D1.2 SECURITY THREAT MODELING FOR AI-BASED SYSTEM ARCHITECTURE

Revision: v.1.2

Work package	WP 1
Task	Task 1.2
Due date	30/11/2022
Submission date	30/11/2022
Deliverable lead	Samuel Marchal (FSC)
Version	1.2
Authors	Samuel Marchal (FSC), Alexey Kirichenko (FSC), Andrew Patel (FSC), Michell Boerger (FOKUS), Nikolay Tcholtchev (FOKUS), Manh-Dung Nguyen (MI), Vinh Hoa La (MI), Ana Rosa Cavalli (MI), Claudio Soriente (NEC), Nicolas Kourtellis (TID), Diego Perino (TID), Andra Lutu (TID), Souneil Park (TID), Prachi Bagave (TUD), Aaron Ding (TUD), Marcus Westberg (TUD), Madhusanka Liyanage (UCD), Shen Wang (UCD), Bartlomiej Siniarski (UCD), Chamara Sandeepa (UCD), Thulitha Senevirathna (UCD)
Reviewers	Joao Gonçalvez (EUR), Drasko Draskovic (MFX)

Grant Agreement No.: 101021808
Call: H2020-SU-DS-2020
Topic: SU-DS02-2020
Type of Action: RIA

Abstract	The security of machine learning-based systems not sufficiently addressed at the present time. Methodologies for modelling threats and assessing the security posture of machine learning-based systems are required. In this document, we review existing approaches to threat modelling conventional and machine learning-based systems. We identify their limitations and provide improvement directions. Among these solutions, we identify a comprehensive list of vulnerabilities exposed by machine learning-based systems and exemplify how they can be used to infer the extent to which machine learning-based systems are exposed to security threats. We perform threat modelling of both centralized and distributed training and inference paradigms. The result of this analysis enables the identification of fine-grained security requirements for machine learning-based systems.
Keywords	Machine learning, artificial intelligence, system security, adversarial machine learning, threat modelling, vulnerability assessment, security requirements

Document Revision History

Version	Date	Description of change	List of contributors

V0.1	24/05/2022	Table of content defined	Samuel Marchal (FSC)
V0.2	15/07/2022	Chapter 2 completed	Samuel Marchal (FSC), Michell Boerger (FOKUS), Nikolay Tcholtchev (FOKUS), Manh-Dung Nguyen (MI), Vinh Hoa La (MI), Ana Rosa Cavalli (MI), Claudio Soriente (NEC), Prachi Bagave (TUD), Aaron Ding (TUD), Marcus Westberg (TUD), Madhusanka Liyanage (UCD), Shen Wang (UCD), Bartlomiej Siniarski (UCD), Chamara Sandeepa (UCD), Thulitha Senevirathna (UCD)
V0.3	11/08/2022	Chapter 3 completed	Samuel Marchal (FSC), Michell Boerger (FOKUS), Nikolay Tcholtchev (FOKUS), Manh-Dung Nguyen (MI), Vinh Hoa La (MI), Ana Rosa Cavalli (MI), Claudio Soriente (NEC), Nicolas Kourtellis (TID), Diego Perino (TID), Andra Lutu (TID), Souneil Park (TID), Madhusanka Liyanage (UCD), Shen Wang (UCD), Bartlomiej Siniarski (UCD), Chamara Sandeepa (UCD), Thulitha Senevirathna (UCD)
V0.4	01/09/2022	Chapter 4 completed	Samuel Marchal (FSC)
V1.0	07/09/2022	Full draft completed	Samuel Marchal (FSC)
V1.1	10/10/2022	Proof-reading and updates	Andrew Patel (FSC), Alexey Kirichenko (FSC), Samuel Marchal (FSC)
V1.2	21/11/2022	Internal review and revision	Samuel Marchal (FSC), Huber Flores (Tartu), Joao, Fernando Ferreira Goncalves (EUR), Drasko Draskovic (Mainflux Labs)

DISCLAIMER

COPYRIGHT NOTICE

Project funded by the European Commission in the H2020 Programme		
Nature of the deliverable:	**R***	
Dissemination Level		
PU	Public, fully open, e.g., web	✓
CL	Classified, information as referred to in Commission Decision 2001/844/EC	
CO	Confidential to SPATIAL project and Commission Services	

* R: Document, report (excluding the periodic and final reports)
DEM: Demonstrator, pilot, prototype, plan designs

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

DEC: Websites, patents filing, press & media actions, videos, etc.
OTHER: Software, technical diagram, etc

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

EXECUTIVE SUMMARY

Artificial intelligence (AI) and machine learning (ML) based systems expose new security vulnerabilities that conventional computer systems do not. In order to ensure their reliability even in adversarial conditions, ML systems must be resilient against attacks that would exploit these vulnerabilities. The recommended approach to securing computing systems is via *threat modeling* where valuable assets that compose a system are listed and security threats and vulnerabilities that can be exploited to achieve an attack against each asset are identified. The result of threat modeling describes the current security posture of a system, and it is typically used to improve it.

While several threat modeling approaches exist and can be applied to machine learning systems, they suffer from their generic yet abstract processes, which have low practical value if not used by experienced security experts. These approaches usually require a high level of technical knowledge about the system being studied, known or potential security vulnerabilities, and an understanding of how attacks work. In the case of ML applications, this knowledge is missing because vulnerabilities and attacks that exploit them are new and still largely unknown to security experts. To address this issue, several new security frameworks specifically tailored for analyzing the security posture of machine learning systems have been introduced by public organizations including ENISA, MITRE, NIST, IBM, Microsoft, and WithSecure. These frameworks address different aspects of a security assessment, from threats to vulnerabilities to impact to risk. However, each framework is an individual initiative and there isn't a lot of consistency between them. Moreover, they suffer from a lack of applicability and require significant knowledge of machine learning security. A comprehensive and usable standard for machine learning system security is yet to be defined.

To improve and make machine learning security frameworks more practical, it is necessary to demonstrate how threat modeling of machine learning systems is performed. We attempt to do this by first providing a comprehensive view of the main algorithmic, supply chain, and deployment vulnerabilities that are specific to ML systems. Then we discuss how these vulnerabilities affect the security of such systems depending on a) the phases in their lifecycle: training or inference, and b) the deployment architecture used: centralized or distributed. This document aims to strike a trade-off between specificity and generalizability of presented threat modelling examples by examining the aforementioned inference and training paradigms. We conclude that no one paradigm is better from a security standpoint and each architecture provides different trade-offs between security, privacy, and performance.

This threat modeling exercise enables the identification of new security requirements for machine learning systems to improve resilience against machine learning-specific attacks. For instance, data and model integrity must be enforced and verified during any exchange. The number of parties involved in model execution and training must be limited, and access to the ML model must be restricted and monitored. The security of the systems used for training and deployment of ML models is paramount and must be protected at all costs.

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

TABLE OF CONTENTS

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

LIST OF FIGURES

LIST OF TABLES

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

ABBREVIATIONS

AI	Artificial Intelligence
AML	Adversarial Machine Learning
API	Application Programming Interface
ASIC	Application Specific Integrated Circuit
BERT	Bidirectional Encoder Representations from Transformers
CIA	Confidentiality, Integrity, Availability
CPU	Central Processing Unit
CVE	Common Vulnerabilities and Exposures
DFD	Data Flow Diagram
DNN	Deep Neural Network
DoS	Denial of Service
DRA	Data Reconstruction Attack
ENISA	European Union Agency for Cybersecurity
FA	Federated Averaging
FPGA	Field-Programmable Gate Array
GPT	Generative Pre-trained Transformer
GPU	Graphical Processing Unit
HPS	High-Performance Computer
IID	Independent and Identically Distributed
MIA	Membership inference Attack
MITM	Man-in-the-Middle
ML	Machine learning
NIDS	Network Intrusion and Detection System
NIST	National Institute of Standards and Technology
PIA	Property Inference Attack
REST	Representational State Transfer
TPU	Tensor Processing Unit

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

1. INTRODUCTION

1.1. SCOPE AND OBJECTIVES

Artificial intelligence (AI), or more precisely, machine learning (ML) techniques are being integrated into an increasing number of systems to enable intelligent automation. As machine learning solutions become components of complex solutions, they will also be targeted by cyberattacks. Vulnerabilities in machine learning solutions can jeopardize the security of the system they are integrated into because a complex system is only as secure as its weakest component. While conventional cybersecurity is required to protect large and heterogeneous computer systems, the security of machine learning-based systems must also be addressed.

Machine learning systems are different in nature from conventional algorithms and computer programs. They learn their behavior and make decisions based on data gathered from the environment they are deployed in. Consequently, machine learning systems present different vulnerabilities, can be exposed to different threats and targeted by different attacks than conventional computer systems. This generates a situation where conventional approaches used to analyze the threats and vulnerabilities of computer systems and the means to protect them may not be best suited or effective when applied to machine learning systems.

The security of computer systems is traditionally examined using threat modeling approaches. Tried-and-tested threat modeling methods exist to analyze conventional computer systems. These are generic enough to be adapted to machine learning systems but raise challenges regarding the large customization required to apply them. This customization can only be achieved using a high level of expertise in security, AI/ML, and security of machine learning, which is currently in short supply. Thus, a few initiatives and associated frameworks have recently emerged to specifically address the security of machine learning systems. However, and by large, these are isolated and conceptually divergent, leaving no holistic standard for threat modeling of machine learning systems.

In this document, we explore security threats against machine learning-based systems and methods to identify them. We are interested only in the security threats that target machine learning systems but are irrelevant to systems not employing machine learning components. Since threats are an abstract concept, which exist only if vulnerabilities exist and can be exploited, we also cover the analysis of vulnerabilities of, and attacks against machine learning systems. We want to identify all major machine learning-specific vulnerabilities that can compromise the confidentiality, availability and integrity of machine learning models and their data. Using this knowledge of vulnerabilities and the attacks that can exploit them, we infer how machine learning systems are exposed to security threats based on a) phases in their lifecycle: training or inference, and b) deployment architecture used: centralized or distributed.

This analysis will provide a more fine-grained view and present a comparison of the different threats against machine learning-based systems depending on design choices. It will hopefully enable machine learning practitioners to make more informed decisions and better ponder security requirements during the design and deployment of machine learning-based systems. It will also empower security experts with a better knowledge of machine learning-specific security threats and the processes required to identify them. Based on this analysis, we will also

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

infer fine-grained security requirements to help better protect machine learning-based systems against cyberattacks.

1.2. STRUCTURE AND ORGANIZATION

This document defines the necessary technical background in Section 2, including useful security concepts and main machine learning system architectures for training and inference. We also describe the threat modeling process and give several examples of existing approaches to threat model conventional computer systems.

In Section 3, we discuss the limitations of conventional threat modeling approaches with regards to machine learning systems. We continue by presenting vulnerabilities that machine learning-systems expose, namely algorithmic, supply chain, and deployment vulnerabilities, and provide some examples of attacks that exploit them. Finally, we review frameworks that have been recently proposed by both the industry and public organizations to start addressing the security of machine learning-based systems.

Section 4 is dedicated to security analysis of machine learning-based systems based on phases in their lifecycle: training or inference, and on the deployment architecture they use: centralized or distributed. For each paradigm, we define the access to machine learning system assets by the different parties involved, we identify the vulnerabilities exposed by these systems and describe the effectiveness and likelihood of different attacks that can exploit them.

Finally, Section 5 concludes this document by identifying new security requirements for machine learning-based systems.

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

2. BACKGROUND

2.1. DEFINITION OF SECURITY CONCEPTS

To assess the security and model the threats against machine learning-based systems, it is paramount to understand the basic security concepts and terms. This section defines and highlights the relationship between the main security concepts, which are depicted in Figure 1.

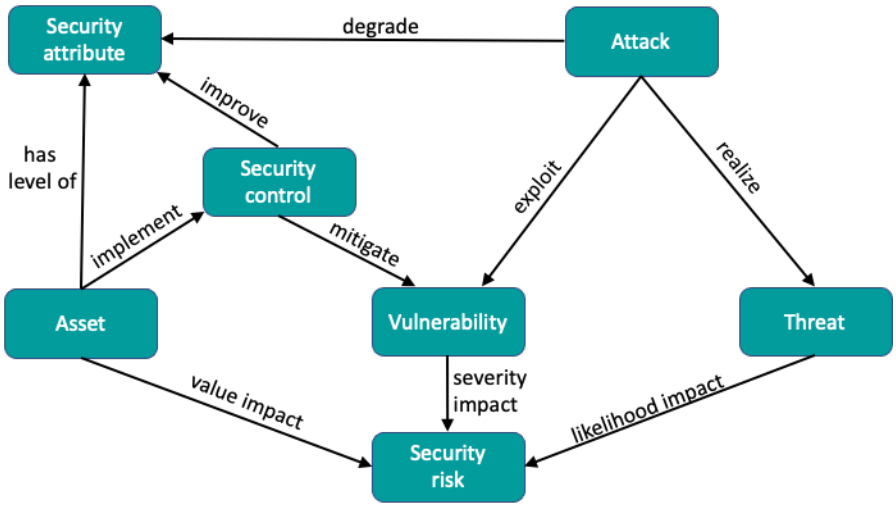

FIGURE 1: SECURITY CONCEPTS AND THEIR RELATIONSHIP TO ONE ANOTHER

2.1.1. SECURITY ATTRIBUTE

According to NIST Special Publication 800-53 [1] titled "Security and Privacy Controls for Information Systems and Organizations", a security attribute is "*an abstraction that represents the basic properties or characteristics of an entity with respect to safeguarding information. Typically associated with internal data structures—including records, buffers, and files within the system—and used to enable the implementation of access control and flow control policies; reflect special dissemination, handling or distribution instructions; or support other aspects of the information security policy.*"

Security attributes may be associated with either active or passive assets by means of attribute binding where the attribute type and value are also defined. The type and possible values of a security attribute depend on the context and type/values that are meaningful in one context may not be so in another context. By binding a security attribute to an asset, an organization can enforce information security policies like access control or information flow control. Popular attributes that find wide applicability in many scenarios are included in the CIA triad – Confidentiality, Integrity, and Availability. Confidentiality sets the limits of disclosure of an asset. Thus, a confidential asset is designed to be kept inaccessible/unreadable to all parties, aside from those authorized. Integrity refers to the state of an asset being unchanged or untampered from its original state. Finally, Availability refers to an asset being accessible when needed, despite system or service disruption. Nevertheless, these generic security attributes have to be defined and understood in the context of machine learning systems since they have a different meaning than for conventional systems.

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

2.1.2. VULNERABILITY

A vulnerability in IT can be defined as a "*weakness in the information system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat source*" [2]. In other words, it is a flaw or a security implication that can arise due to various design choices in an organization's asset associated with its information systems. An attacker taking advantage of such a security vulnerability could compromise the system's confidentiality, integrity, and accessibility. A few examples of known vulnerabilities are Heartbleed, Shellshock/Bash, and POODLE [3]. Tainted datasets and unrestricted access to model parameters on deployed machine learning models can be considered vulnerabilities in the domain of AI/machine learning security.

The Common Vulnerabilities and Exposures (CVE) is the de facto international standard for identifying vulnerabilities that feed into the US National Vulnerability Database [4]. CVE is an international, community-driven effort to identify, define, and catalogue publicly disclosed cybersecurity vulnerabilities, and is used globally to drive vulnerability awareness. CVE is an open data registry and vulnerabilities can be conveniently accessed via their assigned CVE IDs by any stakeholder who wishes to protect their systems against attacks.

Vulnerabilities can go undetected during the implementation/design phase for various reasons. In an informed environment, certain design decisions are taken based on the severity of the vulnerability, and reasonable amendments are made to mitigate the security risk through suitable security control methods. In this way, the security attributes of the organization's assets are preserved in the stakeholders' interests.

2.1.3. THREAT

A threat can be defined as "*any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, or individuals through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service. Also, the potential for a threat-source to successfully exploit a particular information system vulnerability*" [2]. An adversary exploiting system vulnerabilities can give rise to threats causing impairment on asset confidentiality, integrity, and/or availability.

A threat in a system can give rise to more threats causing a chain reaction effect. It is vital to identify a threat's origin (human or natural such as earthquakes, power failures, etc.) and source (intent and method) before applying suitable threat management and control techniques [5]. Security threats can be identified through threat modelling and analysis.

