learning, such as power consumption, EM emission, memory access pattern.

Threat Agent: gray-box scenario.

Attack Method: A malicious fault is introduced into the operations of hardware, processing target machine learning model's operations [35].

Impact and Harm: Negates the confidentiality of the targeted machine learning model. This may lead to the intellectual property theft.

Security Requirements: The machine learning system must be resistant to model extraction attacks.

Security Control:

1. Differential privacy: addition of noise to deviate the outputs from the original [35].
2. Secure multi-party computation: joint computations are conducted within confidential environment [35].
3. Homomorphic encryption: calculations are conducted through confidential means, allowing operations on encrypted data without revealing the original data [35].
4. Adversarial machine learning: incorporation of data about adversarial techniques into the model's training process [35].
5. Watermarking techniques: embedding of watermarks into model's parameters, or algorithmic analysis of the model due to over-parametrization, or entanglement between watermark and training data features [35].
6. Vulnerability detection: risk assessment method for machine learning mode, evolution of the models pre-release [35].

v. [IT.T.13], [TD.T.4] Membership Inference Attack

Attack definition: A membership inference attack (MIA) is a type of privacy attack where an adversary tries to determine whether a specific data record was part of the training dataset of a machine learning model [6], [20], [36], [38], [39]. It exploits the tendency of ML models to behave differently on data they have been trained on compared to unseen data [6], [36], [38],

[38], [39]. A successful MIA signifies that the privacy of the training data was not sufficiently protected when the trained ML model is released [39].

System Asset: machine learning model, ML system input/API.

Business Asset: training data, input data.

Security Criteria: confidentiality of training data and input data.

Vulnerability:

1. There is a difference in gradient behavior between training (member) and non-training (non-member) records, which reflects membership information due to overfitting or gradient convergence during training.
2. The more output classes the model has, the more data is leaked.
3. It is possible to learn additional information about the training data sample from the target model's output.
4. It is possible to infer membership of samples within the target machine learning model.

Threat Agent: white-box and black-box scenarios.

Attack Method:

1. GAN's discriminator model is used to deduce whether a particular data instance was part of the training dataset of a target model. This is achieved by exploiting differences in the model's response to known (trained on) versus unknown input data [38].
2. Shadow models are trained on datasets similar to the target model’s dataset to replicate its behavior. An attack model is trained on the shadow models' outputs to distinguish between training and non-training data based on prediction behavior. The attack model is utilized if the target sample is a part of the targeted model's training dataset [20].
3. Synthesis of generated data through optimization against the latent code. The distance between the generated data and target data is measured to determine membership [38].
4. The target model is requested with perturbed features. With the target model's outputs toward requests a local linear regression model is trained. An autoencoder is to extract membership features from the approximated gradients for training of the local attacks model, which would be used to classify if the target sample is a member of the target model's training dataset or not [39].
5. [9] source defines the following attack method: 1. An attacker trains a shadow model on the part of the shadow dataset, acquired from the same distribution as the target model's training data. The model is trained to determine if the requested sample is a part of the shadow model's training dataset or not. Finally, the attacker queries the shadow model with the entire shadow dataset, to determine, if it is a part of the originally targeted model. 2. Based on the produced prediction and the shadow training data set, an attack model is trained, a binary membership classifier. (If the attacker has a part of the original training dataset, the attack model can be trained directly.) 3. The attacker then queries the target model with a target sample to determine if the sample is a part of its dataset. 4. If the attacker has access to the targeted model, then the target model's gradients can be incorporated into the training of the attack model. The attack model is then used to determine if the target sample is a part of the targeted model, based on the posteriors and predicted label for the supplied target sample.
6. An attacker with white box access feeds a noise sample to the targeted model to receive its posteriors. Afterwards, through back-propagation over model's parameters the input is optimized, thus producing a representative sample of a class [9].

7. With a shadow dataset a generative adversarial network (GAN) can be trained. The GAN model is provided with optimized inputs, with an aim of generating samples that reach high posteriors on the target model [9].
8. The attacker utilizes embeddings of the target sample from the target model to train a classifier, which will predict the sample's target attributes [9].

