STRIDE-AI: An Approach to Identifying Vulnerabilities of Machine Learning Assets

Lara Mauri
DI – Università degli Studi di Milano
Milan, Italy
lara.mauri@unimi.it

Ernesto Damiani
C2PS – Khalifa University
Abu Dhabi, UAE
ernesto.damiani@ku.ac.ae

***Abstract*—We propose a security methodology for Machine Learning (ML) pipelines, supporting the definition of key security properties of ML assets, the identification of threats to them as well as the selection, test and verification of security controls. Our proposal is based on STRIDE, a widely used approach to threat modeling originally developed by Microsoft. We adapt STRIDE to the Artificial Intelligence domain by taking a *security property-driven* approach that also provides guidance in selecting the security controls needed to alleviate the identified threats. Our proposal is illustrated via an industrial case study.**

***Index Terms*—Artificial Intelligence security, Threat modeling, Vulnerability assessment.**

I. Introduction

Machine Learning (ML) is playing a major role in a wide range of application domains. However, when ML models are deployed in production, they can be fooled and misled in ways that can have profound security implications. The call to improve the security of Artificial Intelligence (AI) systems [1] has attracted widespread attention from the ML community, which has given rise to a new vibrant line of research in security and privacy of ML models and related applications. The ML literature is now plentiful with research papers addressing the security of ML models, including several valuable survey papers discussing individual vulnerabilities and possible defensive techniques. Existing work mainly focuses on ingenious mechanisms that allow attackers to compromise the ML-based systems at the two core phases of the learning process, that is the training and the inference stages.

In this work, we investigate the distinct, though related, problem of defining a practical methodology for assessing ML-based systems' security. We approach the problem from the point of view of the security practitioner who has to deal with ML-based systems rather than from the one of the AI expert dealing with security. We argue that, given the current variety and scope of threats and attacks to ML models, there is some confusion about what exactly the security analyst is expected to do to alleviate them. The goal of this paper is to propose an *asset-centered* methodology for identifying threats to ML-based systems. Our proposal is based on STRIDE [2], a well-known and widely used approach to threat modeling originally developed by Microsoft. STRIDE has been identified by Microsoft itself and by independent agencies like the European Union Agency for Cybersecurity (ENISA) as a promising starting point for AI threat modeling. We argue that our extension to the original STRIDE provides an *ML-specific*, *security property-driven* approach to threat detection which can also provide guidance in selecting the security controls needed to alleviate the identified threats.

The rest of the paper is structured as follows. Section II offers a quick overview of relevant research in AI security. Section III illustrates the reference ML life-cycle and presents the key ML data assets together with the *failure modes* specific for each of them. Section IV describes our STRIDE-AI methodology to identifying threats to ML data assets, while Section V applies it to a real use case selected from the AI-ML applications developed in the TOREADOR H2020 project. Finally, Section VI draws our conclusions.

II. Related Work

A. Secure Machine Learning

Many research papers have investigated the security issues of ML so far, so we will start from surveys. Barreno and Huang [3]–[5] were among the first to systematically survey the literature on attacks against ML systems. Subsequent work has addressed attacks and defence strategies with varying degrees of breadth and depth [6]–[9]. An overview of the evolution of this research area over the last few years can be found in [10]. Recently, the US National Institute of Standards and Technology (NIST) has published a taxonomy of adversarial machine learning complemented by a terminology of related key concepts [11]. The NIST document is aimed at establishing a common language for future standardization. Also the ISO/IEC JTC 1/SC 42 international standard committee's work includes several topics in the AI security area. The report ISO/IEC TR 24028:2020 [12], for instance, analyzes some threats that may contribute to the erosion of trust in AI systems and briefly discusses approaches to managing them. In parallel with the above efforts, research investigated techniques for preventing undesired behavior of ML models by controlling *a priori* their training sets and parameters [13]–[19]. For example, Huang et al. [17] have proposed an automated verification framework for checking safety of feed-forward deep neural networks based on exploration of regions around data points of interest to search for specific adversarial

978-1-7281-5684-2/20/$31.00 ©2021 IEEE

manipulations. The focal point of the work by Raghunathan et al. [14] was a method for producing *certificates of robustness* for two-layer neural networks, which can be optimized at training time jointly with the network itself. The recent work by our group [19] proposed a framework suitable for practical certification of distributed ML-powered applications based on statistical monitoring of ML models' behavior at inference time.