2.1.4. ATTACK

An attack is defined as "*an operation, whether in offence or defence, intended to alter, delete, corrupt, or deny access to a computer data or software for the purposes of propaganda or deception; and/or partly or totally disrupting the functions of the target computer, computer system or network, and related computer-operated physical infrastructure if any; and/or producing physical damage extrinsic to the computer, computer system or network*" [6]. An attacker may attempt to exploit a vulnerability of a system to materialize an actual attack. To achieve the objectives of an

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

attacker, they can follow characteristics of being harmonized, organized, launching in enormous scales, being regimented, scrupulously designed with demanding time and resources [7].

Examples of attacks include security attacks such as *Denial of Service (DoS)*, *eavesdropping*, *man-in-the-middle (MITM) attacks*; privacy attacks, including *Data Reconstruction Attack (DRA)*, *Property Inference Attack (PIA)*, and *Membership Inference Attack (MIA)*. In the domain of machine learning, two example categories of attacks are *poisoning attacks* and *evasion attacks*.

2.1.5. SECURITY RISK

In the former standard, security risk is defined as "*a measure of the extent to which an entity is threatened by a potential circumstance or event, and typically a function of: (i) the adverse impacts that would arise if the circumstance or event occurs; and (ii) the likelihood of occurrence*", according to NIST SP 800-30 "Guide for Conducting Risk Assessments" [8].

The common understanding of security risk is considered the triplet, including assets/values, threats, and vulnerabilities. The most generic version of three-factor perspectives is as follows: *Security risk = f(asset value, threat, vulnerability)*, where *f* denotes a function, for example a multiplication. The security risk assessment model [9] consists of four principal steps as follows:

- **Identification.** Identify all critical assets in the system and relevant sensitive data. For each critical asset, create a risk profile.
- **Assessment.** Assess identified security risks for critical assets, and analyze the correlation between assets, threats, vulnerabilities, and mitigating actions and then determine how to plan risk mitigation.
- **Mitigation**. Define a mitigation action and enforce security controls for each identified security risk.
- **Prevention**. Finally, implement and deploy tools to minimize threats and vulnerabilities.

2.1.6. ASSET

In general, an asset is defined as "*an item of value to stakeholders, that may be tangible (e.g., a physical item such as hardware, firmware, computing platform, network device, or other technology component) or intangible (e.g., humans, data, information, software, capability, function, service, trademark, copyright, patent, intellectual property, image, or reputation)*", according to NIST SP 800-160 Vol. 2 [10]. Furthermore, the value of an asset is determined by stakeholders based upon the value of the loss of this asset during the whole life cycle of the system.

In the cybersecurity domain, an asset is any data or device owned by an *organization*, that is involved in system activities, including hardware (e.g., servers and switches), software (e.g., critical applications) and confidential data. Therefore, security assets are prime targets for various *vulnerabilities* and *threats*. The principal goal of information *security controls* is to protect and ensure the *security attributes*, including confidentiality, integrity and availability (CIA), of assets from cybersecurity attacks, illegal access, use, disclosure, alteration, destruction, and theft. Security experts must assess the impact of each potential threat and then apply appropriate mitigation actions.

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

2.1.7. SECURITY CONTROL

Security controls are safeguards and countermeasures designed to protect the *security attributes* of an asset: confidentiality, integrity and availability. They are used to reduce the *security risk* related to the asset by mitigating its *vulnerabilities*. Security controls are implemented by or on the asset to improve its security. They can be the following:

- ***Corrective*** controls are meant to limit the impact of a security incident, e.g., by patching the existing vulnerability.
- ***Detective*** controls are meant to identify a security incident, e.g., by deploying a method to detect attacks that exploit the vulnerability.
- ***Preventive*** controls are meant to prevent a security incident from occurring, e.g., by blocking attacks that exploit the vulnerability.
- ***Deterrent*** controls are meant to dissuade attackers from causing a security incident, e.g., by defining sanctions for exploiting the vulnerability.

Security controls can be further classified according to the way they are implemented. They can be physical, procedural/administrative, technical or legal/regulatory. This report is primarily aimed at technical security and machine learning experts. Consequently, and for the sake of conciseness, we primarily focus on technical controls meant to address technical vulnerabilities in machine learning systems.

Security controls are often mapped to information security standards. Their implementation ensures that information systems comply with defined security standards in addition to improving the security of systems on which they are deployed. For example, the ISO/IEC 27001 standard for information security specifies 114 security controls to protect information systems [11]. The National Institute for Standards and Technology (NIST) cybersecurity framework also lists over 100 security controls in its core definition [12].

Security controls are linked to the mitigation specification of well-known vulnerabilities, such as the ones listed in the Common Vulnerabilities and Exposures (CVE) repository [4], to indicate which control(s) can mitigate each identified vulnerability.

2.2. MACHINE LEARNING SYSTEM ARCHITECTURE

The security threats and risks towards a system can be partially understood through its *attack surface*. An attack surface is the set of functionalities that can be interacted with, and that might allow an adversary to affect the functioning of the system. In a typical information system, the attack surface consists of its interfaces (machine-to-machine interfaces and user interfaces), and where an attacker could execute code, or intercept or modify stored or transferred data. To understand the attack surface of machine learning systems, we present their architecture, first on a general level and then based on different architectures that exist for training machine learning models and using them for inference.

2.2.1. MACHINE LEARNING SYSTEM OVERVIEW

2.2.1.1. Machine learning pipeline & machine learning model lifecycle

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

The attack surface of machine learning systems significantly resembles the attack surface of traditional software systems. Figure 2 shows a typical machine learning pipeline with the main assets that interact with the machine learning model at different stages of its lifecycle.

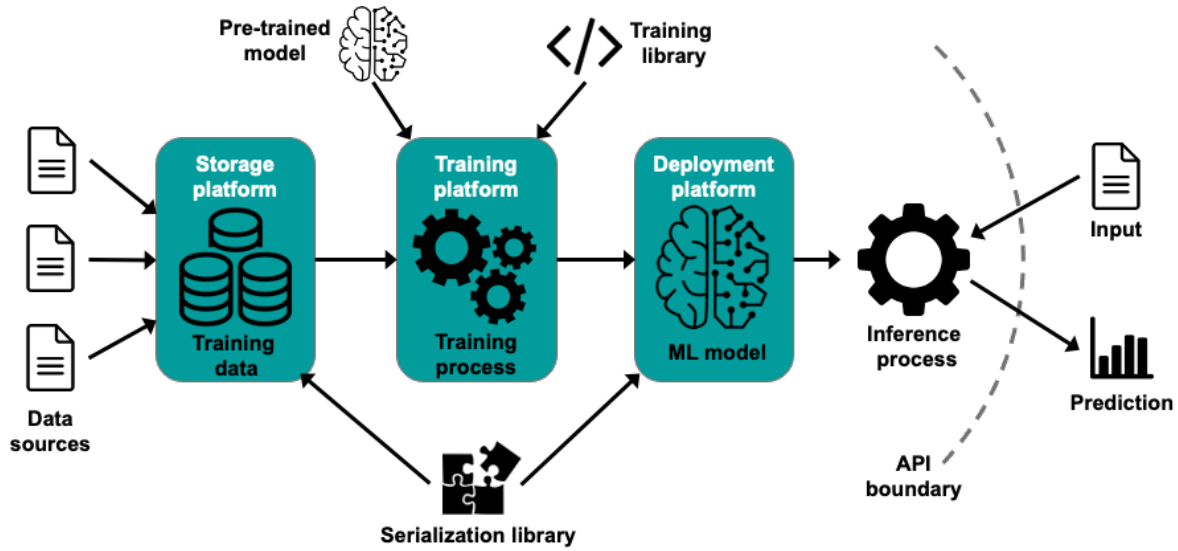

FIGURE 2: OVERVIEW OF A MACHINE LEARNING PIPELINE, FUNCTIONALITY AND DATA FLOWS

From right to left, the 'intelligent' predictions or decisions in the picture are performed by the machine learning model, which is the asset deployed as a part of the operational system. The machine learning model infers predictions based on inputs, usually through a programming interface (the API boundary in the picture).

Before a machine learning model is deployed, it is trained. The training process involves feeding the model various inputs and changing the model's parameters until it makes expected predictions. The model may be based upon an existing pre-trained model, thus reducing the training effort. The training process also relies on external libraries that implement common machine learning training algorithms and serialization algorithms to store and transfer both data and machine learning models. As a part of the training process, the model will also be validated, using a separate set of data reserved for that purpose. The machine learning model may also be re-trained, occasionally or continuously (online learning), and re-validated each time this happens.

Training and validation data are provided through data sources and a data storage system, on the left-hand side of the picture. Data sources can be numerous and of different types, including private customer data, public data from a repository, and commercial data.

The assets in this figure can be divided in three categories:

- Assets that are common to both conventional software systems and machine learning systems, and that can be protected using conventional security approaches. These include:
    - External libraries and tools, e.g., those used for training and data handling
    - Platforms, e.g., used for storage, training and deployment

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

- The query interface towards the machine learning model

- Assets that exist both in conventional software systems and machine learning-based systems, but have different characteristics in the case of machine learning systems (discussed in the following section):
  - Inputs to inference, which are analogous to inputs of a traditional, non-machine learning software
  - Output predictions, which are analogous to the outputs of traditional software
  - The machine learning model itself, which is analogous to a traditional software that processes inputs and produces an output
- Assets specific to machine learning systems:
  - Data sources
  - Training and validation data and associated processes
  - Pre-trained model(s)

2.2.1.2. Specificity of machine learning systems

In contrast to a traditional software program, the behavior of a machine learning model is learned from the data it is trained with. This has four main implications that, compared to conventional software systems, increase its attack surface, and introduce new vulnerability types.

First, as the model behavior is learned from training data, information contained in the training data is inherently embedded into the machine learning model and by transitivity into its predictions. This means that the machine learning model and its predictions could be used to compromise the confidentiality of the training data and its data sources, even if the training data is well protected using encryption and secure storage mechanisms.

Second, the fact that the model behavior is learned from data means that an attacker can compromise a machine learning model by compromising its training data or its data sources. Any compromise (e.g., loss of integrity) of these assets prior to training will be transferred into the model during training.

Third, it is hard to verify machine learning models. Unlike traditional software libraries, it is difficult to read the code of a machine learning model and identify potential flaws and threats (which can be further aggravated by the non-deterministic decision-making of some models). Furthermore, while input-output-based validation of machine learning models can produce statistical evidence of their expected functionality, validation results cannot prove their correctness for all the possible inputs. This naturally brings about supply chain attack risks since third-party machine learning models are often used either “as is” or as a foundation for training other models.

Fourth, it is difficult to reliably detect adversarial data inputs, both for training and for inference. The reason why machine learning is used in the first place is to cope with the fact that we cannot define explicit rules to model some data or a phenomenon in an explicit manner. Consequently, there are typically no easy and scalable means to decide whether some input data is benign or malicious. Although a generic input format can be defined for expected inputs (e.g., a picture could be defined through its bitmap size and color depth), this level of input

SPATIAL project is funded by the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 101021808.

syntax definition is generally too broad to detect adversarial inputs. This broad definition of input is often a requirement, since inputs are taken from the environment in which the machine learning system is deployed or from its users, and such input spaces cannot be narrowly defined. It also means that attackers can compromise machine learning systems by manipulating their environment or by controlling some of their users.

These four implications explain why machine learning systems expose additional and different vulnerabilities compared to traditional software systems. These new vulnerabilities also imply the exposure to additional and different threats that must be considered when assessing and implementing security in machine learning systems.

2.2.2. TRAINING ARCHITECTURES

Model training is the most time- and resource-consuming stage of a machine learning pipeline (see Figure 2). The performance, efficiency, and scalability of this process are of utmost importance. To improve training, three approaches and corresponding training architectures have been established in recent years. Each architecture enables different security threats and vulnerabilities.

2.2.2.1. Centralized training

In centralized training, the training process is centralized on a single server, where both the data and the machine learning model are stored in one place [13]. This training paradigm is depicted on the left side of Figure 3. Data is collected from one or several data sources. Nowadays, to be able to process the steadily increasing amount of available data [13] [14] and achieve scaling at industrial levels, the central server is typically a high-performance computer equipped with one or more Graphical Processing Units (GPU) that enable the large number of parallel arithmetic operations typically needed during training. Such a machine offers sufficient computational, storage, and memory resources to be capable of consuming all the data and iteratively optimizing the machine learning model during training. The advantage of this approach is that the full knowledge represented in the data is directly available during the model optimization process. A centralized training architecture does not require any synchronization processes.

The centralized approach comes with some drawbacks. First and foremost, a centralized training architecture does not scale well with larger amounts of data or model parameters. In recent years, a steadily increasing trend towards more and more available data has been observed [13] [14]. This elevated data availability enables the development of even more complex and higher-performing machine learning models with an exponentially increasing number of parameters [15]. However, central servers have reached their limits in terms of both storing and processing data and model parameters [13] [15]. To cope with an increased amount of data and model parameters while still maintaining a centralized training architecture, the central server would need to be scaled vertically by enhancing its computing and storage capacities or utilizing hardware that is specially designed and optimized for machine learning processes (e.g., TPUs, ASICs, or FPGAs) [15] [16]. However, this approach is very cost-intensive and limited by current technological developments as well as physical constraints (e.g., minimal transistor size and corresponding chip size) [14]. Another disadvantage is that the central server represents a single point of failure in the underlying system. If the operation of the server is disrupted, the whole training process may be compromised.

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

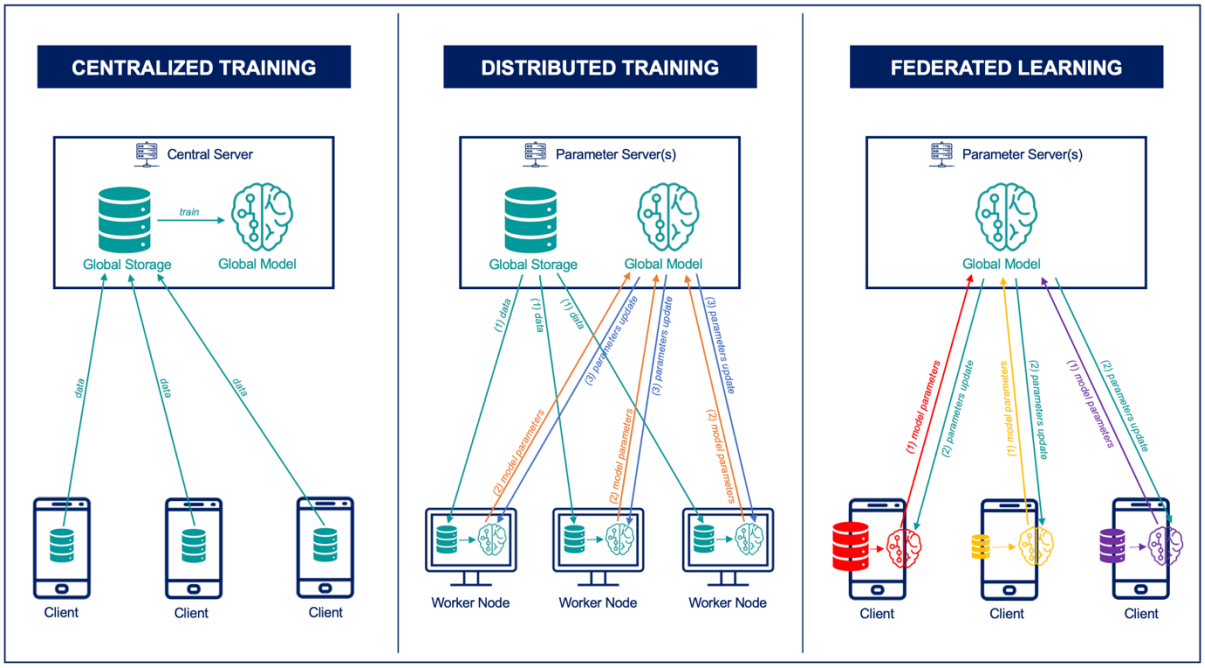

FIGURE 3: SIMPLIFIED AND SCHEMATIC ILLUSTRATION OF THE CENTRALIZED TRAINING (LEFT), DISTRIBUTED TRAINING (CENTER), AND FEDERATED LEARNING (RIGHT) ARCHITECTURES. THE DEPICTED DISTRIBUTED TRAINING ARCHITECTURE REPRESENTS A DATA PARALLELISM APPROACH.