Impact and Harm: Negates the confidentiality of the targeted machine learning model. This leads to privacy leakage and potential legal repercussions.

Security Requirements: The machine learning system must be resistant to membership inference attacks.

Security Control:

1. The defensive mechanism utilized in this study is Differential Privacy (DP), applied to Generative Adversarial Networks (GANs) to protect against privacy invasion attacks, specifically membership inference attacks. Differential privacy introduces controlled noise into the model training process to obscure the presence or absence of any single data point in the training set, thus aiming to protect individual data privacy [38].
2. Differential privacy: calibrated noise is added to the gradients during training, applied gradient clipping to bound the sensitivity of the training algorithm [38].
3. Regularization - a technique that reduces the overfitted nature of a model. It generalizes the model and reduces information leakages about the training dataset [20].
4. Differential privacy: adds mathematically bounded noise to the training process or to the final model parameters, limiting the influence of any single data record on the model's outcomes [20].
5. Utilize Differential Privacy, Differentially Private Stochastic Gradient Descend (DP-SGD). This method adds Gaussian noise to gradient during the target model's training. The method is capable of mitigating inference attacks without significantly deteriorating model's utility [9].
6. Knowledge Distillation (KD) method can reduce membership inference attack risks. KD method transfers knowledge from the larger target model to the smaller distilled model. The distilled model can be more resource efficient than the original model [9].

a. [TD.T.2] Membership Inference Attack: Shadow Training Method

Attack definition: The Shadow Training Method is an approach used in membership inference attacks where the attacker trains multiple "shadow models" to mimic the behavior of the target machine learning model [9], [20], [35], [39]. The attacker uses these models to understand how the target model might behave differently on data it has seen during training versus data it has not [20], [39].

System Asset: ML system input/API.

Business Asset: training data.

Security Criteria: confidentiality of training data.

Vulnerability:

1. Model tends to memorize its training data in case of over-parametrization.
2. It is possible to learn additional information about the training data sample from the target model's output.

Threat Agent: white-box and black-box scenarios.

Attack Method: Multiple shadow models are built imitating the target model's structure and training process. These shadow models are trained on datasets where the membership status is known. The outputs (e.g., confidence scores) are collected from these shadow models for both member and non-member data. An attack model is then trained to distinguish between members and non-members based on these outputs. The attacker queries the target model with data of unknown membership and uses the attack model to infer membership status against the target model, covering the same domain [35].

Impact and Harm: Negates the confidentiality of the targeted machine learning model. This may lead to private data leakage and possible legal repercussions.

Security Requirements: The machine learning system must be resistant to malicious membership inference attacks.

Security Control: Differential privacy: addition of noise to deviate the outputs from the original. Secure multi-party computation: joint computations are conducted within a confidential environment. Homomorphic encryption: calculations are conducted through confidential means, allowing operations on encrypted data without revealing the original data. Adversarial machine learning: incorporation of data about adversarial techniques into the model's training process. Vulnerability detection: risk assessment method for machine learning mode, evolution of the model's pre-release [35].

b. [TD.T.3] Membership Inference Attack: Metric-Based Method

Attack definition: A metric-based membership inference attack is an approach where the attacker calculates a metric on the prediction vectors of a data record and compares it to a predetermined threshold to determine its membership status [35]. This type of attack is generally simpler and less computationally expensive than shadow model-based attacks [35].

System Asset: ML system input/API.

Business Asset: training data.

Security Criteria: confidentiality of training data.

Vulnerability: Model tends to memorize its training data in case of over-parametrization. It is possible to learn additional information about the training data sample from the target model's output.

Threat Agent: white-box and black-box scenarios.

Attack Method: Observer target machine learning model's behavior on training data versus unseen data. Analysis of metrics such as confidence scores, loss values, or entropy, patterns indicative of membership are identified. Statistical thresholds or anomalies in the metrics are utilized to determine the membership of the target sample. Membership is inferred when the metrics for a given input differ significantly from those expected for non-members [35].