B. Threat Modeling for Artificial Intelligence

We argue that threat modeling methodologies should play an important role in the security analysis of AI systems by supporting the security expert's understanding of how ML-based systems fail. Until now, only a few studies have taken this perspective for the identification of ML security risks. Researches that have points of connection with ours are presented in [20]–[23]. The recent *AI Threat Landscape* [20], released by the European Union Agency for Cybersecurity (ENISA), sets a baseline for a common understanding on cybersecurity threats to AI. The ENISA report identifies a list of assets within the AI ecosystem and maps threats against them taking into account the stages of the typical AI life-cycle. The *Architectural Risk Analysis* (ARA) [21] proposed by the Berryville Institute of Machine Learning (BIML) features a design-level view. ARA identifies 78 specific risks associated with a generic ML-based system and organizes them into a "top-ten" list. Further efforts to understand how ML-based systems may fail and, consequently, how to respond to such failures have been made by Microsoft through the release of guidelines for the mitigation and triage of AI-specific security threats [22]. Microsoft's approach, which develops, like we do, on STRIDE threat-modeling [2], does not address the security properties definition. Rather, it is based on a taxonomy that classifies ML failure modes into two categories, namely *intentional* and *unintentional* failures. Intentional failures are mapped to a list of attacks reported in the literature [24]. Other recent works explore "turn-key" applicability to ML-based systems of security assessment methods used in the field of software engineering. Wilhjelm and Younis [23] decompose ML models into three components, namely input, processing and output, and discuss how to identify threats based on their attack vectors. In order to rank the impact of the identified risks, the authors make use of a *bug bar* [25] for associating severity levels to threats. Our own recent work [26] proposes such a metric based on the notion of a "gold standard" data set for assessing ML models' degradation.

III. ML-based Application Life-cycle

Although there exist many diverse types of learning tasks [27], the development process of ML-based systems has an intrinsic iterative multi-stage nature [28]. Fig. 1 shows our reference *ML life-cycle*, starting from requirements analysis and ending with the ML model's maintenance in response to changes. While this life-cycle does not cover all possible developments, we will use it to identify the key data assets produced at each phase and to analyze their failure modes. We start by outlining the activities carried out at each stage.

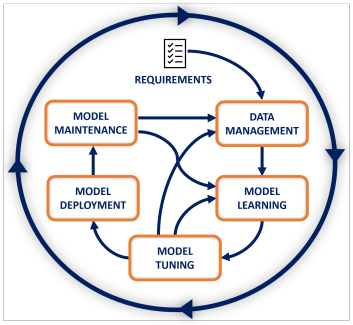

Fig. 1. Our reference ML life-cycle.

The initial stage of the ML life-cycle, *Data Management*, includes a number of steps, a major one being the *ingestion* of the data required for the next stages. Ingestion occurs from multiple sources, and the data collected can either be stored or immediately used. Pre-processing techniques are then used to create a consistent data set suitable for training, testing and evaluation. The next step, *Model Learning*, involves selecting an ML model that handles the task of interest. Depending on the goals and the amount of knowledge available to the learner, different ML techniques can be used, such as supervised, unsupervised and reinforcement learning. In the training process of a supervised ML-based system, a learning algorithm is provided with predefined inputs and known outputs that are used for model training. The learning algorithm computes *error metrics* to determine whether the model is learning well, i.e. it delivers the expected output not only on the inputs it has seen in training, but also on test data it has never seen. The so-called *hyper-parameters*, which control how the training is done (e.g., how the error is used to modify the ML model's internal parameters), are set during the *Model Tuning* stage. While being tuned, the ML model is also validated to determine whether it works properly on inputs collected independently from the original training and test sets. The transition from development to production is handled in the *Model Deployment* stage. In this stage, the model executes *inferences* on real inputs, generating the corresponding results. As the production data landscape may change over time, in-production ML models require continuous monitoring. The final ML life-cycle stage, *Model Maintenance*, monitors the ML model and retrains it when needed.

A. AI Assets

At each stage of the ML life-cycle, multiple *digital assets* are generated and used. Before performing threat assessment, we need to identify the assets the threats may apply to. At-risk assets can be grouped into six different categories, as shown in Fig. 2. For the sake of conciseness, in this paper we mostly focus on the data assets.[1] Table I shows the assets' *failure*

[1]A complete asset list as well as documentation used for the case study (Section V) is available at https://github.com/LaraMauri/STRIDE-AI

modes, i.e. how they may fail to show the properties needed for a correct execution of the corresponding stage.

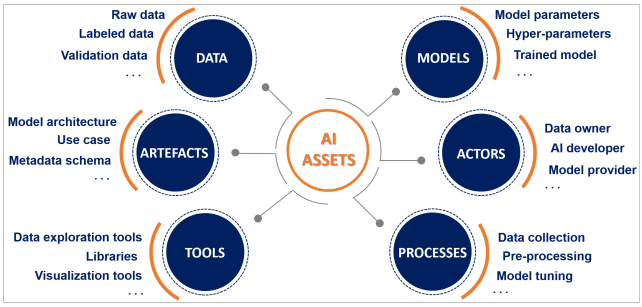

Fig. 2. Assets in the AI ecosystem.