2.2.2.2. Distributed training

Distributed training overcomes the scaling problems of a centralized training architecture [14] [15]. In this scenario, a horizontal scaling approach (also known as "scaling out") is adopted [15]. More precisely, the workload of the machine learning training process is distributed across several worker nodes, and the execution of the training process is parallelized. In the distributed training process, multiple worker nodes jointly develop a common machine learning model. This allows for a significant increase in efficiency, as individual resources can be better utilized [15], and thus, training time can be reduced. This approach is more cost-effective than vertical scaling for the training of larger models since training on expensive hardware is replaced by distributed training on low-cost hardware instances [15]. However, this methodology comes at the cost of increased network communication. Distributed training requires intensive communication and regular synchronization between worker nodes [14]. As a result, the network connectivity between worker nodes influences the efficiency of the entire training process and, therefore, represents a bottleneck in this architecture [14]. A network topology with high bandwidth and low latency between the worker nodes is essential. In general, two fundamental paradigms of distributed training exist - data parallelism and model parallelism.

Data Parallelism

A distributed training approach that follows the data parallelism method is depicted in Figure 3. In this paradigm, training data is partitioned into equally sized chunks and shared with worker nodes, which use their local data partitions to train local models in parallel. Local models are then aggregated into a global model. Data parallelism is typically used to accelerate the training process, or when training data is too large to be processed or stored on a single machine.

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

The most common data parallelism implementations use a central parameter server, depicted in the center of Figure 3. The parameter server is responsible for synchronizing and aggregating local models. After data is partitioned and distributed amongst worker nodes, the parameter server randomly initializes the parameters of a global model prior to the first training iteration. Next, the parameter server sends the initial model weights to all worker nodes. Worker nodes initialize their local models on these weights and then start to train on their local data partition. Once a local training epoch is completed, worker nodes send their updated local model parameters to the parameter server, which aggregates all interim results in the global model by averaging received parameters. The parameter server then sends the updated global model weights to each worker node. These steps are repeated until the parameter server measures a satisfying performance (e.g., accuracy) in the global model. Since the parameter server manifests a bottleneck and single point of failure in this approach, this role is typically replicated [14]. Multiple worker nodes typically act as parameter servers, which then coordinate the aggregation of the global model. This approach can be extended to a point in which no parameter server is required, and the worker nodes are directly connected in a peer-to-peer manner [15].

Model Parallelism

In distributed training, model parallelism is an alternative strategy to data parallelism. This approach is particularly advisable when a machine learning model is too complex, has too many parameters, and cannot be processed or stored on a single machine. In such scenarios, model parallelism is a way to distribute the complex model over several nodes. As the name suggests, model parallelism does not partition the data but the machine learning model itself. Individual components (i.e., layers in the case of deep neural networks) of the machine learning model are distributed to worker nodes, where they are incrementally improved and used for collective inference. In this scenario, all worker nodes have access to the entire training data set. Since individual parts of the machine learning model are distributed among the worker nodes, both training and inference require communication between all worker nodes. Two distinct approaches exist to implement model parallelism and partition the underlying machine learning model, namely vertical and horizontal partitioning [14]. In the former, the machine learning model is split between its layers, and individual layers are distributed to worker nodes. In contrast, in horizontal partitioning, the layers of the machine learning model are partitioned, and a worker node manages partitions of multiple layers. Both approaches are depicted in Figure 4 and discussed in more details in Section 2.2.3.3.

2.2.2.3. Federated learning

A special form of distributed training that implements an extreme form of data parallelism is called Federated Learning (FL). Federated learning aims to preserve privacy by not requiring clients to share local data with a central server. Instead, only updates from a locally trained machine learning model are shared [17] [18]. Therefore, federated learning is beneficial for applications where data privacy is essential. It can also be applied when the training data is already highly distributed. In such cases, federated learning avoids the communication overhead of sending the distributed data to a central repository. The high-level concept of federated learning is illustrated on the right side of Figure 3. This approach allows multiple clients to jointly train a common machine learning model without ever revealing their local data. Instead, the clients use their local data to iteratively train a local model, which is then sent to a central parameter server. After each training iteration, the parameter server aggregates the local client models into a global model using an algorithm called *FederatedAveraging* (FA)

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

[17]. McMahan et al. introduced this algorithm to deal with the problem of unbalanced and non-iid data present in the federated learning scenario [17]. In federated learning, the data is not equally distributed among clients. More precisely, each client can collect a different amount of data that could even follow different distributions in contrast to what global data aggregation would do (e.g., some labels can be overrepresented or underrepresented on some clients). Hence, a naive averaging of machine learning models independently trained on distinct non-iid subsets would result in an arbitrarily weak aggregated model [17]. Therefore, FA follows a different approach.

In the very first step of the FA algorithm, a global model is initialized with random weights and shared with all clients. The remaining steps of the FA algorithm can be described as follows [17] [18]. The parameter server randomly picks a subset of clients and shares the current global model with them. Next, those clients update their local replica of the model using their local data. Afterward, they share their updated model parameters with the parameter server in a synchronized fashion. The parameter server then aggregates the gathered model parameters by using a weighted sum of the individual parameter updates. This procedure is repeated until a satisfying global model is achieved.

There exist alternative peer-to-peer federated learning architectures which do not rely on a central parameter server for aggregation. Local models are exchanged between clients and aggregated by the clients themselves. While still protecting the data confidentiality, peer-to-peer federated learning has a large communication overhead since local models need to be sent more than once to each client participating. Also, the convergence to a single global is complex and requires the implementation of some consensus mechanisms, which further increases the communication and computation overhead for each client. To cope with these overheads, some hybrid federated learning architectures have also been developed.

2.2.3. INFERENCE ARCHITECTURES

Inference is the process where a trained machine learning model is used to produce output predictions based on inputs. Data inference is a lighter operation than model training but it can still cause constraints on resources such as time, bandwidth, and memory, based on the volume and type of data that needs to be processed. Further, inference depends upon how the model is stored in the network, as shown in Figure 4. We discuss below different ways of storing a model and performing inference based upon the scenario in question.

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

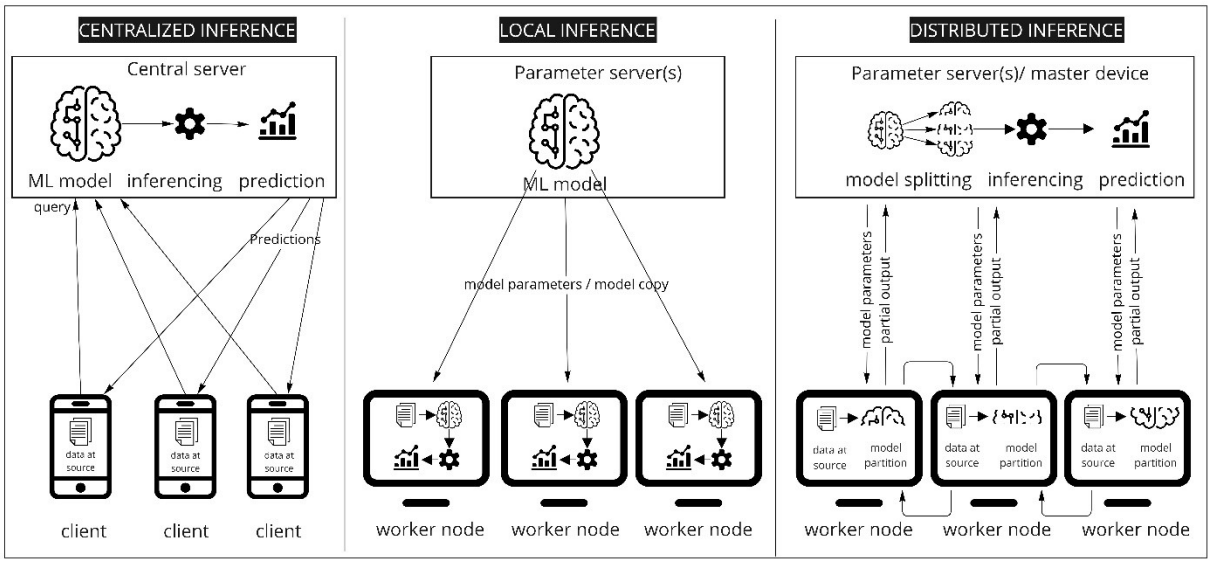

FIGURE 4: SCHEMATIC DIAGRAM FOR CENTRALIZED INFERENCE (LEFT), LOCAL INFERENCE (CENTER), AND DISTRIBUTED INFERENCE (RIGHT) ARCHITECTURES.

2.2.3.1. Centralized inference

Centralized inference is performed when the machine learning model is stored on a central server. This is the traditional way of storing machine learning models, where the model is large, and clients may not have enough storage or computational capacity to perform inference. Clients and servers engage in a client server protocol, where the client creates a query-based communication for the inference tasks, as shown on the left in Figure 4. Typically, this scenario exists in a cloud computing architecture, where most of the computation is performed on the server and the network bears only the communication costs.

Since each input is sent to the server for inference, this type of communication process may pose privacy risks for sensitive data and security risks while transmitting data over the network. If the processing power of the edge device can support data pre-processing, like feature extraction, these tasks are typically performed locally before sending data over the network, which can reduce the privacy exposure. This can also help to distribute the processing between client and server.

Central inference can also incur delays as it is dependent on communication with the server. Sending queries to a server and waiting for a response increases communication time and can also cause network congestion and packet loss for high volume data. Thus, this type of inference is problematic for time-sensitive applications like autonomous vehicles or healthcare applications. Further, communication is also dependent on the availability of the server, and systems such as these also suffer from a single point failure. Thus, the failure of the central server can cause the whole application to fail.

For data types with higher memory requirements (such as images and videos) and for high volume data, high bandwidth communication is required, and APIs for big data can be used to enable more efficient communication. Data from different sources can also be aggregated and sent for efficient use of bandwidth. Some applications that support this type of communication are REST APIs for inference querying, and open-source big data platforms like Hadoop and

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

Apache Spark. These libraries can also perform batch inference for high influx inference requests on a network.

2.2.3.2. Local inference

Local inference can be performed where the machine learning model is small enough to be stored on a local device. If there exists a global model in the network, a copy of such a model is stored at the edge, as illustrated in the center figure of Figure 4. Otherwise, a locally trained model can be used for inference. Local inference also enables special cases of federated learning where the global model can be retrained using local data and without further aggregation, to create a more personalized model. This type of inference can be used in applications needing more tailor-made results based upon one's preferences. Since inference data is locally accessed and does not need to be transmitted over the network, this method also preserves privacy. Thus, it is also suitable for applications that deal with sensitive content, such as personal data.

Since no data is transmitted over the network, local inference also has low communication costs and network congestion is low during inference. This method can be used for time-sensitive applications such as autonomous driving, where latency and transmission overhead need to be minimized. Thus, this method supports fast computation and can provide a better user experience. Nevertheless, the speed of inference also depends on the data type and hardware used. NVidia Jetson is an example of a high-performance piece of hardware that provides high speed edge computing solutions. It comes with an inbuilt GPU for parallel computing and high-speed interface to support larger volumes of incoming data [19]. In addition, plug-in devices like Intel Neural Compute Sticks can also be used to perform local inference on resource constrained edge devices. They do not come with GPUs but do contain specialized hardware acceleration units designed for deep neural network inference [20].

2.2.3.3. Distributed inference

Distributed inference is performed when a machine learning model is too big to fit on a single entity such as an edge device. This process is often used when devices do not contain enough memory to run the full model. In such cases, a machine learning model is split into parts and distributed over several devices (worker nodes) across a network, as shown on the right in Figure 4. Each device runs part of the machine learning model and participates in each inference step.

Deep neural networks are a type of machine learning model that may need to be split such that they can be run on edge devices. Some deep neural networks (such as those used for visual classification and segmentation) contain multiple layers of convolution and pooling in their architecture. These types of deep neural network can be distributed in one of two ways. *Layer-wise splitting* assigns each layer to an individual device as shown in Figure 5. In cases where even a single layer is too computationally expensive to fit on a device, individual layers are *split across convolutional units* as depicted in Figure 6.

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

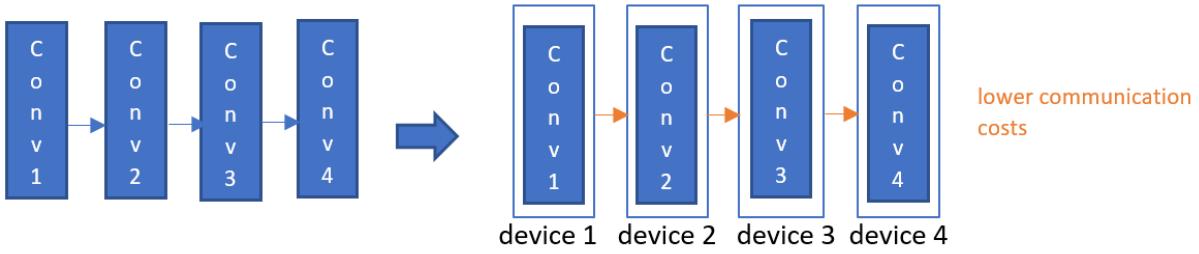

FIGURE 5: LAYER-WISE SPLITTING OF THE NEURAL NETWORK

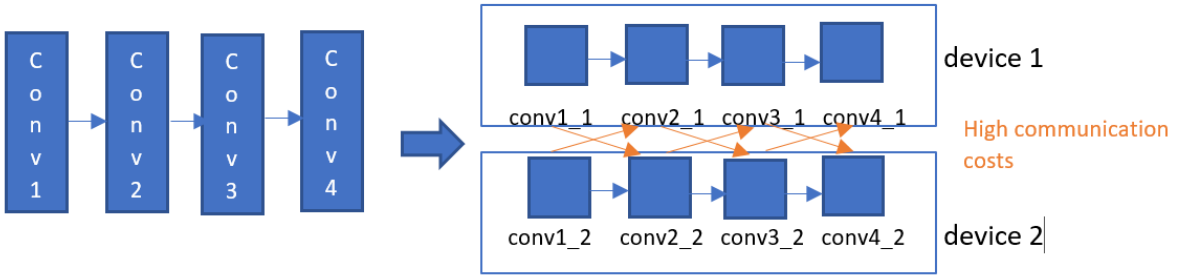

FIGURE 6: SPLITTING ACROSS THE LAYERS OF THE NEURAL NETWORK MODEL

Layer-wise splitting

In layer-wise splitting one or more layers are distributed across different devices of the network based upon their computational capabilities. As shown in Figure 5, layer-wise computation happens in a sequential manner where each device with lower layer (early stages of the network) communicates with the next layer. Each device runs an independent task that can be computed individually. Results are passed to the next device in the sequence, and so on. One drawback to this approach is that the devices later in the sequence must wait for the previous devices to finish their tasks. Task division is based on cluster properties such as the number of workers and their computational capacities. Neural networks of the type discussed usually contain higher dimensionality layers at the start, designed to learn the fine-grained information. Lower layers reduce in size due to high dimensional convolutions. When a model such as this is split layer-wise, computational requirements gradually decrease further down the layers. To handle layers with computational constraints not suited to edge devices, high-dimension layers can be offloaded to the cloud [21].

Across-layer splitting

In this type of splitting, some, or all the layers of the convolutional network are split into a predefined size based on the computational capacity of the edge devices in the network. Since, in neural networks, the low layers are computationally expensive, their computation is divided across multiple devices, as depicted in Figure 6. One drawback of this method is that the devices cannot complete their tasks independently. They need to communicate with multiple devices during the execution of their task as the computation of each layer is dependent on the outputs of the previous layers from different devices. Thus, there is a need for communication with different devices after the execution of each layer. This increases communications overhead on the network. Another drawback of this method is related to the difficulty of providing

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

explanation for decisions that are computed for part on different devices. A global picture of the decision process is usually required to provide explainability.