Impact and Harm: Negates the confidentiality of the targeted machine learning model. This may lead to private data leakage and possible legal repercussions.

Security Requirements: The machine learning system must be resistant to malicious membership inference attacks.

Security Control: *Same as described in the a. [TD.T.2] Membership Inference Attack: Shadow Training Method subchapter.*

vi. [IT.T.11] Cyber-Physical Attack

Attack definition: Cyber-physical attacks against machine learning models refer to attacks that exploit the interaction between the cyber (computing and communication) components and the physical components of a system [14]. These attacks target machine learning models that are integrated into cyber-physical systems (CPS), aiming to cause physical consequences by manipulating the data, the training process, or the model itself [14].

System Asset: processing hardware running the ML model.

Business Asset: input data.

Security Criteria: availability of input data.

Vulnerability: A machine learning system is susceptible to vulnerabilities in software and hardware, on which it depends.

Threat Agent: white-box and black-box scenarios.

Attack Method: Exploit vulnerability in the software or hardware is exploited, on which the target machine learning system depends [14].

Impact and Harm: Negates the availability of the targeted machine learning model.

Security Requirements: The machine learning system must be resistant to malicious cyber-physical attacks.

Security Control: None stated in the target paper.

a. [MOD.T.2] Malicious Hardware Fault Injection

Attack definition: A malicious hardware fault injection attack is a type of hardware-oriented security threat where an adversary intentionally introduces faults or errors into the physical hardware on which a machine learning model is running [40]. This is done to compromise the model's integrity, leading to misclassification or other undesirable behaviors [40]. Unlike software-based attacks, hardware fault injections directly manipulate the ML model's parameters and computation results by tampering with the inference process without manipulating the sample or training data [40].

System Asset: processing hardware running the ML model.

Business Asset: model's operational data.

Security Criteria: integrity of model's operational data.

Vulnerability: The state of machine learning model's operational data can be influenced through hardware state manipulation with fault injection.

Threat Agent: white-box and black-box scenarios.

Attack Method: A malicious fault is introduced into the operations of hardware, processing target machine learning model's operations [40].

Impact and Harm: Negates the integrity of the targeted machine learning model. This may lead to misclassification of benign and malicious inputs.

Security Requirements: The machine learning system must be resistant to malicious hardware fault injection attacks.

Security Control:

1. Binarization method: mimic bit-flip noise on the weights, thus increasing robustness against bit-flip attacks [40].
2. Piece-wise clustering method: adds fixed single bit-width constraint during the training, this increasing robustness against bit-flip attacks [40].
3. Weight reconstruction: averages errors over a grain of weights with their quantization and clipping, thus increasing robustness against bit-flip attacks [40].
4. Defensive quantization: constrains the Lipschitz constant during training to limit mapping sensitivity [40].
5. Hardware with Triple Modular Redundancy (TMR): three copies of the functional circuits are present, majority vote determines correction and masking of faults in copied; imposes higher energy an resource overhead [40].
6. DNN (Deep Neural Networks) accelerator: tolerant to SRAM read faults from voltage variations [40].
7. Word masking and bit masking: round faulty bits to zero; a whole word reset to zero or flipped bits are rest to zero respectively [40].
8. TE-Drop: an error-tolerant design for the MAC units, for example utilizing Razor flip flops module for active fault detection; the detected error is dropped. Hardening of selective memory cells. Application of modular redundancy on sensitive weights. Possible direction - explainable AI, if both inductive and deductive reasonings are incorporated together, this could reduce frequency of logical fallacies [40].
9. Trusted Inference Engine (TIE): Pseudo Random Number Generators (PRNG) and PUF (Physically Unclonable Function) are utilized to decrypt the encrypted machine learning model, stored on off-chip memory. A DNN accelerator with a memory encryption engine, encrypting data in DRAM; also utilizing Integrity Verification (IV) engine for detection of unauthorized operations on the data from the external memory; comes with low overhead (this defense does not account for hardware side-channels). Possible direction - explainable AI, if both inductive and deductive reasonings are incorporated together, this could reduce frequency of logical fallacies [40].

b. [MOD.T.1] Hardware Trojan Attack

Attack definition: A Hardware Trojan (HT) attack is a type of attack where malicious circuitry is covertly inserted into the hardware that implements a machine learning model [40]. This hidden circuitry, once triggered, can manipulate the model's behavior, leading to misclassification, data leakage, or other security breaches [40]. Unlike software-based attacks, HT attacks directly target the hardware to alter the model's parameters and computation results [40].