IV. Modeling Threats to ML Data Assets

We now discuss how assets' failure modes can be used to identify threats. Classically, *Threat Modeling* (TM) is the process of reviewing the security of a system, identifying critical areas, and assessing the risk associated with them. TM is a fundamental phase in the design of any information system because it allows profiling and prioritizing problems as well as assessing the value that potential mitigation would have in alleviating threats. A typical TM process [31] consisting of five steps is shown in Table II.

Several TM methods are available. Popular approaches include PASTA [32], a risk-centered TM framework consisting of seven stages within which different elicitation tools are used, and OCTAVE [33], which is a three-phase method focusing on the assessment of the organizational risks rather than the technological ones. Originally defined by Loren Kohnfelder and Praerit Garg [34], [35], STRIDE is the most mature one. It has been applied to many vertical domains, including cyber-physical systems and healthcare applications [36]–[40]. STRIDE uses a set of six threats based on its acronym, which stands for *Spoofing, Tampering, Repudiation, Information Disclosure, Denial of service*, and *Elevation of privilege*; Table III shows their definitions.

Our discussion of the ML life-cycle and its key data assets (Section III) has covered steps 2 and 3 of the TM process. We need to set the security objectives (step 1), and then proceed with the threat identification step (step 4) and the vulnerability identification one (step 5).

We start by proposing our ML-specific definitions of the classic $CIA^3 - R$ security properties (step 1). Then, we will discuss how assets' failure modes can jeopardize these properties. Finally, we will use the results of our analysis for identifying threats to the ML life-cycle (step 4).

A. Extending STRIDE to ML-based Systems

The $CIA^3 - R$ hexagon [41] includes the six main components of security. Each STRIDE threat corresponds to the violation of a $CIA^3 - R$ security property, as shown in Table IV.

While STRIDE can be directly applied to vertical domains, we argue that applying it to AI-ML requires customization of the desired set of properties in order to match the domain assets' specific failure modes [42]. Below, we propose our ML-specific definition of $CIA^3 - R$ security properties. Then, we map the failure modes of the data assets generated in our ML life-cycle (Section III) to the violation of one or more of these properties. For each mapping, we associate (the effects of) the corresponding failure to one or more STRIDE threats to ML-based systems. Table V shows our proposed definition of an ML-specific $CIA^3 - R$ hexagon. The above definitions are hopefully self-explanatory; for the sake of conciseness, we will only elaborate on our definition of the confidentiality property [43] by considering a classification problem: mapping the items of a data space DS to categories of interest belonging to a set $C = (C_1, ..C_n)$. A representative sample $S \subsetneq DS$ is used to tabulate a partial classification function $f : S \to C$, obtaining a labeled *training set*, which by abuse of notation we shall also call f. We use the training set f to *train* an ML model that will be able to compute another function $F : DS \to C$. Finally, we deploy F into production, using it to classify individuals from DS as needed.

This standard procedure may disclose ML data, for example if the entries in f can be inferred from F.[2] Our definition of confidentiality expresses the property as an achievable goal [44]: observing the execution of a confidential ML model F, one should be able to infer the same information about an entry $e \in f$ as by observing F', obtained using the training set $f - \{e\} + \{r\}$, where r is a random entry. The same holds for validation and augmented data assets.

The resulting threat identification is summarized in Table VI, where we also report some known attacks exploiting vulnerabilities to STRIDE-AI threats.

B. Threats Prioritization

Threat-ranking techniques are used to associate a security risk level to each threat. A popular technique is represented by the so-called *bug bars*, which come in the form of tables listing the criteria used to classify bugs. Recently, Microsoft has released a bug bar [25] to rank ML threats, focusing on intentional malicious behavior against ML-based systems. However, threat prioritisation bug bars are not always easy to explain to non-security-savvy users. One of the first developed methods to assess the severity of threats is DREAD, an acronym referring to five categories (*Damage Potential*, *Reproducibility*, *Exploitability*, *Affected Users* and *Discoverability*). DREAD, which was designed to complement STRIDE, assigns to each threat a value from 1 to 10. As it turned out that it can lead to inconsistent results due to the intrinsic subjectivity of the rating process [52], the DREAD scaled rating system is no longer recommended for exclusive use; yet, it is still used for

[2]For instance, if F is computed using the Nearest-Neighbor technique (i.e. $\forall x \in DS, F(x) = f(p_x)$ where p_x is the point in S closest to x according to some domain distance), f is integral part of the definition of F and is therefore fully disclosed to the external service whenever F is deployed.