2.3. THREAT MODELING

2.3.1. RATIONALE AND GOAL

Threat modeling is a process by which an asset and its environment are examined through the lens of security. It is a theoretical exercise that aims at improving the security of an asset or a system by identifying and understanding potential threats against it. This process allows countermeasures that can prevent or mitigate the effects of these threats to be defined. Threat modelling aims to capture, structure, and analyze all the information that affects the security of a system. It enables informed and rational decision-making about the security efforts required when designing and deploying a system, considering the security risk associated with it. It also documents the security posture of a system, which can be used as assurance in the situation where a security incident happens.

The threat model resulting from the exercise typically includes the following elements:

- A description of the system to be modeled, its functions and components.
- A list of the system's interactions and parties who can interact with it.
- Assumptions about how the system works, which can be validated or challenged.
- A list of potential threats to the system.
- A list of vulnerabilities that can be exploited by attacks that realize defined threats.
- Security controls that can be used to mitigate threats and attacks.
- An approach to validate the model, the threats, and the effectiveness of defined controls.

In addition to producing a "model of threats", the process also identifies potential vulnerabilities and a prioritized list of security improvements to the design, implementation, and deployment of the analyzed system.

Threat modeling can be applied to a wide range of systems and serves as an integral part of the development lifecycle of any system, from the design phase, through the development phase, and onward after the system is deployed and running. Systems get updated, their environments evolve, and threats against them also evolve, causing the need to reassess their security at different stages during their lifecycle. The process remains the same, but the information becomes more granular as the definition of the system is refined, starting from a conceptual and high-level threat model, and refining it throughout the system's lifecycle. The threat model must be at least updated when security incidents occur, new features are released, and when architectural changes are made.

Threat modeling can bring many benefits that go beyond traditional security. It enables the identification of security requirements, it can be used to predict new forms of attack, and it can be used to think about threats beyond conventional attacks that relate to the specific application and workings of the system in question. It is a tool of anticipation, to detect

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

problems before starting implementation, to spot design flaws that testing, and code review would overlook, and to remediate problems before they have a negative impact.

Threat modeling is an integral part of security risk assessment. In contrast to risk assessment, threat modeling puts the asset at the center of the exercise, brining focus onto what an organization deems most important. Each threat model is tailored to a particular system and is very application specific. The threat modelling process often ignores cost in favor of focusing on determining the impact of a security attribute being compromised should a threat realize.

2.3.2. EXISTING APPROACHES

2.3.2.1. STRIDE

The STRIDE framework is a threat modelling and security requirements gathering approach proposed in 1999 by Microsoft [22]. This method was developed in the context of Microsoft's Security Task Force and was recommended for use to secure all Microsoft products [22]. Today, STRIDE continues to be a standard threat modelling approach used by Microsoft and is even integrated into the *Microsoft Threat Modelling Tool* [23]. The STRIDE method structurally identifies threats, vulnerabilities, and security requirements during the design phase of an IT system. STRIDE's goal is to determine and collect a list of threats and vulnerabilities as soon as possible by systematically analyzing the attack surface of a system and identifying critical assets and services. Subsequently, developers and system designers can immediately react to the identified threats and derived security requirements by adapting the system's design. Thus, potential threats can be eliminated, or sophisticated mitigation approaches implemented. STRIDE divides threats and security requirements into six different categories. These can be derived from its name since STRIDE is a mnemonic that stands for [22]:

- **Spoofing identity.** In this threat category, attackers pretend to be a known entity attempting to gain access to a system. For example, attackers could obtain the credentials of legitimate users and use this authentication information to access a system. Thus, identity spoofing violates the authentication of the underlying IT system.
- **Tampering with data.** Here, threats describe the malicious modification of (sensitive) data without authorization. The attacker's goal is to stay undetected as long as possible. As a result, this threat category affects the integrity of the system.
- **Repudiation.** This category describes threats in which an attacker or malicious user cannot be traced for executing a (potentially dangerous) operation, or the execution can be denied. As the name implies, this violates the non-repudiation security property of an IT system.
- **Information disclosure.** This class of threats describes the publication or access of (sensitive) information to unauthorized entities. Any disclosure of sensitive information would violate the confidentiality of the system.
- **Denial of service.** This group describes threats that attempt to exhaust the resources (e.g., computational or storage resources) of a service so that it is temporarily unavailable. Such attacks damage the availability of the underlying IT system.
- **Elevation of privilege.** In this final category, threats that allow unprivileged users to gain privileged access to system components or operations are gathered. By exploiting this unauthorized access, malicious users can compromise and significantly damage the

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

underlying system. Therefore, the elevation of privileges violates the authorization mechanisms of a system.

2.3.2.2. P.A.S.T.A.

PASTA is a popular threat modeling framework introduced by Uceda-Vélez and Morana in 2015 and which stands for Process for Attack Simulation and Threat Analysis [24]. The PASTA framework abstracts the system under examination in seven steps to be carried out as the system is being designed. This methodology can also be applied to existing systems. The main goal of the PASTA framework is to provide a holistic view of the threat landscape by considering both *technical requirements* and *business objectives*. Different from alternative threat modeling frameworks, PASTA considers business context to determine the impact that threats may have on business. PASTA allows analysts to simulate attacks and thus determine potential threats and devise possible countermeasures. As PASTA mainly focuses on attacks, it is defined as a "risk-centric" framework and provides as its output a list of threat-score pairs that may help practitioners evaluate the vulnerabilities of their systems. The seven stages of PASTA are as follows:

- Identify business objectives as well as security and compliance requirements
- Capture the system boundaries and the dependencies between hardware and software components
- Decompose the application into elementary units that can be further analyzed for threat modeling
- Enumerate threats and attacks by prioritizing the most probable ones
- Map vulnerabilities in the system to threats identified in the previous steps. Vulnerabilities found during this step should be enumerated and scored with standard tools.
- An attack modeling stage then uses formal methods for attack analysis that identify how attacks can leverage system vulnerabilities.
- The last stage is used to quantify business impact and identify risk mitigation strategies.

2.3.2.3. Trike

Trike [25] is an open-source threat modeling methodology which focuses on a requirements model designed to ensure that the level of risk assigned to each system's asset is acceptable by its stakeholders. The Trike threat modelling methodology consists of two models, namely Requirement Model and Implementation Model [26].

- **Requirement Model** is the foundation of Trike, enabling coordination among stakeholders and different teams to identify security characteristics of a system – its intended actions, its assets, and who interacts with the system. It defines which existing rules apply to intended actions, and then builds an actor-asset-action matrix to represent all the data in a convenient tabular format.
- **Implementation Model** aims to provide a Data Flow Diagram (DFD) to illustrate both the data flow and user performed actions within a system. It first identifies unintended

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

actions in the system and how they interact with the system's states. It then maps those actions and states of the system into the DFD, enabling a clearer understanding of the implementation of the system's components.

Trike aims to build a risk model from the completed threat model based on all collected data from previous steps, for example assets, roles, actions and threat exposure. It is used to first create attack graphs representing all possible attacks that can compromise the system. The Trike model assesses the threat risk values using a five-point probability scale for CRUD (a.k.a., "create", "read", "update", and "delete") actions and evaluates actors based on their permission level for each action (e.g., "always", "sometimes", and "never"). Based on the calculated threat risk values (e.g., by "*simply multiplying threat exposure by the largest applicable vulnerability risk*" [2]), it then defines security controls or mitigating actions to address prioritized threats and assigned risks.

However, as the Trike model requires some human intervention to perform an attack surface analysis of a system before threat assessment and assignment of acceptable risk scores, it is challenging to apply in large and complex systems. Also, the Trike model is not scalable in nature. Moreover, the usage of the Trike model is limited because its versions 1.5 and 2.0 have not been well documented.

2.3.2.4. DREAD

The DREAD model [27], which has been proposed by Microsoft, "quantitatively assesses the severity of a cyberthreat using a scaled rating system that assigns numerical values to risk categories". The DREAD model allows security teams to better understand security risks and then to quickly reduce risks to an acceptable level. The DREAD model consists of five components to calculate the risk rating for a given threat by asking the following specific questions.

- **Damage** (**D**): How bad is the problem? This component enables understanding of the potential damage a particular threat is capable of causing.
- **Reproducibility** (**R**): Does the attack work reliably? This component identifies how reliable the attack is and how easy it is to replicate.
- **Exploitability** (**E**): How much work is it to launch the attack? The main goal of this component is to analyze the system's vulnerabilities and to discover how much expertise and effort an attacker must put in to launch the attack on the system.
- **Affected Users** (**A**): How many users would be affected by the attack, for example all the users or just some of them?
- **Discoverability** (**D**): Is it easy to detect the attack? This component aims at determining how easy it is to identify and detect potential cyberattacks in the system infrastructure.

The DREAD model is solid because of the independence between its five factors. The overall threat rating is calculated by summing the scores obtained across these five factors. Inputs are weighted between 0 and 10. Depending on the overall threat rating, the risk severity categories for a threat are assigned to critical, high, medium, and low. Overall, the DREAD model enables calculation, comparison, and prioritization based on the severity of discovered threats. Furthermore, the DREAD model is customizable and adaptable to almost any situation.

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

However, the major disadvantage of using the DREAD model is that extensive cybersecurity expertise and up-to-date domain knowledge about potential attack vectors and vulnerabilities is required to ensure that risk analysis is complete and accurate.

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

3. THREAT MODELING MACHINE LEARNING SYSTEMS

3.1. LIMITATIONS OF EXISTING THREAT MODELS

As discussed in Section 2.3.2, many threat modelling approaches such as STRIDE, DREAD, and Trike were developed to analyze and improve the security of information and software systems. Given these approaches are generic enough to be applied to literally any system, it follows that they should be suitable for analyzing the security of machine learning systems. The initial steps of threat modeling relate to developing a system description – its functions, components, parties interacting with it, and assumptions about how it works. Such a description is valid and applies to threat modelling in machine learning systems. Similarly, PASTA's description of the system environment and its relationship to business objectives, can also be applied to machine learning systems since the description is independent of technical specifications and only relates to functionality. The initial steps of a threat modelling exercise also enable the identification of sensible threats against machine learning systems since they are similar to conventional information systems. Such threats relate to the compromise of security attributes (confidentiality, integrity, availability) of the system's assets.

Even though existing threat modelling processes can be used for the initial threat analysis of systems containing machine learning functionality, they must be executed by experts who understand business objectives, asset values, and who have a technical understanding of both cyber security and machine learning. While many security experts are technically savvy in the realm of conventional information systems, they often have little technical understanding of machine learning systems and algorithms. This limits their ability to perform a thorough analysis and identify relevant threats.

The problem with applying existing threat modeling approaches to machine learning systems increases further when attempting to identify model-specific attack vectors, exploitable vulnerabilities, and relevant security controls. These steps require expertise and up-to-date domain knowledge about potential attack vectors and vulnerabilities related to the system's assets. The requirement for technical domain expertise is a common limitation of virtually all threat models. However, for conventional systems that have been used and studied for a long time, this limitation is partly addressed by decades of historical documentation based upon observations and analyses of real world cyberattacks. As a result, information about known vulnerabilities, attack vectors, and security controls is often baked into threat modelling frameworks for conventional systems, such as STRIDE.

Even though many vulnerabilities and proof-of-concept attacks have been demonstrated against machine learning models, not a lot of documented security knowledge exists in this field. Many new attack methodologies and vulnerabilities are, indeed, still being uncovered. Beyond the early conventional steps of threat modeling, additional security analysis is necessary to mitigate machine learning-specific threats and to improve the security of the data, models, and training/inference processes themselves.

The limited integration of technical and security information regarding the type of assets that machine learning systems are composed of in existing threat modeling approaches and frameworks holds these frameworks back with regards to effectively threat modeling machine learning-based systems. Vulnerabilities and attack vectors against these assets cannot be

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

effectively identified, causing potential attack vectors to be missed. For instance, if a security analyst does not know that information contained in training data will become embedded into a resulting machine learning model and in its predictions (discussed in Section 2.2.1.2), they won't be able to identify threats related to attacks designed to expose confidential data. This lack of information can lead to vulnerabilities being missed and an underestimation of the impact of successful attacks against the system. Altogether, this lack of domain knowledge can lead to underestimation of security risks against the system and an inability to decrease its resilience against attacks.

Security threats against machine learning systems can be quite different to threats against other systems. Threat modelling approaches designed for systems that utilize machine learning need to be specifically designed to address vulnerabilities and attack vectors against assets that those systems are comprised of. The link between threats, vulnerabilities, attack vectors and security controls, need to be linked if security risk are to be properly mitigated. In the remaining sections of this chapter, we address these issues by identifying vulnerabilities specific to machine learning systems and reviewing initiatives that have emerged to improve the threat modeling of machine learning systems.

3.2. VULNERABILITIES OF MACHINE LEARNING SYSTEMS

In this section, we present vulnerabilities specific to machine learning systems and classify them into three categories – algorithmic, supply chain, and platform. Table 1 further summarizes these vulnerabilities together with associated security attributes that can lead to compromise.

The CIA (confidentiality, integrity, and availability) approach works in a slightly different way in the context of machine learning models. Confidentiality is not limited to the prevention of access to a machine learning model but also to ensuring that its output predictions do not leak information that can be used to understand and reproduce its decision making or reconstruct its training data. Integrity relates to preserving expected behavior, level of performance, and quality of predictions under any conditions, including attack. Availability refers to the idea that accurate predictions are produced, that reflect those seen in testing, and in a timely manner.

TABLE 1: OVERVIEW OF VULNERABILITIES AGAINST MACHINE LEARNING SYSTEMS, THE MACHINE LEARNING LIFECYCLE PHASE THEY TARGET (TRAINING OR INFERENCE), AND THE SECURITY PROPERTY THEY COMPROMISE (CONFIDENTIALITY, INTEGRITY OR AVAILABILITY).

Vulnerability	ML lifecycle phase	Threatened assets & security properties		
		Training data	ML model	Predictions
Model poisoning	Training	(Integrity)	Integrity	Integrity Availability
Model evasion	Inference			Integrity
Model stealing	Inference		Confidentiality	
Training data inference	Inference	Confidentiality		
Compromised machine learning library	Training + Inference	Confidentiality	Integrity	Integrity Availability
Compromised pre-trained model	Training		Integrity	Integrity Availability
Compromised serialization library	Training + Inference	Integrity Confidentiality	Integrity	Integrity

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

Compromised training platform	Training	Confidentiality Integrity Availability	Confidentiality Integrity Availability	
Compromised deployment platform	Inference		Confidentiality Integrity Availability	Integrity Availability

3.2.1. ALGORITHMIC VULNERABILITIES

Machine learning models expose new algorithmic vulnerabilities that are not present in conventional systems. They are outlined in the following sections.

3.2.1.1. Model poisoning

Model poisoning is a type of attack designed to alter a machine learning model via influence over its training data or its training process [28]. Model poisoning attacks compromise the integrity of a machine learning model. In a data poisoning attack, an adversary injects malicious data inputs to the model's training set designed to distort the model's ability to accurately classify inputs. In this way, an attacker alters the accuracy of the machine learning model for their own purposes. A model attacked in this fashion may become impractical for real-world use. As such, data poisoning attacks can also impact the availability of a machine learning system.

Multiple approaches exist to launching data poisoning attacks [28]. For instance, a *label modification* technique can be used to modify the labels of existing data in supervised learning datasets. Such an attack may be launched when an adversary cannot inject inputs directly into the existing training data. Alternatively, attackers can inject their own inputs into the training set using a *data injection attack*. Finally, in a *data modification attack*, the adversary alters the existing training data rather than adding new samples into the training set.

An alternative possibility for poisoning is backdoor attacks. It is a form of adversarial attack where the attacker poisons a small part of the training data and creates trigger patterns in the model that can be activated during inference [29]. The model may perform well on most inputs, but its accuracy may drop only for specific inputs with backdoor triggers, such as inputs that satisfy some secret or inputs having certain property chosen by the attacker. These poisoning attacks will therefore affect the integrity of the machine learning model.

In the federated learning paradigm, model poisoning attacks take a different form. Rather than poisoning their local data, clients will directly tamper with their local machine learning model. Their goal is to generate a local machine learning model which, when aggregated with other benign local machine learning models, will result in a compromised global model with low accuracy.

When considering reinforcement learning, model poisoning attacks take yet another form since the data is directly taken unlabeled from the environment. One approach to poisoning in reinforcement learning is similar to label modification. However, given that labels are computed using the reward function in reinforcement learning, poisoning relies on compromising the reward function used to train the machine learning model such that an incorrect reward is computed for valid inputs.