System Asset: processing hardware running the ML model.

Business Asset: model's operational data.

Security Criteria: confidentiality, integrity, availability of model's operational data.

Vulnerability: The state of machine learning model's operational data can be influenced through a hardware trojan inserted into the integrated circuits.

Threat Agent: black-box scenario.

Attack Method: A hardware trojan is embedded into an integrated circuit. The embedded hardware trojan modifies the chain of operations based of a trigger, thus modifying the expected chain of operations during machine learning operation [40].

Impact and Harm: Negates the confidentiality, integrity and availability of the targeted machine learning model. This may lead to misclassification of benign and malicious inputs, degraded performance or private data leakage.

Security Requirements: The machine learning system must be resistant to malicious hardware trojan attacks.

Security Control:

1. MAC (Multiply-and-Accumulate) operations, low-precision data representations, bound-constrained dynamic range compression: limit error propagation and aggregation; quantizing data to lower bit precision reduces the proportion of vulnerable parameters. Activation output clipping [40].
2. Hardware root-of-trust: safeguarding DNN IP cores and private data. An obfuscation framework employing key-dependent backpropagation algorithm to lock some neurons of the model; on-chip key can recover the correct functionality of the model [40].
3. *Same as described in the a. [MOD.T.2] Malicious Hardware Fault Injection subchapter.*

c. [IT.T.10], [MP.T.1] Side-Channel Attack

Attack definition: A hardware side-channel attack is a type of attack that exploits vulnerabilities in the physical implementation of a machine learning (ML) model to extract sensitive information, such as model parameters, training data, or the model's architecture [40]. Instead of targeting the ML algorithm directly, these attacks measure and analyze side-channel information that is correlated with the ML assets [40].

System Asset: processing hardware running the ML model.

Business Asset: model's parameters, input data.

Security Criteria: confidentiality of model's parameters and input data.

Vulnerability: The state of machine learning model's operational component data can be inferred through measurable attributes of the machine learning, such as power consumption, EM emission, memory access pattern.

Threat Agent: black-box scenario.

Attack Method: Measurable attributes of hardware processing target machine learning model's operation are observed. The observed and measured metrics are analyzed to infer the machine learning model's structure [40].

Impact and Harm: Negates the confidentiality of the targeted machine learning model. This may lead to the loss of intellectual property.

Security Requirements: The machine learning system must be resistant to malicious side-channel attacks.

Security Control:

1. Internal operation shuffling: order of execution is mixed to modify the scheduled operation time [40].
2. Computation masking: taints sensitive operations with random values, to eliminate dependencies between the private data and the side-channel attributes [40].
3. Augmenting masking: masks adder trees and ReLU (Rectified Linear Unit) [40].

4. BoMaNet masking: uses gate-level Boolean masking for splitting secrets, thus reducing relation between secret related computations and side-channel attributes [40].
5. Data quantization: mitigates weight matrices leakage based of off cache access patterns [40].
6. Cache partitioning: distinct portions of the last-level cache are allocated to different applications to eliminate cache interference between the attacker and the victim. Reduction of hardware profiler's precision: reduces side-channel leakage from context-switching penalties [40].
7. Oblivious RAM (ORAM:) reduces side-channel information leakage based of off memory patterns an timing, shuffles and re-encrypts the data to conceal access pattern [40].
8. Memory-Trace Obliviousness (MTO): reduces side-channel information leakage based off memory patterns an timing. Creation of fake memory access with TIE (Trusted Inference Engine) [40].
9. Randomization of the width of the coalescing unit and merge of transactions: mitigates GPU memory timing side-channel attacks [40].
10. GPUGuard: detects spy programs through a decision tree method, thus mitigating side-channel attacks. Possible direction - explainable AI, if both inductive and deductive reasonings are incorporated together, this could reduce frequency of logical fallacies [40].