TABLE I
ML DATA ASSETS AND THEIR FAILURE MODES.

ML data asset	Failure mode
Functional requirements model the domain of interest, the problem to be solved, and the task to be executed by the ML model. *Non-functional requirements* identify architectural (hardware) and code (software) needs.	Requirements may fail when they are built in isolation from the circumstances that make the ML model necessary. Specifically, functional requirements about the ML model's accuracy may fail by not taking into account the adverse effect of non-functional properties mandated by regulations and by not considering the severity of information leaks.
Raw Data refers to any type of information gathered at the Data Management stage, before it is transformed or analyzed in any way.	Raw data assets fail when they are not sufficiently representative of the domain or unfit for the ML model business goal (e.g. due to sample size and population characteristics). Data size does not always guarantee it is representative of a domain. If data selection is biased towards some elements that have similar characteristics (a phenomenon called *selection bias*) then, even a large data set will not be representative enough. Assessing whether data is representative enough cannot be done *a priori*; it is only possible after having identified the purpose for collecting the data. Selection bias has been described in the scientific literature as due to malpractice [29].
Labeled Data refers to sets of scalar or multi-dimensional data items used at the Model Learning stage. This data is tagged with informative labels, for the purpose of training a supervised ML model.	Labeled data sets fail when enough data items are deleted or omitted, a sufficient number of spurious labeled data is included into the data set, or enough labels are modified. When the labeled data set is used for the purpose of training an ML model, all such modifications affect the model inference (e.g., shifting the model's classification boundary).
Validation Data is also used at the Model Learning stage, but differs from ordinary labeled data in usage and, usually, in collection circumstances. Validation data sets are mostly used to perform an evaluation of the ML model in-training, e.g. by stopping training (*early stopping*) when the error on the validation set increases too much [30], as this is considered a sign of *over-fitting*.	Validation data sets fail when their labeled data items are modified. Modification of validation data affects how the error computed on the validation data set fluctuates during training, and even a single modification on the validation set may be enough for introducing a spurious error increase that could cut short the training. Elimination of outliers in the validation data set may alleviate/prevent failure.
Augmented Data is labeled data that is complemented at the Model Tuning stage by additional data produced by transformations or by generative ML models. Augmentation increases labeled data sets' diversity, which is supposed to prevent over-fitting.	Augmented data sets may fail due to inconsistency with the training set they are derived from. Heuristic data augmentation schemes are often tuned manually by humans, and defective augmentation policies may cause ML models to lose rather than gain accuracy from the augmented data.
Held-out Test Cases (HTCs) are inputs used to test ML models in production, i.e. in the Model Maintenance stage. HTCs include special inputs of high interest for the application.	The rationale for HTCs is that even if an ML model keeps showing good accuracy, its performance on specific inputs may become unacceptable. HTCs fail when the ML model's accuracy metrics computed on them does not correspond to the business goals of the application. Careless selection of HTCs has been known to trigger unneeded model retraining.
Inferences are results computed by ML models based on real inputs, according to the task of interest in the Model Deployment and Model Maintenance stages.	Inferences may fail by showing high entropy, i.e. conveying little information useful for the ML task at hand.

TABLE II
A 5-STEP TM PROCESS.

Step		Description
1	Objectives Identification	*States the security properties the system should have.*
2	Survey	*Determines the system's assets, their interconnections and connections to outside systems.*
3	Decomposition	*Selects the assets that are relevant for the security analysis.*
4	Threat Identification	*Enumerates threats to the system's components and assets that may cause it to fail to achieve the security objectives.*
5	Vulnerabilities Identifications	*Examines identified threats and determines if known attacks show that the overall system is vulnerable to them.*

TABLE III
STRIDE THREATS IN A NUTSHELL.

Threat	Description
Spoofing Identity	*A user takes on the identity of another. For example, an attacker takes on the identity of an administrator.*
Tampering with Data	*Information in the system is modified by an attacker. For example, an attacker changes a data item.*
Repudiation	*Information about a transaction is deleted in order to deny it ever took place. For example, an attacker deletes a login transaction to deny he ever accessed an asset.*
Information Disclosure	*Sensitive information is stolen and sold for profit. For example, information on user behavior is stolen and sold to a competitor.*
Denial of Service (DoS)	*This involves exhausting resources required to offer services. For example, in a DoS against a data flow the attacker consumes network resources.*
Elevation of Privilege (EoP)	*This is a threat similar to spoofing, but instead of taking on the ID of another, the attacker elevates his own security level to an administrator.*

quick preliminary threat assessments, as we will do for the case study described in the next section.