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

3.2.1.2. Model evasion

Model evasion attacks [30] are the most common attacks on machine learning systems. Evasion attacks target the inference phase of the machine learning model lifecycle and compromise the integrity of the machine learning model's predictions. Such attacks aims to change the expected (and often correct) output of the machine learning model using well-crafted malicious inputs, a.k.a. *adversarial examples*, to confuse machine learning models into making incorrect predictions. Evasion attacks typically aim to obtain a misclassification while making minimal modifications to the sample to be misclassified. For example, an attacker can implement an evasion attack to bypass a network intrusion detection system (NIDS) by minimally modifying malicious network packets while preserving their malicious utility and remaining undetected by the NIDS.

From an attacker's point of view, the more information about the target AI-based system is available, the more likely an attacker can successfully trick the model. Evasion attacks do not require any access to the model's training datasets but do require some level of knowledge of the target model. According to the threat model, existing adversarial attacks can be classified into three categories: white-box, grey-box, and black-box attacks, with differences being in the knowledge and capabilities of the attacker [31]. While in white-box attacks, attackers have complete knowledge of the target model, including model architecture and parameters, the knowledge of attackers in black-box attacks is limited to only querying the target model to obtain complete or partial information, thus making the generation of adversarial examples more difficult. In grey-box attacks, adversaries are assumed to have limited knowledge of the structure of the target model. Some evasion attacks against well-known AI models are proposed in the literature, for example kernel-based classifiers [32] and deep neural networks [33].

3.2.1.3. Model stealing

Model stealing attacks, also called *model extraction*, are adversarial machine learning attacks that compromise the confidentiality and the intellectual property of a machine learning model during inference. They can be used to steal the machine learning model, and as a steppingstone to launch other attacks, e.g., white-box evasion attacks. The confidentiality of machine learning models deployed behind a prediction API are theoretically protected because they only provide query/response interactions to their clients, and do not reveal the internal decision process of the model. This deployment protects the intellectual property, competitive advantage, and business value of the machine learning model for its owner who has invested time, money and expertise to train it.

However, machine learning models leak information about their internal decision logic through the query/response interactions provided during inference. By carefully crafting adversarial queries to the machine learning model (via an API), an attacker can exploit the information contained in returned predictions to reconstruct a surrogate machine learning model with similar performance and a similar behavior as the victim model. Thus, the target model's predictions serve to leak information and compromise its confidentiality. Model extraction attacks are typically iterative. Adversaries make several queries and use them to train a surrogate model. This surrogate is then used to find new queries that will improve its performance and the process repeats until a satisfying copy of the victim model is obtained. Several such model stealing attacks have been demonstrated and are able to reconstruct the surrogate of complex machine learning models using hundreds of thousands to millions of queries [34] [35] [36]. For instance, the popular natural language processing, BERT transformer

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

model, can be stolen for somewhere between €100 and €1000 [37]. Also, a credit risk prediction model can be stolen using the German Credit Card dataset within 10 minutes and using only 1150 legitimate queries [38].

The effectiveness of a model stealing attack depends on the knowledge the attacker has about the target model. Limiting the information available about the model architecture, its training algorithm, its optimization method, or its training data limits its vulnerability against a model stealing attack. Increasing the granularity of predictions, e.g., label vs probability, and reducing details communicated to the client, limits information leakage and mitigates the model's vulnerability to such attacks.

3.2.1.4. Training data inference

Data inference attacks take advantage of the information leaked by machine learning systems and use this information to compromise the confidentiality of training data and threaten the privacy of individuals or organizations whose data was used in training sets. Two main types of inference attacks exist – membership inference, and model inversion (a.k.a. attribute inference attack).

Membership inference attacks assume a situation where access to a model is readily available (as was explained in our definition of black-box attacks). Such an attack attempts to identify if a record is included in the training data. There are many cases where this attack can have a serious impact, such as when the attack attempts to uncover sensitive personal data such as purchase records, location, or medical records. The basic idea of a membership inference attack is to learn the difference between the target model's behavior with inputs that were already seen in training data set and from inputs from unseen data. A representative methodology, shadow training, can be found in [39]. More vulnerable target models are those that are overfitted and trained with data with less diversity since this difference will be clearer.

Model inversion attacks assume a situation where an attacker already has partial knowledge about a data record and tries to infer the information of the missing attributes. The attack methodology is similar to that of membership inference attacks. In this case, the attacker repeatedly queries the target model with different possible values of a missing attribute and analyzes outputs to discover the value that is indeed in the corresponding record of the training data set [40]. Again, overfitted models are more vulnerable to this attack since their behavior for the records in the training data and other general records varies more. A study of a practical example of this attack can be found in [41], which is able to extract personal attributes of online social media users from publicly available attributes.

3.2.2. SUPPLY CHAIN VULNERABILITIES

External components, libraries and pieces of code used in machine learning systems can be potentially compromised and used to attack the resulting model or system. The provenance and integrity of these external components must be verified to prevent supply chain attacks. Besides conventional supply chain vulnerabilities related to compromised third party software, machine learning systems leverage specific third-party components like machine learning training libraries, pre-trained machine learning models and serialization libraries, which can be compromised by supply chain attacks. Verifying the integrity of these components raises new

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

challenges compared to verifying the integrity of conventional software and it creates new threats against machine learning systems.

3.2.2.1. Compromised machine learning libraries

The most common machine learning supply chain attack relates to compromising machine learning training and inference libraries. Most machine learning libraries, e.g., TensorFlow, PyTorch, ScikitLearn, etc. are maintained by open communities and are susceptible to potentially malicious actors. The detection of such a compromise is challenging since it is difficult to validate the output of a training algorithm. Most machine learning training algorithms are stochastic, the machine learning model they train depends upon some randomness, and we cannot formalize or verify what is a valid non-compromised machine learning model.

The compromise of machine learning libraries often consists of augmenting the training algorithm with extra code, which adds extra functionality to the trained machine learning model. One example is the addition of backdoors in the machine learning model during training in the same way as for backdoor poisoning attacks, but without the need to compromise the training data [42]. Another example of library compromise aims to encode part of the training dataset into the trained machine learning model, such that this sensitive data can be later leaked when the model is made publicly available for inference [43]. This training data can be extracted and reconstructed either through direct access to the model or simply by querying it during inference.

3.2.2.2. Compromised pre-trained models

Due to the monetary cost, artificial intelligence expertise, and large data volume required to train high-performance machine learning models, many AI practitioners resort to using pre-trained machine learning models that they repurpose for different tasks using transfer learning. Transfer learning consists of retraining a powerful pre-trained machine learning model for a different task while preserving and benefiting from the knowledge already embedded in it. Transfer learning is a very popular practice because it reduces costs and the need for a large volume of data to train a baseline high-performance machine learning model. The GPT-3 language model or the Inception image model are examples of pre-trained models commonly used as basis for transfer learning.

However, transfer learning comes with security threats since it leads to using untrusted machine learning models trained by external parties and which can be potentially compromised. As we already pointed out, it is very difficult to verify whether a machine learning model is compromised, since it is not possible to specify its valid behavior for all possible inputs. For instance, attacks have been demonstrated where pre-trained machine learning models are poisoned with backdoors that transfer to any machine learning model re-trained from this compromised base model [44].

3.2.2.3. Compromised serialization libraries

Serialization libraries are commonly used to package training data and machine learning models. These libraries typically come from external parties and are used not just for machine learning applications, but to package a variety of different types of data. Intentionally malicious modifications could be made to such libraries such that serialized data and machine learning models will be compromised after being unpacked. It has been demonstrated that serialized datasets can be modified to inject poisoning or backdoor attacks during the deserialization

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

process [45]. Similarly, machine learning model integrity and accuracy can be compromised by performing random weight modifications during deserialization.

3.2.3. DEPLOYMENT VULNERABILITIES

External cloud services are commonly used to control the cost of training and deploying machine learning models. Some cloud services even provide customed solutions for machine learning specific tasks, e.g., AWS Sagemaker or Microsoft Azure machine learning, which provide standard machine learning libraries and monitoring solutions as part of their offering. The compromise of training or deployment platforms can jeopardize the security of machine learning systems despite having all their components and algorithms secured. Secure and trusted machine learning processes and components can be replaced by malicious ones at will by the platform supposed to execute them.

3.2.3.1. Compromised training platform

In the context of system security, if the platform where training occurs is compromised, any training-focused adversarial attack can be executed since the adversary would have full access to both the model and its training data. In this scenario, the integrity of the machine learning model can be compromised, its performance can be degraded, and it can be backdoored. The adversary may augment the machine learning model with additional functionalities designed to leak information about the training data. A compromised training platform thus presents a significant threat against the confidentiality and integrity of both a machine learning model and its data.

An adversary who compromises a training platform can also reveal confidential information about both training data and the target model. This threat is more subtle since it would be very difficult to notice, especially if the adversary is honest but curious. The adversary could easily copy and distribute the trained machine learning model – this represents the same consequences as a model stealing attack: compromising both business advantage and intellectual property.

3.2.3.2. Compromised deployment platform

The machine learning model's deployment platform is the system where the model is hosted for inference. It this platform is compromised, the integrity of the model itself and of its predictions are also compromised. In this case, an adversary can choose to alter the system to return predictions that differ from those expected by the deployed model. The adversary can also replace the inference model with their own. This scenario can also be considered an evasion attack. By disabling the system, the adversary can deny the availability of the service to its clients. If an adversary gains access to a machine learning model's deployment platform, the confidentiality of the machine learning model will also be compromised since it can be copied to another system and potentially leaked to others. An adversary compromising a deployment platform will be able to easily perform data inference attacks since they will have unrestricted white-box access to the model.

3.3. FRAMEWORKS FOR MACHINE LEARNING SECURITY

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

To address security threats against machine learning systems, several public organizations and companies have started initiatives to systematize the knowledge and approaches for securing these systems. We present some of the major initiatives in this section.

3.3.1. NIST ADVERSARIAL MACHINE LEARNING TAXONOMY

The National Cybersecurity Center of Excellence (NCCoE) of National Institute of Standards and Technology (NIST) developed a taxonomy and terminology [46] specific to Adversarial Machine Learning, which inclusively refers to possible adversarial manipulations of machine learning systems.

The taxonomy and terminology broadly cover adversarial attacks against machine learning systems, defenses, and their consequences. The document hierarchically organizes the taxonomy for each of the three following dimensions: Attacks, Defenses, and Consequences. We show an example subset of the taxonomy hierarchy in Figure 7. The taxonomy is built based on recent surveys (for example, [47] and [28]) of the literature on adversarial machine learning, and it identifies common aspects among them.

This taxonomy organizes the domain upon key dimensions and provides a structural view of the space. For example, attacks are classified by *target* (from physical sensors to machine learning models), *technique* employed (e.g., obtaining data access, data poisoning, and model evasion) and type of *knowledge* available to the adversary (e.g., black box, gray box, white box attacks). Defenses are organized according to the steps of the machine learning model lifecycle they are applied, namely against *training time attacks* and *testing time (inference) attacks*. Consequences are classified according to the security attributes of the CIA triad: integrity violation, availability violation, and confidentiality violation.

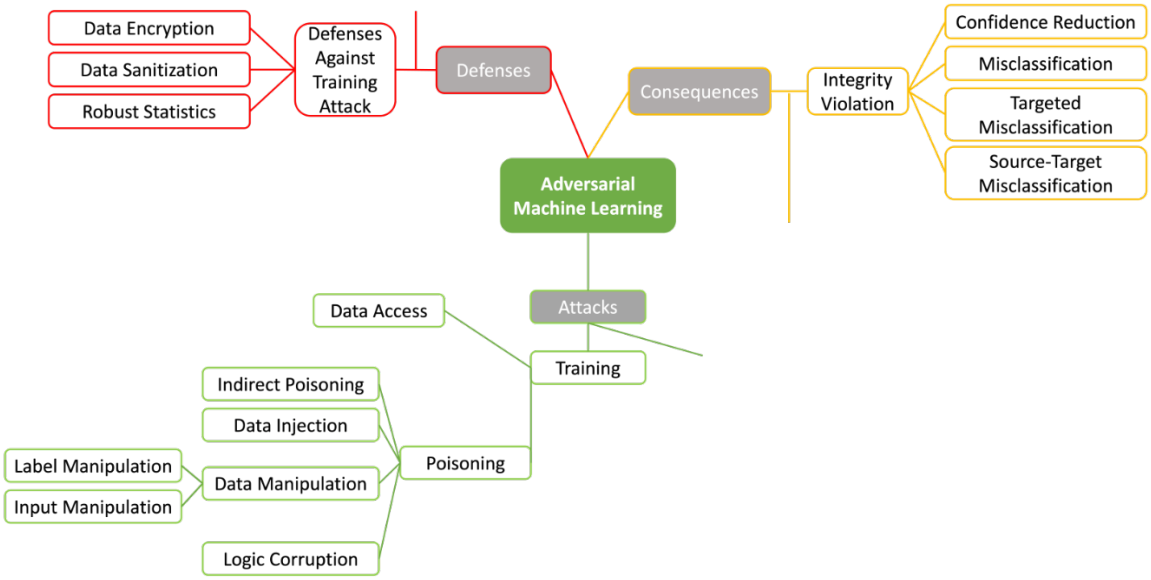

FIGURE 7: EXAMPLE OF THE NIST TAXONOMY ORGANIZED IN A HIERARCHICAL MANNER

3.3.2. ENISA SECURING MACHINE LEARNING ALGORITHMS REPORT

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

In 2021, the European Union Agency for Cybersecurity (ENISA) published a study named "Securing Machine Learning Algorithms" [48] that focuses on cybersecurity threats specific to machine learning algorithms and machine learning-based applications. Based on a systematic review of relevant literature (more than 200 documents), the study provides a taxonomy for machine learning algorithms, emerging threats, vulnerabilities, and appropriate security controls for mitigation. By drawing attention to threats specific to machine learning algorithms and applications, the study aims to support the risk analysis and threat identification process performed by developers and designers of machine learning-based applications and systems.

In their provided taxonomy, ENISA attempted to cover all steps of a typical machine learning pipeline i.e., data collection, data pre-processing, model training, model evaluation, monitoring, and more. Precisely, the identified threats, vulnerabilities, and recommended security controls were mapped to the individual steps of the machine learning pipeline. This enables a more granular and comprehensive analysis of the security of the whole machine learning pipeline in which developers and system designers can identify threats for each step and address them adequately. In this context, ENISA identified six high-level threats in their study: *evasion*, *oracle*, *poisoning*, *model and data disclosure*, *compromise of machine learning application components*, and *failure or malfunction of machine learning application*. Besides the listed high-level threats, seven-sub threats were determined. Based on the identified threats, several concrete vulnerabilities associated with these threats were discussed in the study.

The study identifies 37 recommended security controls that can be applied to mitigate the risk posed by the identified threats and vulnerabilities. ENISA categorized security controls into three categories: *traditional security controls* related to organizational and security policies, *classical technical security* controls, and *security controls specific to machine learning-based applications* and systems. Furthermore, security controls were mapped to specific threats and associated vulnerabilities. An excerpt of this mapping for the high-level *Evasion* threat is provided in Table 2.

The study concludes that traditional security controls are not enough to protect machine learning-based applications and, thus, need to be completed by the proposed security controls specific to machine learning applications. The study stresses that there is no unique strategy that can be applied to secure machine learning algorithms. Instead, appropriate mitigation measures depend on the specific (business) use case and often represent a trade-off between security and performance. Since no general protection strategy is available, the authors aimed to provide the broadest possible overview of threats, associated vulnerabilities, and security controls in their study. Depending on use-case-specific threat modelling, it is up to the developers to pick and employ appropriate mitigation measures that are proportional the use-case-specific threat level.