vii. [IT.T.6] Property Inference Attack

Attack definition: A property inference attack is a type of privacy attack that aims to infer confidential information or attributes about the training dataset used to train a machine learning model [35]. The attacker attempts to determine certain properties or characteristics that are present in the training data, which the model provider does not want to reveal [35]. This attack doesn't directly manipulate the model but extracts private information without disrupting the model's normal training process [35].

System Asset: ML system input/API.

Business Asset: input data.

Security Criteria: confidentiality of input data.

Vulnerability: Model tends to memorize its training data in case of over-parametrization. It is possible to learn additional information about the training data sample from the target model's output.

Threat Agent: white-box and black-box scenario.

Attack Method: Target model's output and parameters are analyzed to determine properties of the utilized training dataset [35].

Impact and Harm: Negates the confidentiality of the targeted machine learning model. This may lead to private data leakage and possible legal repercussions.

Security Requirements: The machine learning system must be resistant to malicious property inference attacks.

Security Control:

1. Differential privacy: addition of noise to deviate the outputs from the original [35].

2. Secure multi-party computation: joint computations are conducted within confidential environment [35].
3. Homomorphic encryption: calculations are conducted through confidential means, allowing operations on encrypted data without revealing the original data [35].
4. Adversarial machine learning: incorporation of data about adversarial techniques into the model's training process [35].
5. Vulnerability detection: risk assessment method for machine learning mode, evaluation of the models pre-release [35].

viii. [IP.T.1] Fingerprinting Attack

Attack definition: A fingerprinting attack aims to uniquely identify a specific machine learning model instance or to determine which model or family of models is being used in a black box setting [41]. The goal is to derive a signature that is unique to a particular model, similar to human fingerprint biometry [41].

System Asset: ML system input/API.

Business Asset: intellectual property (IP).

Security Criteria: confidentiality of intellectual property.

Vulnerability: The model's unique decision boundaries and output patterns serve as a "fingerprint"; benign inputs reveal characteristic outputs.

Threat Agent: black-box scenario.

Attack Method: A set of benign queries are sent to the target set of models. The responses from models are compared to the known response from the targeted model for statistical similarity. [41]

Impact and Harm: Negates the confidentiality of the targeted machine learning model.

Security Requirements: The machine learning system must be resistant to fingerprinting attacks.

Security Control:

1. Randomized smoothing - addition of noise to the input, output classes are aggregated [41].
2. Pruning of the last layer [41].

ix. [IT.T.7] Denial of Service Attack

Attack definition: A Denial of Service (DoS) attack aims to disrupt the normal functioning and reduce the availability of a machine learning system, making it unusable for legitimate users [28], [31]. This is typically achieved by overwhelming the system with a high volume of requests or resource-intensive tasks, exhausting its computational resources [10].

System Asset: ML system input/API.

Business Asset: input data.

Security Criteria: availability of input data.

Vulnerability:

1. The target system hosting the machine learning system can be overwhelmed and made unavailable with computationally expensive requests.
2. There is no restrictions on the amount and frequency of inference requests.
3. Operation of machine learning has high computational demands.

Threat Agent: white-box and black-box scenarios.

Attack Method:

1. The adversary overwhelms the system by creating many computationally expensive input requests [31].
2. Continuously conduct a high frequency of computationally demanding inference requests [10].

Impact and Harm: Negates the availability of the targeted machine learning model. This leads to potential system failures, service unavailability, an increase in processing times, and a reduction in quality of service.

Security Requirements: The machine learning system must be resistant to denial-of-service attacks.