V. USE CASE

We will now apply STRIDE-AI to a real use case selected from the AI-ML applications developed in *TOREADOR* H2020 project[3]. The cyber-security of such applications is a goal of another ongoing H2020 project, THREAT-ARREST[4]. We focus on a scenario contributed by the Light-source company (henceforth LIGHT), one of the major European renewable energy providers, who is a partner in both projects. In this use case, AI-ML techniques are employed for failure prediction and power flow optimization on the energy grid. The use case architecture is shown in Fig. 3 and its operation is described by the following steps:

1) Sensors placed at power station sites send data about power and other variables to a local Data Logger card equipped for network communication. The Data Logger

[3]https://cordis.europa.eu/project/id/688797
[4]https://cordis.europa.eu/project/id/786890

TABLE IV
THREATS VS. $CIA^3 - R$ PROPERTIES IN STRIDE-AI.

Property	Threat
Authenticity	Spoofing
Integrity	Tampering
Non-repudiability	Repudiation
Confidentiality	Information Disclosure
Availability	Denial-of-Service (DoS)
Authorization	Elevation-of-Privilege (EoP)

TABLE V
ML-SPECIFIC $CIA^3 - R$ HEXAGON.

Property	ML-specific definition
Authenticity	*The output value delivered by a model has been verifiably generated by it.*
Integrity	*Information used or generated throughout a model's life-cycle cannot be changed or added to by unauthorized third parties.*
Non-repudiation	*There is no way to deny that a model's output has been generated by it.*
Confidentiality	*Using a model to perform an inference exposes no information but the model's input and output.*
Availability	*When presented with inputs, the model computes useful outputs, clearly distinguishable from random noise.*
Authorization	*Only authorized parties can present inputs to the model and receive the corresponding outputs.*

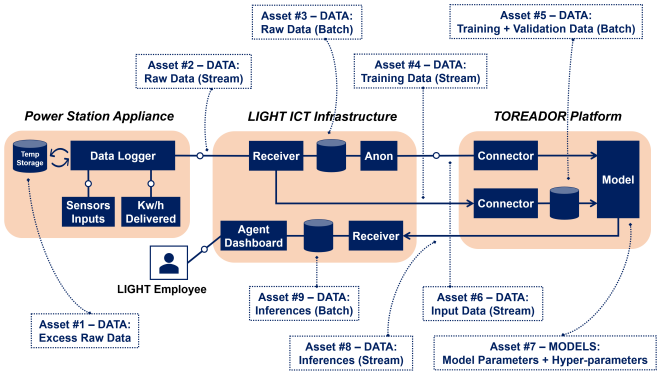

Fig. 3. The Predictive Maintenance Architecture.

has a local root-of-trust.

2) The Data Logger card forwards the data to a Receiver within the LIGHT ICT infrastructure.
3) The Anonymity (Anon) agent retrieves the data and applies some pre-processing, including metadata removal. Removal is not performed for training data, whose metadata are dummy.
4) Pre-processed input data is streamed to a Connector on the TOREADOR platform.
5) Input data is stored in the TOREADOR platform, then fed to an ML model for training/validation or failure prediction.
6) In production, inference (Raw Prediction Data) is returned to the LIGHT platform (predictions are not returned during training).
7) LIGHT employees log into a Dashboard on the LIGHT platform and drill into the inferences.
8) Raw Prediction data is used for further action (including pro-actively maintaining components at the power station).

With reference to the TM process in Table II, a security expert applying the STRIDE-AI methodology has a ready-made set of objectives (the $CIA^3 - R$ hexagon) and assets categories (the ones on Fig. 2). The expert starts by performing architecture decomposition and identifying the relevant assets, as shown by the annotations in Fig. 3. For each asset, the expert chooses a set of properties of interest within $CIA^3 - R$. For the sake of conciseness, let us consider only one of the identified data assets, the *training data stream* (Asset #4). The complete mapping of all identified assets, properties and threats is provided in the Appendix (Table VIII).

The training data stream asset belongs to the *Labeled Data* asset category; therefore, it should exhibit the Integrity and Authenticity properties to prevent known failures (Table VI).[5] The third step is *threat identification*. Again from Table VI, the security expert identifies *Tampering* and *Spoofing* as the threats corresponding to the assets' property profile.