TABLE 2: AN EXCERPT OF THE PROVIDED MAPPING OF THREATS, CONCRETE VULNERABILITIES, AND RECOMMEND SECURITY CONTROLS

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

Threats I *sub-threats*		Vulnerabilities	Security Controls	Threats references
Evasion		Lack of detection of abnormal inputs	Implement tools to detect if a data point is an adversarial example or not	13, 34, 37, 48, 49, 51, 53, 56, 59, 60, 62, 65, 66, 67, 73, 80, 81, 82, 83, 84, 90, 95, 97, 100, 107, 109, 110, 121, 125, 139, 144, 154, 155, 162, 163, 169, 170, 175, 181, 183, 185, 199, 200, 201, 202, 204, 205, 206, 207, 209, 211, 213, 215
			Include ML applications in detection and response to security incident processes	
		Poor consideration of evasion attacks in the model design implementation	Choose and define a more resilient model design	
		Lack of consideration of attacks to which ML applications could be exposed	Integrate ML specificities to awareness strategy and ensure all ML stakeholders are receiving it	
		Lack of training based on adversarial attacks	Add some adversarial examples to the training dataset	
		Use a widely known model allowing the attacker to study it	Lack of security process to maintain a good security level of the components of the ML application	
			Use less easily transferable models	
			Assess the exposure level of the model used	
		Inputs totally controlled by the attacker which allows for input-output-pairs	Apply modifications to inputs	
	Use of adversarial examples crafted in white or grey box conditions (e.g. FGSM…)	Too much information available on the model	Reduce the available information about the model	34, 35, 48, 51, 56, 59, 60, 62, 65, 80, 81, 82, 100, 109, 110, 125, 139, 144, 154, 170, 204, 209
		Too much information about the model given in its outputs	Reduce the information given about the model	

3.3.3. MITRE ADVERSARIAL MACHINE LEARNING THREAT MATRIX

The Adversarial machine learning Threat Matrix [49] is an open-source MITRE ATT&CK style framework [50] intended to position attacks against artificial intelligence-based systems. It was created by both industry and academic research groups to enable security researchers to navigate and orient themselves through the landscape of new and upcoming threats against artificial intelligence systems. In recent years, machine learning systems developed by big companies such as Google and Tesla were evaded or misled by cyberattacks and the industry seemed to be unprepared for adversarial machine learning vulnerabilities. This matrix aims to address this issue. As machine learning threats are based upon limitations in underlying machine learning algorithms and data analysis processes, unlike existing software and hardware system vulnerabilities, the creation of the Adversarial machine learning Threat Matrix with a curated set of machine learning vulnerabilities and adversarial behaviors is indeed necessary and important.

The Adversarial Machine Learning Threat Matrix mimics the layout of the popular MITRE ATT&CK framework for cybersecurity, allowing cybersecurity and machine learning engineers to easily understand adversarial behaviors in terms of *tactics* and *techniques*. Figure 8 presents an excerpt of the Adversarial Machine Learning Threat matrix, which covers 12 attack stages against artificial intelligence-based systems as columns from left to right. Each column is a "*Tactic*", that corresponds to broad categories of adversary techniques. Each cell is a "*Technique*", which belongs to a tactic group. Some popular adversarial machine learning techniques are discussed in detail in Section 3.2 of the report. For example, the technique

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

"*Evade machine learning Model*" [51] allows us to explore existing model evasion attacks with some real case-study examples and then understand the attack procedure step by step.

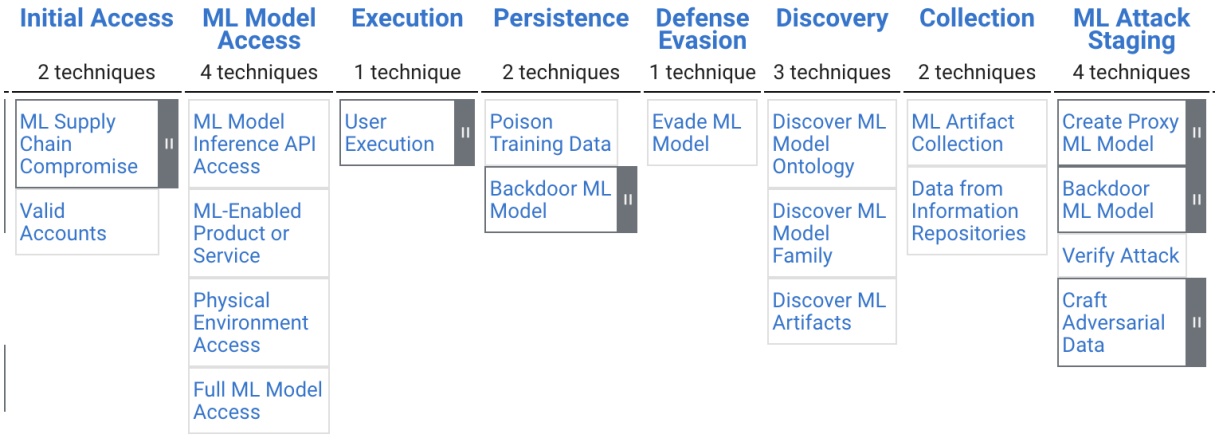

FIGURE 8: AN EXCERPT OF THE ADVERSARIAL MACHINE LEARNING THREAT MATRIX

The Adversarial Machine Learning Threat Matrix is still an early attempt to develop knowledge related to how machine learning systems are vulnerable to machine learning attacks. New attacks and real case studies against production machine learning systems need to be updated frequently in this framework to reflect the rapidly evolving adversarial machine learning attack lifecycle and new machine learning threat vectors [52].

3.3.4. MICROSOFT THREAT MODELLING AI/MACHINE LEARNING SYSTEMS

To improve threat modeling practices specific to artificial intelligence-based systems, Microsoft has created two living documents – "*Failure Modes in Machine Learning*" [53] and "*Threat Modeling AL/ML Systems and Dependencies*" [54], that will evolve over time with the threat landscape. The former is written for a wide interdisciplinary audience, like lawyers and policy makers. It organizes failures and consequences of attacks into different categories. The latter targets more technical users, like security engineers and data scientists. It is designed to help them identify potential machine learning threats and vulnerabilities and then use the Microsoft threat modeling framework to plan for appropriate countermeasures.

Failure Modes in Machine Learning [53] consists of two main sections about intentional and unintentional failure modes. It provides a brief definition of attacks and includes illustrative examples from the literature. As shown in Table 3, the intentional failure mode section provides details about intended attacks, compromised security attributes by asset, knowledge required by an attacker, and access and authorization violation information.

TABLE 3: AN EXCERPT OF INTENTIONALLY MOTIVATED FAILURES SUMMARY TABLE [53]

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

Scenario Number	Attack	Overview	Violates traditional technological notion of access/authorization?
1	Perturbation attack	Attacker modifies the query to get appropriate response	No
2	Poisoning attack	Attacker contaminates the training phase of ML systems to get intended result	No
3	Model Inversion	Attacker recovers the secret features used in the model by through careful queries	No
4	Membership Inference	Attacker can infer if a given data record was part of the model's training dataset or not	No
5	Model Stealing	Attacker is able to recover the model through carefully-crafted queries	No

Threat Modeling AL/ML Systems and Dependencies [54] is divided into two sections, focusing on some new questions to ask when threat modeling artificial intelligence-based systems and specific mitigation solutions used at Microsoft against existing attacks, respectively. Still considering model stealing attacks, the following questions could be asked in a security review and proactive/protective mitigation actions like minimizing or obfuscating the details returned in prediction APIs are proposed:

- *"What is the impact of your model being copied/stolen?"*
- *"What would it take to get your model to return a result that tricks your service into denying access to legitimate users?"*
- *"Can your model be used to infer membership of an individual person in a particular group, or simply in the training data?"*

3.3.5. WITHSECURE SECURITY SELF-ASSESSMENT QUESTIONNAIRE

Assessing the security risk associated with a machine learning system is currently challenging due to three main factors:

- A lack of awareness about vulnerabilities and attacks specific to machine learning systems.
- A lack of understanding of the attack vectors leading to exploitation of vulnerabilities once machine learning models are integrated into larger systems.
- Limited availability of experts with a deep understanding of both security and machine learning.

To partially address these challenges, and similar to Threat Modeling AL/ML Systems and Dependencies [54], but in a more comprehensive manner, WithSecure designed three questionnaires to assist machine learning practitioners, security experts, and decision makers in the risk assessment process [55]. These questionnaires contain leading questions to help understanding of security risks associated with machine learning systems. The questions were designed to help respondents gain understanding into the vulnerabilities and possible attacks against their own machine learning systems. The questions also hinted at measures that can be

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

adopted to reduce vulnerabilities and mitigate attacks. The three questionnaires were defined as follows:

- **Risk and impact assessment,** designed to assess how well the respondent manages the security risks associated with their machine learning system(s). It analyzed approaches to threat analysis and impact assessment both in a generic context and when considering security threats specific to machine learning systems.
- **Attack surface and vulnerabilities assessment,** designed to help identify the attack surface of machine learning systems and to discover potential vulnerabilities at several stages during their lifecycle.
- **Security of your machine learning system assessment,** designed to assess the security and robustness of machine learning systems. The questionnaire was intended to help identify if the respondent was aware of security threats against their own systems and how they might discover potential vulnerabilities. It explored whether recipients had processes and techniques in place to mitigate potential attacks.

Each questionnaire was designed to be answered individually by people in different roles. An organization answering all three questionnaires gained a complete picture of the security posture of the assessed machine learning system. The goals of these three questionnaires were four-fold:

- Raising awareness about security threats against machine learning systems.
- Assisting machine learning practitioners to assess the security of their own machine learning systems.
- Sharing solutions and practices that can improve the security of machine learning systems.
- Using the collected answers to infer global trends about the current state of machine learning system security.

3.3.6. VULNERABILITY ASSESSMENT TOOLS

Practical tools have been developed to empirically assess the vulnerabilities of machine learning systems. These tools can be launched against actual machine learning systems to evaluate their reliability and robustness against attacks that exploit known algorithmic vulnerabilities, including evasion, poisoning, and data inference attacks. These tools can be used on deployed systems, to assess vulnerabilities, or during system development to improve robustness. Several free and open-source adversarial machine learning libraries of this kind are available. They typically provide interfaces to popular machine learning frameworks such as TensorFlow, PyTorch, and scikit-learn.

The Adversarial Robustness Toolbox (ART) library initiated by IBM and maintained by the Linux Foundation [56] is among the first and most complete of these security assessment libraries, providing several evasion, poisoning, extraction, and data inference attacks. Nevertheless, completeness comes with complexity, and ART requires significant machine learning expertise and knowledge of adversarial machine learning attacks to be used effectively. Microsoft Counterfit [57] is another generic tool for security testing of machine learning systems that can be operated with less familiarity with machine learning skills thanks to its command line

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

interface. Many other specialized libraries have also been published like CleverHans [58], which focuses on evasion attacks or MIA [59] for data inference attacks. Many of these libraries also include defenses against the attacks they provide.

These empirical assessment tools are useful for developing an understanding of vulnerabilities in the context of machine learning systems, which is an important step of for threat modelling. However, current libraries for machine learning vulnerability assessment implement attacks that are restricted to a few tasks and type of models (such as neural networks). They are also limited to specific data types (such as image data), and they do not generalize over other data types. Consequently, they cannot be applied to any machine learning context and are mostly useful for studying neural networks that process images and text. These libraries need to be improved further to be more generic and easier to apply on any machine learning system.

3.3.7. ARTIFICIAL INTELLIGENCE INCIDENT DATABASE

The artificial intelligence incident database is an open repository of documented failures from real-world artificial intelligence systems. Its goal is to record and make available information about unforeseen and dangerous failures of intelligent systems, such that designers of future artificial intelligence systems may avoid repeating documented bad outcomes. The database is intended to be used by artificial intelligence system architects, developers, and policy makers to mitigate future risks of failure in artificial intelligence systems. Anyone can contribute to it by reporting the artificial intelligence incidents they have witnessed. It currently contains over 1,000 reported incidents such as autonomous cars killing pedestrians, trading algorithms causing a market flash crash, and facial recognition systems misidentifying innocent people as criminals and causing their arrest. The database is in ongoing development, and it now includes taxonomies to analyze and document incidents in a consistent manner. The format of each incident report is partly inspired from the aviation and computer security industries. In their current form, reported incidents contain information about the consequences of the incident based on the Microsoft artificial intelligence Fairness Checklist [60] and about the data used in the failing artificial intelligence system using the datasheet for datasets taxonomy [61].

The artificial intelligence incident database is not security oriented nor restricted to security incidents. Reported incidents can be caused by genuine mistakes, misconfigurations, or simply unforeseen events that are not malicious or adversarial in nature. Moreover, the current taxonomy for reporting incidents does not cover information that would be required for security incidents, such as the vulnerability exploited, severity of the vulnerability, or means to fix it. Nevertheless, it is already a relevant source of information in its current form and can be used to identify how artificial intelligence systems can fail. Non-adversarial failures can often be translated into security threats and associated vulnerabilities to exploit. The database is an ongoing initiative that can be further enhanced to be more security focused, by augmenting incident reports with artificial intelligence security taxonomies such as the MITRE adversarial matrix or the ENISA machine learning security taxonomy [48], as well as conventional information about security incidents such as the taxonomy of vulnerabilities from CVE [4].

3.4. SUMMARY

Threat modeling is a complex exercise that goes beyond the simple identification of threats. It requires system descriptions, definitions of functionality, identification of components and parties that interact with them. Based on such a description, security threats against the system

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

can be identified, typically by a security expert. Existing threat modeling approaches are suited for performing initial steps, since they are generic to any system – they only require high-level knowledge of the machine learning system and conventional security expertise.

Challenges appear when identifying vulnerabilities that can be exploited by identified threats. Specific assets of machine learning systems, such as machine learning models, machine learning libraries, and data expose new vulnerabilities that were unknown until recently, and which are not included in conventional threat modeling frameworks or conventional security taxonomies. These vulnerabilities create new attack vectors for which there are little to no security controls. To cope with the shortcomings of existing threat modeling approaches, we presented these new vulnerabilities and classified them in algorithmic, supply chain, and deployment vulnerabilities. Although we described attacks that exploit these vulnerabilities, discussions about security controls available to mitigate these attacks were omitted. Existing security controls are not suitable for preventing attacks – they only mitigate them, and they are typically application-, model- and/or data-specific. Thus, they are not generic enough to be included in global recommendations.

Several parallel efforts have also been initiated by public organizations, including NIST, ENISA, and MITRE and companies including IBM, Microsoft, and WithSecure to cope with the lack of knowledge on vulnerabilities and attacks against machine learning systems. New security taxonomies specific to machine learning systems have been created, guiding questionnaires have been proposed to support threat modeling, repositories for gathering and documenting artificial intelligence incidents, and practical vulnerability assessment tools have been created. Nevertheless, while these efforts are complementary, they are still isolated and must be made compatible and integrated together to effectively assist in the threat modeling effort. Moreover, while taxonomies and questionnaires are rather generic, vulnerability assessment tools are restricted to a few machine learning systems and still incomplete.

When introducing a threat modeling approach, one must meet a trade-off between generality – making the approach applicable to many systems – and specificity – making it easily actionable without a high level of expertise.

Based on the description of vulnerabilities presented in Section 3.2, we aim to strike a balance between genericity and specificity by threat modeling dominant training and inference architectures. We focus on challenging threat modeling steps, namely identification of vulnerabilities they expose and attack vectors that exploit them. In the next section we illustrate three challenging steps of the threat modeling process for each architecture:

- Defining entity access to machine learning system assets based on architecture.
- Identifying the algorithmic, supply chain, and deployment vulnerabilities exposed.
- Presenting attacks that can exploit them and a discussion of their effectiveness and likelihood.

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

4. SECURITY ANALYSIS OF MACHINE LEARNING ARCHITECTURES

This section presents an analysis of security threats against machine learning systems from the perspective of their lifecycle. Machine learning-specific security threats arise mostly during either training or inference phases. Depending on the architecture used during each of these two phases, different threats exist.

4.1. TRAINING ARCHITECTURES

The training phase of the machine learning system lifecycle exposes one algorithmic vulnerability: the model poisoning attack. It further exposes all supply chain vulnerabilities, as well as the compromised training platform vulnerability.

Compromised pre-trained machine learning models present a supply chain vulnerability. Starting with a pre-trained model exposes the model's owner to the risk of it having been compromised by a potentially untrusted source. The same compromised pre-trained model might be shared with several parties, exposing them all to the threats described in Section 3.2.2.2.