Security Control:

1. Limit the feedback that is provided to the attacker or provide intentionally misleading responses. This may reduce the utility of the responses and computational cost of the request [31].
2. Input validation, validate the size [10].
3. Dynamically allocate necessary resources for model's performance [10].
4. Rate limit the amount of requests from a particular user [10].
5. Throttle and timeout resource-intensive on-going model operations [10].
6. Monitor the resource usage for possible anomalies [10].
7. Design graceful degradation, achieve partial functionality under heavy load [10].
8. Regulate and load-balance queued actions [10].

x. [IT.T.8] Denial of Wallet Attack

Attack definition: A Denial of Wallet (DoW) attack is a type of attack where an adversary exploits the cost-per-use model of cloud-based AI services by generating an excessive number of operations or resource-intensive tasks [10]. This leads to unsustainable financial burdens on the service provider, potentially causing financial strain or even financial ruin [10].

System Asset: ML system input/API.

Business Asset: input data.

Security Criteria: availability of input data.

Vulnerability: High cost of model services' operation.

Threat Agent: black-box scenarios.

Attack Method: Inversion of embeddings, leading to recovery of source information. Utilize gradient-based (white-box) or learning-based (black-box) methods to invert the target embeddings. The produced mappings will partially reveal the original input to the model [10].

Impact and Harm: Negates the confidentiality of previously provided input to the machine learning system, by extension model's confidentiality is compromised in addition. This may lead to legal repercussions.

Security Requirements: The machine learning system's actions and decisions must be resistant to embedding inversion attacks.

Security Control:

1. Implement permission and access control to the embedding store [10].
2. Monitor and log the data retrieval activities [10].
3. Audit and validate integrity of the data stores [10].
4. Operate with data retrieved only from the trusted sources [10].
5. Monitor an devaluate the influence of RAG on the model's performance [10].

xi. [IDE.T.1] Embedding Inversion Attack

Attack definition: An Embedding Inversion Attack exploits vulnerabilities to invert embeddings and recover significant amounts of source information, compromising data confidentiality [10].

System Asset: Supporting IT infrastructure.

Business Asset: input data embeddings.

Security Criteria: confidentiality of input data embeddings.

Vulnerability:

1. Embedding vectors retains enough information to enable reconstruction of significant parts of the original text.
2. The leakage or unauthorized access to the embeddings.

Threat Agent: white-box and black-box scenarios.

Attack Method: Inversion of embeddings, leading to recovery of source information. Utilize gradient-based (white-box) or learning-based (black-box) methods to invert the target embeddings. The produced mappings will partially reveal the original input to the model [10].

Impact and Harm: Negates the confidentiality of previously provided input to the machine learning system, by extension model's confidentiality is compromised in addition. This may lead to legal repercussions.

Security Requirements: The machine learning system's actions and decisions must be resistant to embedding inversion attacks.

Security Control:

1. Implement permission and access control to the embedding store [10].
2. Monitor and log the data retrieval activities [10].
3. Audit and validate integrity of the data stores [10].
4. Operate with data retrieved only from the trusted sources [10].
5. Monitor an devaluate the influence of RAG on the model's performance [10].

xii. [MLMDP.T.1] Supply Chain Attack

Attack definition: A supply chain attack targets vulnerabilities in the machine learning (ML) supply chain to compromise the integrity, security, and trustworthiness of ML models and their

deployment platforms [10]. These attacks exploit weaknesses in third-party components, training data, pre-trained models, and deployment infrastructure [10].

System Asset: enabling software components.

Business Asset: machine learning model development process.

Security Criteria: integrity, availability of machine learning model development process.

Vulnerability:

1. Deployment platform vulnerabilities.
2. Traditional software vulnerabilities: code and dependencies flaws.
3. Corrupted, vulnerable 3rd party pre-trained models.
4. Usage of unmaintained software, models.
5. Vulnerable LoRa adapters, a malicious adjustment of the pre-trained model.
6. Development platform vulnerabilities.

Threat Agent: black-box scenarios.

Attack Method: Infection or impersonation of software components, public machine learning models, development environment or data resource, on which the target machine learning model system depends on [10].

Impact and Harm: Negates the integrity and availability of the targeted machine learning model. This attack may lead to the loss of integrity of training data, deployment platform, biased output, security breach.