The next step of the TM process is *vulnerabilities identification*, where the security expert interacts with the system developers to describe under which conditions the threats associated to the assets can materialize. For Asset #4, the *Tampering* threat corresponds to attackers making changes or injecting spurious data in the sensor data stream, while the *Spoofing* threat corresponds to attackers posing as the TOREADOR ML model to the LIGHT data platform, and as the LIGHT platform to the TOREADOR model. The expert needs to assess whether the threats are *atomic* or *composite*, i.e. they require one or more conditions to hold for being exploited. The (simplified) *threat trees* for the *Spoofing* and *Tampering* threats are shown in Figures 4 and 5, respectively.

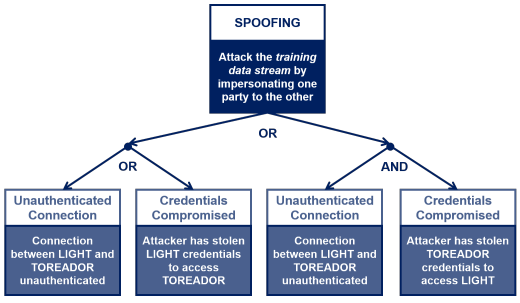

Fig. 4. The Spoofing threat tree.

The *Spoofing* threat tree corresponds to a standard failure mode of distributed platforms. The *Tampering* threat tree is

[5]Other asset categories will have different property profiles. For example, the library used for training the ML model will need Integrity and Authenticity (to ensure its code has not been tampered with) and Availability (to make sure it will be capable of performing the training or inference task within the deadline).

TABLE VI
MAPPING DATA ASSETS' FAILURE MODES TO $CIA^3 - R$ HEXAGON.

Asset	Properties	Threats	Known attacks
Requirements	Availability	DoS	*While no direct attacks to requirements have been reported, unexpected legal liabilities deriving from defective requirements have been described in a number of concrete cases [45], including ML models for medical diagnostics.*
Raw Data	Authenticity, Confidentiality, Availability, Authorization	Spoofing, Disclosure, DoS, EoP	*Attacks by data owners introduce selection bias on purpose when publishing raw data in order to affect inference to be drawn on the data. Reported examples [46] include companies who release biased raw data with the hope competitors would use it to train ML models, causing competitors to diminish the quality of their own products and consumer confidence in them. In perturbation-style attacks, the attacker stealthily modifies raw data to get a desired response from a production-deployed model [24]. This compromises the model's classification accuracy.*
Labeled Data	Authenticity, Integrity	Spoofing, Tampering	*Append attacks target availability by adding random samples to the training set to the point of preventing any model trained on that data set from computing any meaningful inference. Other modifications to the training data set (backdoor or insert attacks) jeopardize the ML model's integrity by trying to introduce spurious inferences [47]. Attackers randomly draw new labels for a part of the training pool to add an invisible watermark that can later be used to "backdoor" into the model.*
Augmented Data	Integrity, Availability	Tampering, DoS	*Adversarial data items tailored to compromise ML model inference can be inserted during data augmentation [48] in order to make them difficult to detect.*
Validation Data	Integrity, Availability	Tampering, DoS	*Attacks can shorten the training of the ML model by compromising just a small fraction of the validation data set. "Adversarial" training data generated by these attacks are quite different from genuine training set data [49].*
Held-Out Test Cases	Integrity, Confidentiality, Availability	Tampering, Disclosure, DoS	*Evaluating an ML model's performance on HTCs involves reducing all of the information contained in the HTCs outputs to a single number expressing accuracy. The literature reports* slicing attacks *[50], which poison the held-out data set to produce misleading results. Slicing attacks introduce specific slices of data that doctor the model's accuracy, making it very different from how it performs on the in-production data set.*
Inferences	Authenticity, Integrity, Availability, Authorization	Spoofing, Tampering, DoS, EoP	*Inferences need to carry informative content. The literature reports* eavesdropping attacks *(a survey can be found in [51]) to distributed ML models involving eavesdropping on inferences.*

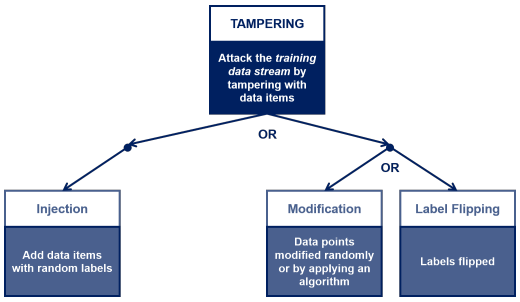

Fig. 5. The Tampering threat tree.

more ML-specific, as it affects the training data stream by injecting spurious data items (i.e., adding data that do not come from LIGHT, with random or chosen labels) or by modifying data items that come from LIGHT by flipping the labels. The difference between the *Tampering* threat sub-trees (Fig. 5) is relevant for the expert's assessment, as the leftmost sub-tree can be deleted as the expert knows that the data items are signed by the Data Logger cards, but not the rightmost one: Loggers' signature would not prevent label flipping on training data, as labels are added on the LIGHT platform before streaming the training set to TOREADOR.