4.1.1. CENTRALIZED TRAINING

In centralized training the data sources, storage platform and training platform have access to training data. The storage and training platform are typically under the same authority and can be a single entity – the central server – which is also the system where the model is trained. The central server is the only party with access to the machine learning model.

Algorithmic vulnerability: poisoning attack

The first poisoning attack example defined in this document is illustrated by a centralized training architecture setup, where the training data is aggregated in a central location and training happens on a single machine. Poisoning attacks can be performed in two different ways in this setting. In the first method, one or several malicious *data source(s)* are used in a *data injection attack*. Data points are maliciously crafted such that when they are aggregated with legitimate data sources, part of the whole training dataset will be poisoned. The second method involves a malicious data storage platform that can perform a *data modification attack* on the whole training dataset that has been aggregated. The data modification attack is a bigger threat than the data injection attack since the adversary has the knowledge and ability to modify the whole training dataset. The data modification attack is more effective than data injection in centralized training and a malicious data storage platform is thus a bigger threat than a malicious data source with regards to poisoning attacks.

Label modification attacks can be performed if external parties (not necessarily represented in our machine learning pipeline in Figure 2) are able to modify labels after data is collected, e.g., through feedback. This attack is less effective than data injection and data modification, and it is thus a moderate threat since adversaries will only be able to modify a small number of labels through feedback, and they will not have knowledge of the whole training set.

Supply chain vulnerabilities

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

A machine learning model's training process can be exposed to supply chain vulnerabilities in the form of compromised machine learning libraries supplied by potentially malicious external parties. Such an attack can be used to backdoor or poison a machine learning model, causing it to display lower-than-expected accuracy on respectively targeted or untargeted data. Such an attack can also cause the model to leak information about its training data during inference.

Compromised serialization libraries affect all parties in the centralized training pipeline. Data sources and data storage platforms can easily by exposed to compromised data serialization libraries if they are not verified during the continuous build process. This is a relatively common scenario given that each data source may typically source its own library to serialize the data that must be sent to the storage platform. Thus, any part of the pipeline may potentially inadvertently source a compromised library. Similarly, the data storage platform itself may source a compromised library for serializing the aggregated dataset. The threat to the data storage platform is once again more important than it is on individual upstream data sources since it the threat affects the whole dataset, not just part of it. Finally, the training platform may source a compromised library for serializing the machine learning model before delivering it to the deployment platform. This threat is also severe since it leads to a complete and definite compromise of the machine learning model. Compromise at the data source or storage platform can be partly mitigated during training with data sanitization techniques.

Deployment vulnerability: compromised training platform

Centralized training architectures are exposed to threats against the privacy and confidentiality of the training data. Training data is often gathered from multiple clients and stakeholders. Privacy violations can occur if the gathered data contains sensitive or personal information. Data sources must fully trust the central storage and training server. This central entity must be adequately secured to prevent any compromise by an attacker, since the latter would get full access to all the sensitive training data at once.

The central storage and training server also represents a single point of failure in the underlying system. In case its operation is disrupted, or if attackers succeed in compromising it, the security of the entire system is affected. An attacker compromising a training platform can replace a trained machine learning model with any alternative malicious model or simply make it unavailable. Central servers require high security protection to ensure the integrity and availability of training data, the training process, and the resulting machine learning model. Centralized training environments that use properly secured reputable cloud services are already likely well secured from such attacks.

4.1.2. DISTRIBUTED TRAINING

In distributed training, the data sources, storage platform and training platform all participate in the training process, similar to the centralized training case. However, and in contrast to centralized training, the training platform is composed of a central parameter server and several worker nodes. Both the parameter server and worker nodes have access to the training data and to the machine learning model, which they train jointly.

Algorithmic vulnerability: poisoning attack

The poisoning threat coming from malicious data sources and malicious storage platform, with respectively data injection and data modification attacks, applies in the same manner to

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

centralized and distributed training. In addition, worker nodes involved in distributed training can be compromised and poison the training of the model.

In the data parallelism paradigm, compromised worker nodes can maliciously modify the chunk of data allocated to them with a data modification attack. A poisoned model is aggregated with other local machine learning models into a global model. In this scenario, the poisoned local model will partly compromise the global model. Poisoning attacks in distributed training settings can be improved if the malicious worker node directly modifies its local model instead of its training data. Knowing the model aggregation process, the worker node can craft its local model so that it overwrites the local model contributions from legitimate worker nodes. This poisoning attack is very effective in distributed training settings, and it is called a model replacement attack [62].

In the model parallelism paradigm, a compromised worker node can tamper with the intermediate data it produces during the optimization of its part of the model which is used as input to other nodes for optimizing their own part of the model. It can also tamper with the part of the model it is supposed to train, e.g., one layer of a neural network can be modified. In both cases, poisoning performed by a single worker node is a bigger threat against model parallelism than it is against data parallelism. While each worker node can modify only part of the model in model parallelism, a single malicious worker node will affect the whole model and can completely destroy its accuracy. For instance, modifying a single layer of a neural network, changes completely the whole model since all layers are interdependent. In any case, distributed training brings new parties in the training process, the worker nodes, increasing the attack surface for, and vulnerability to poisoning attacks when compared to centralized training.

Supply chain vulnerabilities

The exposure to using malicious machine learning libraries is extended to worker nodes in distributed training. It is expected that every worker node would use the same machine learning training library, as provided by the parameter server. Thus, the threat of compromised machine learning library is not increased when compared to centralized training: either all machine learning libraries are compromised if the central server has a compromised library, or they are not. Data serialization libraries can however be different for different worker nodes in the data parallelism model, where they are used to unpack batches of training data assigned to them. Each batch of training data can be compromised and poisoned if the receiving worker node uses a compromised serialization library to unpack it.

Deployment vulnerability: compromised training platform

The training platform is not a single entity anymore in distributed training. It is composed of the parameter server and the worker nodes. With more entities having access to the training data, data parallelism increases the threats against data privacy and confidentiality, when compared to centralized training. The likelihood of training data leakage increases since it requires only one node to be compromised and worker nodes may not be as secure as the parameter server. The impact of compromising a single node is nonetheless lower than a compromise of the parameter server or the central server in centralized training since it only leads to the adversary having access to a subset of the training data. In model parallelism, most worker nodes do not have access to the training data, only to intermediate parameters, reducing the threat of data leakage. On the other hand, a few worker nodes responsible for training the input layer of the machine learning model would have access to the whole training dataset. If one of these nodes

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

gets compromised, the adversary can leak the whole training set and compromise the privacy of clients that contributed to it.

Beyond confidentiality, malicious worker nodes can also compromise the availability of their local model updates. In data parallelism, such attack could significantly delay the global training process, but measures can be deployed to cope with unavailability of worker nodes. For instance, their updates can be disregarded and only local models from responsive nodes could be aggregated. In model parallelism, such an availability attack stops the training process completely since each worker node is dependent on updates from another worker node. A single worker node under an availability attack would prevent the training process from completing. Integrity attacks can also be performed on malicious nodes and such attacks were described previously as part of the algorithmic vulnerability to poisoning attacks.

4.1.3. FEDERATED LEARNING

Federated learning is different from distributed training because data sources, called *clients*, are also the worker nodes that participate in the distributed training process. Consequently, there is no data storage platform in this paradigm. Each client has access to its own training data, their local machine learning model, and to the global machine learning model. The centralized parameter server has access to all local machine learning models and to the global machine learning model, but it does not have access to any training data.

Algorithmic vulnerabilities

Poisoning can be performed on clients because federated learning is typically performed on many untrusted clients. Thus, the likelihood of several malicious clients being involved in federated learning is high. Each client can poison its training data or its own local machine learning model in the same way as for the worker nodes in distributed training, with the goal that their poisoned local model will compromise the aggregated global model. Once again, the model replacement attack is a serious threat, since it is an effective poisoning attack that requires the compromise of a single client in federated learning [62].

Federated learning also exposes an additional algorithmic vulnerability that other training paradigms do not: training data inference. A malicious parameter server can run training data inference attacks on the local machine learning models sent by clients in order to reconstruct their local training data [41]. This threatens the confidentiality of training data and the privacy of clients. This threat is even more important in peer-to-peer federated learning architectures where local machine learning models are shared with many other clients, many of which are not trusted. Moreover, some malicious clients can also compromise the confidentiality of the training data held by other clients in the central aggregation paradigm. By re-training the global machine learning model in a specific manner and sharing the resulting local machine learning model with the parameter server, a malicious client can simultaneously train a discriminator and a generator machine learning model (Generative Adversarial Network) able to reconstruct the training data held by another client [63]. Federated learning is the only training paradigm that exposes the training data inference threat, which applies mostly to the federated training of neural network models.

Supply chain vulnerabilities

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

The exposure to using malicious machine learning and serialization libraries applies to any client independently, since they typically choose libraries of their own for training local machine learning models and exchanging them with the parameter server. There is thus a higher chance that one or a few clients use compromised libraries that would threaten the integrity of their local model updates. The parameter sever is exposed to using a compromised aggregation library, which can tamper with the aggregation process of local models and threaten the integrity of the global model, leading to it being poisoned or backdoored.

Deployment vulnerability: compromised training platform

Federated learning offers good privacy and confidentiality guarantees for client data because no local data is shared with the parameter server or between clients. While algorithmic vulnerabilities which can compromise this property exist, there also exist defense solutions to mitigate them, such as secure aggregation [64].

On the other hand, and as pointed out earlier, the likelihood for clients to be compromised is high. This threat has little impact on the availability of local machine learning models. The federated learning process is designed to deal with potential client's drop-off during training. The process relies on a large pool of clients, from which only a few are randomly selected to contribute to the training during each round. Thus, if a compromised client becomes unavailable, it can be easily replaced by another client from the pool. On the other hand, malicious clients pose a threat to the integrity of the global model and to training data confidentiality as previously pointed out. They are also a threat to the global model confidentiality since all clients receive a copy of the global model and can thus easily steal it and share it with third parties if they wish. In the peer-to-peer federated learning paradigm, the likelihood of local model aggregation compromise is also high since this aggregation is performed by any client.

4.2. INFERENCE ARCHITECTURES

The inference phase of the machine learning system lifecycle exposes three algorithmic vulnerabilities: model evasion, model stealing and training data inference attacks. It further exposes vulnerabilities related to compromised machine learning libraries and serialization libraries, as well as a compromise of the inference platform.

4.2.1. CENTRALIZED INFERENCE

In centralized inference architectures, the machine learning model is deployed on a central server which provides an API for clients to submit queries and receive predictions. Consequently, the central server / deployment platform has access to the queries submitted by clients, the machine learning model, and its predictions. Clients have access to their own inputs and to the predictions from the machine learning model, which can have different levels of granularity, e.g., label, probability, prediction vector, etc.

Algorithmic vulnerabilities

Model evasion attacks can be run by API clients against a machine learning model deployed on a central server. In this setting, the attacker neither has direct access to the machine learning model nor knows what type of model they are interacting with, because the machine learning model is behind an API. This is a black-box evasion attack. Attackers can only repeatedly query

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

the machine learning model API with different inputs to get predictions in the form of labels, probabilities, or prediction vectors. Forging a successful adversarial example that can evade the target machine learning model requires many queries and the evasion attack will have a lower success rate than if performed in a white-box setting [65] [66]. The granularity of the predictions returned by the prediction API impacts the success of the attack. Fine-grained prediction vectors lead to more successful black-box evasion attacks than coarse-grained label predictions.

Model stealing is an attack originally developed in the centralized inference setting, where a malicious API client wants to steal the machine learning model hosted on a central server and kept secret behind a prediction API [34]. This setting is thus highly exposed to model stealing attacks using API queries. The importance of the threat will mainly depend on the granularity of predictions. Coarse-grained predictions will again make model stealing attacks more difficult than fine-grained predictions [35]. Some rate limiting process for requests to the query API can also slow down model stealing attacks.

Machine learning models deployed on a central server are vulnerable to all training data inference attacks which can be run in a black-box manner by API clients to infer information about the training data. Membership inference is among the most effective attacks in this black-box setting [39]. Once again, the granularity of predictions heavily impacts the success of data inference attacks, and all data inference attacks will be less successful in this black-box setting than they would be in a white-box setting.

Supply chain vulnerabilities

Supply chain vulnerabilities are performed by exposing the central server to compromised machine learning libraries or compromised serialization. Both compromises threaten the integrity of model predictions. Such attacks lead to either the inputs to the machine learning model or its decision process being tampered. In such scenarios, the attacker will aim to maliciously modify model predictions. The serialization library used to format queries sent to the inference API can also be potentially compromised. This compromise leads to the same consequence of obtaining incorrect predictions as response from the inference API.

Deployment vulnerability: compromised deployment platform

A compromised deployment platform has heavy implications with respect to the confidentiality, integrity, and availability of a machine learning model and its predictions. Both assets can be completely modified to become incorrect, the server can refuse to provide an inference service, or it can leak the model to a third party. Nevertheless, the compromise of a central server to that extent is unlikely, especially if a trusted service provider is used. Considering a more realistic honest but curious deployment platform, the confidentiality of client queries can be threatened. If client queries contain personal identifiable or privacy-sensitive data, the malicious server can access and leak this information in a stealthy way that would remain unnoticed.

4.2.2. LOCAL INFERENCE

In local inference, a machine learning model, potentially trained by an external party, is deployed on clients. Thus, each client has access to the machine learning model, its predictions and to queries made to the model. If there is a central entity coordinating the distribution of machine

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

learning models to clients, it has only access to the machine learning model and it has no access to queries or predictions. The main threats in this scenario relate to the machine learning model and its predictions if these predictions are meant to be used by a party other than the client itself.

Algorithmic vulnerability

Model evasion attacks can be run by clients against the locally deployed model, via local inference. This type of attack is useful if the predictions from the machine learning model are not used by the client but by an external party, e.g., a malware detection system. Each client has direct access to the machine learning model, and it knows its type and its internal decision logic. Thus, white-box evasion attacks are possible. If such a model is deployed on many clients, an attacker can forge adversarial examples against its own local model that will evade machine learning models deployed on other clients. This type of attack can be used to craft malware samples that evade client-side machine learning-based malware detectors.

Model stealing attacks can be performed against client-side models. but only if the model is not distributed in clear to clients, (such as if it is delivered to and run in a trusted execution environment that protects its confidentiality through a query API). Such attacks are very similar to those that can be performed against the centralized inference paradigm. The difference in the client-side scenario is that it is harder to implement mitigation techniques such as query rate limiting, since the owner has less control over the client-side system. Overall, model stealing attacks are easier to run in local inference settings.

Training data inference attacks can be run against local machine learning models in order to infer information about the data used to train the model. It is an important threat in local inference since typically no client using the machine learning model has contributed to the training data (except if the model was trained using federated learning). The most effective white-box data inference attacks can be run in this setting since the adversary has direct access to the machine learning model and they know its type and its internal decision logic. Model inversion attacks that are able to fully reconstruct training data from scratch can be run effectively in this setting [41].

Supply chain vulnerabilities

In a local inference setting, the compromise of machine learning libraries poses a significant threat to the integrity of the decision making of the model and thus the integrity of its predictions. A compromised serialization library can also pose a threat during the delivery of the machine learning model from an external party and to the client. The local machine learning model's integrity can be compromised during this step, which happens before inference. Vulnerabilities related to compromised data serialization libraries do not apply here, since the data is generated and processed on the client and thus does not need to be serialized.

Deployment vulnerability: compromised deployment platform

The deployment platform is the client in this paradigm. The main threats from a compromised client are against the machine learning model. A malicious client can compromise the confidentiality, integrity, and availability of the machine learning model. A secondary threat exists against the predictions from the machine learning model, if used by an external party. A malicious client can compromise the integrity and availability of these predictions that are supposed to be delivered to the external party.

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

4.2.3. DISTRIBUTED INFERENCE

In distributed inference, a machine learning model is split into multiple pieces and distributed across several devices or worker nodes. Each worker node only has access to part of the machine learning model. Depending on the paradigm, worker node other than the one making the request may have full, partial, or no access to the input data and to the model's prediction. For instance, in layer-wise splitting, the node executing the first layer of the DNN model has access to the input data, the node executing the last layer has access to the predictions and all other nodes do not have access to any meaningful data. In across-the-layer splitting, each node has access to part of the input data and part of the prediction.