Security Requirements: The machine learning system's actions and decisions must be resistant to supply chain attacks.

Security Control:

1. Vet data sources, suppliers, terms and conditions, privacy policies [10].
2. Regular review and audit suppliers' security and terms [10].
3. Vulnerability scanning [10].
4. Vulnerable software patching or a virtual patching [10].
5. Remove unused dependencies and unnecessary features [10].
6. Monitor dependencies for their state, versions and vulnerabilities [10].
7. Source components and components from official source s[10].
8. Conduct red teaming, penetration testing, integrity checking against 3rd party models [10].
9. Maintain an inventory of components, a Software Bill of Materials (SBOM) [10].
10. Utilize code signing for externally supplied code [10].
11. Monitor and audit collaborative and development environments. Example: HuggingFace SF_Convertbot Scanner [10].
12. Utilize integrity checks and vendor attestation APIs against apps and models [10].

xiii. [MP.T.3] Model Manipulation Attack

Attack definition: A model manipulation attack involves an adversary directly altering the parameters, logic, or architecture of a machine learning model with the intent to compromise its performance, security, or integrity [27], [36]. This type of attack differs from traditional adversarial attacks that focus on crafting malicious input samples or poisoning training data [27]. Instead, the adversary gains access to the model itself and modifies it to achieve specific malicious goals [27].

System Asset: machine learning model.

Business Asset: model's parameters.

Security Criteria: integrity of model's parameters.

Vulnerability:

1. Lack of model integrity verification.
2. Unauthorized access to the model.

Threat Agent: white-box scenarios.

Attack Method:

1. A loss function is optimized combining cross-entropy loss to misclassify target samples and weight regularization to minimize parameter deviations from the original model, ensuring that selected malicious samples are classified as benign. The final modifications are applied to the target ML model, and malicious samples are submitted [27].
2. Using backpropagation, the final layers are optimized to force the target model into assigning incorrect labels to the chosen malicious samples while minimizing changes to overall model accuracy. Only the fully connected layers are modified through gradient descent with an added constraint, while initial convolutional layers are frozen, preserving the model's ability to generalize while misclassifying specific target malicious samples. The final modifications are applied to the target ML model, and malicious samples are submitted [27].

Impact and Harm: Negates the integrity of the targeted machine learning model. This leads to a reduction in model's accuracy.

Security Requirements: The machine learning system's actions and decisions must be resistant to model manipulation attacks.

Security Control: None stated in the target paper.

IV. Codebases

The codebases of the threat model's interactive web page and empirical test of threat applicability can be found within the following code repository: https://github.com/CybRookie/llm_threat_modeling_research_artifacts. In addition, the Excel sheet with the SLR extracted data is provided. The first "threats" sheet within entries marked with an asterisk - *, indicates that the content was not directly stated within the contents of the reviewed paper, but was inferred by the author of this thesis from the present context. In the sheet "aggregated_threats", entries references with curly brackets, mark references which analyzed threats against an LLM type of ML model. Finally, a copy of the parent JailbreakV-28K benchmark open dataset with jailbreak prompts, as well as subsets used in the experiment. Followed by the target model's responses. Considering the present harmful content, **be warned**:

The stored datasets contain offensive content that may be disturbing. This material is provided for educational and research purposes only.

License

Non-exclusive licence to reproduce the thesis and make the thesis public

I, Mihhail Karagjaur ,

(author´s name)

1. grant the University of Tartu a free permit (non-exclusive licence) to

reproduce, for the purpose of preservation, including for adding to the digital archives of the University of Tartu until the expiry of the term of copyright, my thesis

Asset-Oriented Threat Analysis for Large Language Model Systems ,

(title of thesis)

supervised by Raimundas Matulevičius ;

(supervisor´s name)

2. grant the University of Tartu a permit to make the thesis specified in point 1 available to the public via the web environment of the University of Tartu, including via the digital archives, under the Creative Commons licence CC BY NC ND 4.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright;

3. am aware of the fact that the author retains the rights specified in points 1 and 2;

4. confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Mihhail Karagjaur
15/05/2025