TABLE VII
DREAD RATING FOR THE SPOOFING AND TAMPERING THREATS.

Threats	D	R	E	A	D	Average [Rating]
Spoofing	5	2	4	2	7	4.0 [Medium Risk]
Tampering	7	2	6	2	5	4.4 [Medium Risk]

We can now compute the relative priorities of the *Spoofing* and *Tampering* threats by utilizing a DREAD scorecard, whose scores are shown in Table VII. For the training data stream asset, the potential damage that could result from a spoofing attack includes backdoor generation and substitution of data about power emission or consumption, which could alter metrics and reporting. The expertise required to perform spoofing varies depending on which branch of the *Spoofing* threat tree is considered. As the TOREADOR platform supports a two-factor authentication, in case the LIGHT users' credentials consist only of a username and password, it may prove easier to gain access to TOREADOR by password cracking or by stealing the credentials (e.g, via phishing emails). An attack exploiting the *Spoofing* threat may result in tampering. If an attacker is able to tamper with the training data stream, she can perform label flipping or other types of data manipulation. The exploitation of this threat depends on the presence of constraints on data manipulation; as labels are added on the LIGHT platform, while data points are collected and signed by Data Loggers, adding new data with random labels requires more effort than flipping labels of existing data to obtain a particular output in the presence of certain input values.

VI. DISCUSSION AND CONCLUSIONS

In this paper, we outlined STRIDE-AI, an approach to identifying threats to ML models. Our methodology was illustrated with the help of a use case. We are well aware that no threat identification method is effective without providing also guidance in selecting the security controls needed to alleviate the identified threats. Unfortunately, no security control framework specifically designed for ML models is

available. While security control selection is outside the scope of this paper, we contributed to addressing this point by putting forward the idea of a trusted environment for ML models training and operation [53]. We argue that *Distributed Ledger Technologies* (DLTs) can provide a complete security control framework for ML [54]. DLT can support achieving CIA^3-R properties, making interfering with ML inference results less attractive for attackers.

ACKNOWLEDGMENT

This research was supported in part by the EU-funded project THREAT-ARREST (grant agreement No. H2020-786890) and in part by the Northrop Grumman Master Agreement fund provided for the C2PS for the project "Customization of Cyber-Physical Systems Testing" (No. 8434000041).

ORCID

Lara Mauri: https://orcid.org/0000-0002-4024-1015
Ernesto Damiani: https://orcid.org/0000-0002-9557-6496

APPENDIX

TABLE VIII
COMPLETE MAPPING FOR THE ASSETS IDENTIFIED IN THE USE CASE.

Use case asset	**Properties**	**Threats**
#1. Excess Raw Data	Authenticity, Integrity	Spoofing, Tampering
#2. Raw Data (Stream)	Authenticity, Integrity	Spoofing, Tampering
#3. Raw Data (Batch)	Authenticity, Integrity, Authorization	Spoofing, Tampering, EoP
#4. Training Data (Stream)	Authenticity, Integrity	Spoofing, Tampering
#5. Training + Validation Data (Stream)	Authenticity, Integrity, Non-repudiability, Authorization	Spoofing, Tampering, Repudiation, EoP
#6. Input Data (Stream)	Authenticity, Integrity, Non-repudiability	Spoofing, Tampering, Repudiation
#7. Model Parameters + Hyper-parameters	Integrity, Non-repudiability	Tampering, Repudiation
#8. Inferences (Stream)	Authenticity, Integrity, Non-repudiability, Availability	Spoofing, Tampering, Repudiation, DoS
#9. Inferences (Batch)	Authenticity, Integrity, Availability, Authorization	Spoofing, Tampering, Dos, EoP

REFERENCES

[1] T. G. Dietterich, "Steps Toward Robust Artificial Intelligence," *AI Magazine*, vol. 38, no. 3, pp. 3–24, Oct. 2017.

[2] S. Hernan, S. Lambert, T. Ostwald, and A. Shostack, "Threat Modeling - Uncover Security Design Flaws Using The STRIDE Approach," *MSDN Magazine*, pp. 68–75, 2006.

[3] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can Machine Learning Be Secure?" in *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 16–25.

[4] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," *Mach. Learn.*, vol. 81, no. 2, pp. 121–148, 2010.

[5] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial Machine Learning," in *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, ser. AISec '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 43–58.

[6] N. Papernot, P. D. McDaniel, A. Sinha, and M. P. Wellman, "SoK: Security and Privacy in Machine Learning," in *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018*. IEEE, 2018, pp. 399–414.