Algorithmic vulnerabilities

In a distributed inference setting, it is difficult to run any attack that would exploit the algorithmic vulnerabilities of machine learning systems. A single node neither has access to the full model, preventing any white-box attack, nor access to the input and predictions at the same time. This prevents black-box attacks that require both inputs and predictions. The node that makes inference requests can run evasion, model stealing, and training data inference attacks in the same way as in the centralized inference scenario, in a black-box manner. Vulnerabilities to these three attacks are similar to the centralized inference scenario, with the exception of the fact that attacks exploiting them would take longer to complete since the inference process is slower due to communication overhead between hosts.

Supply chain vulnerabilities

Every node involved in a distributed inference process is vulnerable to attack via compromised machine learning libraries. If a single node uses a compromised machine learning library, the whole inference process will be compromised, and the machine learning model will return incorrect predictions. The same applies for serialization libraries, that are used for data exchange between nodes during the inference process. If one intermediate data sample is compromised during serialization, it jeopardizes the integrity of the whole inference process. Since all nodes are typically independent devices, that can be potentially compromised by different parties, the threat of one node using either a compromised machine learning or serialization library is very high. Each node represents a single point of failure since all are equally as important.

Deployment vulnerability: compromised deployment platform

If a single node in a distributed inference process has been compromised, the integrity and availability of its machine learning model and predictions are threatened. The confidentiality of the machine learning model and the input data is generally less threatened by the deployment platform compared to other inference architectures since no single node has full access to either of these assets. Nevertheless, a worker node, which is also a client in this setup, will have more knowledge about the machine learning model than a client in the centralized inference paradigm. It can know the type and general structure of the machine learning model along with some of its parameters which can help inferring the rest of the parameters via a model stealing attack. Nevertheless, it is worth noting that worker nodes in distributed inference are more susceptible to compromise than a server in a centralized inference scenario. Worker nodes are usually simple devices with limited computing capability and low security.

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

5. SECURITY REQUIREMENTS FOR ML ARCHITECTURES

Two high level requirements for machine learning models related to their security and to the privacy of the data they use have been previously identified:

- **MOD.RQ.5: Resilience against adversarial machine learning attacks** – machine learning models must be resilient against adversarial machine learning attacks
- **MOD.RQ.6: Protect data privacy** - machine learning models should not leak information about their training data

The detailed analysis of security threats against artificial intelligence-based systems presented in this report led to a definition of more fine-grained security and privacy requirements for machine learning models. These requirements aimed to mitigate the vulnerabilities specific to machine learning systems, in order to minimize the security risk generated by the threats we identified. As part of the security requirements for artificial intelligence-based systems, we identified the following requirements designed to address four algorithmic vulnerabilities:

- **SEC.RQ.2: Resilience against poisoning attack** – machine learning models MUST be resilient against poisoning attacks
- **SEC.RQ.1: Resilience against evasion attack** - machine learning models MUST be resilient against evasion attacks
- **SEC.RQ.4: Resilience against model stealing attack**- machine learning models MUST be resilient against model stealing attacks
- **SEC.RQ.6: Resilience against training data inference attack** - machine learning models MUST be resilient against training data inference attacks

The document further recommended additional fine-grained security requirements to mitigate algorithmic, supply chain, and platform vulnerabilities inherent in artificial intelligence-based systems. These new security requirements, which are summarized in Table 4, are as follows.

Enforcement and verification of training data integrity: The integrity of training data must be guaranteed between data sources and the training platform. This reduces the chance that poisoning attacks will happen.

Enforcement and verification machine learning model integrity: The integrity of the machine learning model must be guaranteed between the training platform and the inference platform.

Limited number of parties involved in machine learning model training and inference: Every party involved in a machine learning pipeline can be a potential attacker. Considering distributed training and inference, the number of parties involved in these processes should be restricted to the required minimum.

Control and monitoring of clients' access to machine learning model: Increased knowledge and access to a machine learning model increases the success of most attacks against machine learning models during inference. This access should be restricted to a bare minimum to ensure functional requirements. The machine learning model should be as isolated as possible from its clients and every interaction must be monitored to detect potential abuse.

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

Secure training platform: The training platform for the machine learning model must be properly secured to prevent any compromise.

Secure deployment platform: The platform where the machine learning model is deployed for inference must be properly secured to prevent any compromise.

Verification of inputs from external parties: The provenance and integrity of inputs provided or computed by external parties, e.g., training data, local models, part of global model in distributed training, intermediate prediction results in distributed inference, etc. should be verified before use.

Vetting and verification of external libraries: Providers of external libraries must be trusted. In addition, the provenance and integrity of external machine learning and serialization libraries used to process machine learning model or data should be verified.

TABLE 4: SUMMARY OF NEW SECURITY REQUIREMENTS

Identifier	Security Requirements	Priority
SEC.RQ.9	The integrity of the training data MUST be guaranteed between the data sources and the training platform.	MUST
SEC.RQ.10	The integrity of the machine learning model MUST be guaranteed between the training platform and the inference platform.	MUST
SEC.RQ.11	The number of parties involved in machine learning model training and inference SHOULD be restricted to the required minimum	SHOULD
SEC.RQ.12	The machine learning model SHOULD be as isolated as possible from its clients and every interaction must be monitored to detect potential abuse.	SHOULD
SEC.RQ.13	The training platform for the machine learning model MUST be properly secured to prevent any compromise.	MUST
SEC.RQ.14	The deployment platform for the machine learning model MUST be properly secured to prevent any compromise.	MUST
SEC.RQ.15	The provenance and integrity of inputs provided or computed by external parties SHOULD be verified.	SHOULD
SEC.RQ.16	The provenance and integrity of external machine learning and serialization libraries used to process machine learning model or data SHOULD be verified.	SHOULD

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

REFERENCES

[1] National Institute of Standards and Technology, "Security and Privacy Controls for Information Systems and Organizations," https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final, 2022.

[2] C. Paulsen and R. Byers, "NISTIR 7298 Revision 3: Glossary of key information security terms," National Institute of Standards and Technology Interagency, 2019.

[3] ENISA, 2022. [Online]. Available: https://www.enisa.europa.eu/topics/csirts-in-europe/glossary/vulnerabilities-and-exploits. [Accessed August 2022].

[4] Cve.org, 2022. [Online]. Available: https://www.cve.org/About/Overview. [Accessed August 2022].

[5] S. Fenz and A. Ekelhart, "Formalizing information security knowledge," in *Proceedings of the 4th international Symposium on information, Computer, and Communications Security*, 2009.

[6] M. Roscini, Cyber Operations and the Use of Force in International Law, Oxford University Press, 2014.

[7] M. Uma and G. Padmavathi, "A survey on various cyber-attacks and their classification," *Int. J. Netw. Secur.,* vol. 15, no. 5, pp. 390-396, 2013.

[8] National Institute of Standards and Technology, "Guide for Conducting Risk Assessments - Special Publication 800-30," 2012.

[9] "Security Risk Assessment - Synopsis," [Online]. Available: https://www.synopsys.com/glossary/what-is-security-risk-assessment.html. [Accessed August 2022].

[10] National Institute for Standards and Technology, "Developing Cyber-Resilient Systems: A Systems Security Engineering Approach - SP 800-160," 2021.

[11] International Organization for Standardization (ISO), "ISO/IEC 27001: Information Security Management".

[12] National Institute for Standards and Technology, "NIST Cybersecurity framework," [Online]. Available: https://www.nist.gov/cyberframework. [Accessed August 2022].

[13] R. Gu, C. Niu, F. Wu, G. Chen, C. Hu, C. Lyu and Z. Wu, "From Server-Based to Client-Based Machine Learning: A Comprehensive Survey," *ACM Comput. Surv,* vol. 54, no. 1, pp. 6:1-6:36, 2021.

[14] M. Langer, Z. He, W. Rahayu and Y. Xue, "Distributed Training of Deep Learning Models: A Taxonomic Perspective," *IEEE Trans. Parallel Distrib. Syst.,* vol. 31, no. 12, pp. 2802-2818, 2020.

[15] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg and T. Verbelen, "A Survey on Distributed Machine Learning," *ACM Comput. Surv.,* 2020.

[16] B. Ramsundar and R. B. Zadeh, TensorFlow for deep learning: from linear regression to reinforcement learning, O'Reilly Media, 2018.

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

[17] B. McMahan, E. Moore, D. Ramage, S. Hampson and B. A. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

[18] S. Abdulrahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi and M. Guizani, "A Survey on Federated Learning: The Journey From Centralized to Distributed On-Site Learning and Beyond," *IEEE Internet Things J.,* vol. 8, no. 7, p. 5476–5497, 2021.

[19] NVIDIA, "NVIDIA Jetson: Advanced AI Embedded Systems," [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/.. [Accessed August 2022].

[20] Intel Corporation, "Intel Neural Compute Stick 2 (Intel® NCS2)," [Online]. Available: https://www.intel.com/content/www/us/en/developer/tools/neural-compute-stick/overview.html. [Accessed August 2022].

[21] H. Wang, Z. Qu, Q. Zhou, H. Zhang, B. Luo, W. Xu, S. Guo and R. Li, "A Comprehensive Survey on Training Acceleration for Large Machine Learning Models in IoT," *IEEE Internet of Things Journal,* vol. 9, no. 2, pp. 939-963, 2022.

[22] L. Kohnfelder and P. Garg, "The threats to our products," 1 April 1999. [Online]. Available: https://adam.shostack.org/microsoft/The-Threats-To-Our-Products.docx. [Accessed August 2022].

[23] Microsoft, "Threats - Microsoft Threat Modeling Tool - Azure," [Online]. Available: https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats. [Accessed August 2022].

[24] T. Uceda-Vélez and M. Morana, Intro to PASTA, Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis, Wiley, 2015.

[25] "TRIKE," [Online]. Available: https://github.com/octotrike/trike. [Accessed August 2022].

[26] P. Saitta, B. Larcom and M. Eddington, "Trike v.1 Methodology Document," 2015. [Online]. Available: https://www.octotrike.org/papers/Trike_v1_Methodology_Document-draft.pdf . [Accessed August 2022].

[27] J. D. Meier, A. Mackman, M. Dunner, S. Vasireddy, R. Escamilla and A. Murukan, "Improving web application security: Threats and countermeasures," Microsoft Corporation, 2003.

[28] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition,* no. 84, pp. 317-331, 2018.

[29] S. Aniruddha, A. Subramanya and H. Pirsiavash, "Hidden trigger backdoor attacks," in *Proceedings of the AAAI conference on artificial intelligence*, 2020.

[30] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto and F. Roli, "Evasion Attacks against Machine Learning at Test Time," in *Proceedings of the 2013th European Conference on Machine Learning and Knowledge Discovery in Databases*, 2013.

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

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

[32] B. Biggio, I. Corona, B. Nelson, B. Rubinstein, D. Maiorca, G. Fumera, G. Giacinto and F. Roli, "Security evaluation of support vector machines in adversarial environments," in *Support Vector Machines Applications*, 2014.

[33] N. Papernot, P. McDaniel, X. Wu, S. Jha and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *IEEE symposium on security and privacy (SP)*, 2016.

[34] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *25th USENIX security symposium*, 2016.

[35] M. Juuti, S. Szyller, S. Marchal and N. Asokan, "PRADA: protecting against DNN model stealing attacks," in *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2019.

[36] T. Orekondy, B. Schiele and M. Fritz, "Knockoff nets: Stealing functionality of black-box models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019.

[37] K. Krishna, G. S. Tomar, A. P. Parikh, N. Papernot and M. Iyyer, "Thieves on Sesame Street! Model Extraction of BERT-based APIs," in *International Conference on Learning Representations*, 2019.

[38] E. Javid, D. Lowd and D. Dou, "On Adversarial Examples for Character-Level Neural Machine Translation," *arXiv preprint arXiv:1806.09030,* 2018.

[39] S. Reza, M. Stronati, C. Song and V. Shmatikov, "Membership inference attacks against machine learning models," in *IEEE symposium on security and privacy (SP)*, 2017.

[40] S. Yeom, I. Giacomelli, M. Fredrikson and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *IEEE 31st computer security foundations symposium (CSF)*, 2018.

[41] N. Z. Gong and B. Liu, "Attribute inference attacks in online social networks," *ACM Transactions on Privacy and Security,* vol. 21, no. 1, pp. 1-30, 2018.

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

[43] C. Song, T. Ristenpart and V. Shmatikov, "Machine learning models that remember too much," in *ACM SIGSAC Conference on computer and communications security*, 2017.

[44] K. Kurita, P. Michel and G. Neubig, "Weight Poisoning Attacks on Pretrained Models," in *58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[45] E. Quiring and K. Rieck, "Backdooring and poisoning neural networks with image-scaling attacks," in *IEEE Security and Privacy Workshops (SPW)*, 2020.

[46] T. Elham, K. J. Burns, M. Hadjimichael, A. D. Molina-Markham and J. T. Sexton, "A taxonomy and terminology of adversarial machine learning," NIST IR, 2019.

[47] A. N. and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access,* vol. 6, pp. 14410-14430, 2018.

[48] European Union Agency for Cybersecurity, "Securing machine learning algorithms," ENISA, 2021.

[49] MITRE, "MITRE ATLAS™ (Adversarial Threat Landscape for Artificial-Intelligence Systems)," [Online]. Available: https://atlas.mitre.org/. [Accessed August 2022].

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

[50] MITRE, "MITRE ATT&CK," [Online]. Available: https://attack.mitre.org/. [Accessed August 2022].

[51] MITRE, "Evade ML Model," [Online]. Available: https://atlas.mitre.org/techniques/AML.T0015. [Accessed August 2022].

[52] MITRE, "Adversarial ML Threat Matrix github repository," [Online]. Available: https://github.com/mitre/advmlthreatmatrix. [Accessed August 2022].

[53] Microsoft Corp., "Failure Modes in Machine Learning," [Online]. Available: https://docs.microsoft.com/en-us/security/engineering/failure-modes-in-machine-learning. [Accessed August 2022].

[54] Microsoft Corp., "Threat Modeling AI/ML Systems and Dependencies," [Online]. Available: https://docs.microsoft.com/en-us/security/engineering/threat-modeling-aiml. [Accessed August 2022].

[55] WithSecure Corporation, "A security self-assessment questionnaire for machine learning-based systems," 2021. [Online]. Available: https://blog.f-secure.com/a-security-self-assessment-questionnaire-for-machine-learning-based-systems/. [Accessed August 2022].

[56] IBM, "Adversarial Robustness Toolbox: A Python library for ML Security," [Online]. Available: https://adversarial-robustness-toolbox.org/. [Accessed August 2022].

[57] Microsoft Corp., "Counterfit," [Online]. Available: https://github.com/Azure/counterfit. [Accessed August 2022].

[58] "CleverHans (latest release: v4.0.0)," [Online]. Available: https://github.com/cleverhans-lab/cleverhans. [Accessed August 2022].

[59] B. Kulynych and M. Yaghini, "MIA: A library for running membership inference attacks against ML models," [Online]. Available: https://github.com/spring-epfl/mia. [Accessed August 2022].

[60] Microsoft Corp., "AI Fairness Checklist," [Online]. Available: https://www.microsoft.com/en-us/research/project/ai-fairness-checklist/. [Accessed August 2022].

[61] T. M. J. Gebru, B. Vecchione, J. Vaughan, H. Wallach, H. Iii and K. Crawford, "Datasheets for datasets," *Communications of the ACM,* vol. 64, no. 12, pp. 86-92, 2021.

[62] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics*, 2020.

[63] B. Hitaj, G. Ateniese and F. Perez-Cruz, "Deep models under the GAN: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017.

[64] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017.

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.

[65] A. Ilyas, L. Engstrom, A. Athalye and J. Lin, "Black-box adversarial attacks with limited queries and information," in *International Conference on Machine Learning*, 2018.

[66] P. Y. Chen, H. Zhang, Y. Sharma, J. Yi and C. J. Hsieh, "ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proceedings of the 10th ACM workshop on artificial intelligence and security*, 2017.

[67] Z. Zhao, K. M. Barijough and A. Gerstlauer, "DeepThings: Distributed Adaptive Deep Learning Inference on Resource-Constrained IoT Edge Clusters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 37, no. 11, pp. 2348-2359, 2018.

SPATIAL project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101021808.