[7] N. Papernot, "A Marauder's Map of Security and Privacy in Machine Learning: An overview of current and future research directions for making machine learning secure and private," in *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security, CCS 2018, Toronto, ON, Canada, October 19, 2018*. ACM, 2018, p. 1.

[8] L. Muñoz-González and E. C. Lupu, *The Security of Machine Learning Systems*. Cham: Springer International Publishing, 2019, pp. 47–79.

[9] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2805–2824, 2019.

[10] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognit.*, vol. 84, pp. 317–331, 2018.

[11] E. Tabassi, K. J. Burns, M. Hadjimichael, A. D. Molina-Markham, and J. T. Sexton, "A Taxonomy and Terminology of Adversarial Machine Learning," US Department of Commerce, National Institute of Standards and Technology - draft NISTIR 8269, 2019. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8269-draft.pdf

[12] "Technical Report ISO/IEC TR 24028:2020 Information technology — Artificial intelligence — Overview of trustworthiness in artificial intelligence," 2020.

[13] J. Steinhardt, P. W. Koh, and P. Liang, "Certified Defenses for Data Poisoning Attacks," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 3520–3532.

[14] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified Defenses against Adversarial Examples," *CoRR*, vol. abs/1801.09344, 2018. [Online]. Available: http://arxiv.org/abs/1801.09344

[15] E. Wong and Z. Kolter, "Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope," ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 5286–5295. [Online]. Available: http://proceedings.mlr.press/v80/wong18a.html

[16] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks," in *Computer Aided Verification*. Cham: Springer International Publishing, 2017, pp. 97–117.

[17] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety Verification of Deep Neural Networks," in *Computer Aided Verification*, R. Majumdar and V. Kunčak, Eds. Cham: Springer International Publishing, 2017, pp. 3–29.

[18] N. Carlini, G. Katz, C. W. Barrett, and D. L. Dill, "Ground-Truth Adversarial Examples," *CoRR*, vol. abs/1709.10207, 2017. [Online]. Available: http://arxiv.org/abs/1709.10207

[19] E. Damiani and C. A. Ardagna, "Certified Machine-Learning Models," in *SOFSEM 2020: Theory and Practice of Computer Science*. Cham: Springer International Publishing, 2020, pp. 3–15.

[20] ENISA, "AI Cybersecurity Challenges – Threat Landscape for Artificial Intelligence," December 2020. [Online]. Available: https://www.enisa.europa.eu/publications/artificial-intelligence-cybersecurity-challenges

[21] G. McGraw, H. Figueroa, V. Shepardson, and R. Bonett, "An Architectural Risk Analysis of Machine Learning Systems," Berryville Institute of Machine Learning (BIML), 2020. [Online]. Available: https://berryvilleiml.com/interactive/

[22] A. Marshall, J. Parikh, E. Kiciman, and R. S. S. Kumar, "Threat Modeling AI/ML Systems and Dependencies," 2019. [Online]. Available: https://docs.microsoft.com/en-us/security/engineering/threat-modeling-aiml

[23] C. Wilhjelm and A. A. Younis, "A Threat Analysis Methodology for Security Requirements Elicitation in Machine Learning Based Systems," in *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE Computer Society, 2020.

[24] R. S. S. Kumar, D. R. O'Brien, K. Albert, S. Viljöen, and J. Snover, "Failure Modes in Machine Learning Systems," *CoRR*, vol. abs/1911.11034, 2019. [Online]. Available: http://arxiv.org/abs/1911.11034

[25] A. Marshall, J. Parikh, E. Kiciman, and R. S. S. Kumar, "AI/ML Pivots to the Security Development Lifecycle Bug Bar," 2019. [Online]. Available: https://docs.microsoft.com/it-it/security/engineering/bug-bar-aiml

[26] L. Mauri and E. Damiani, "Estimating Degradation of Machine Learning Data Assets," *ACM Journal of Data and Information Quality*, vol. 1, no. 1, 2021.

[27] M. de Prado, J. Su, R. Dahyot, R. Saeed, L. Keller, and N. Vállez, "AI Pipeline - bringing AI to you. End-to-end integration of data, algorithms and deployment tools," *CoRR*, vol. abs/1901.05049, 2019. [Online]. Available: http://arxiv.org/abs/1901.05049

[28] E. Damiani and F. Frati, "Towards Conceptual Models for Machine Learning Computations," in *Conceptual Modeling - 37th International Conference, ER 2018, Xi'an, China, October 22-25, 2018, Proceedings*, ser. Lecture Notes in Computer Science, vol. 11157. Springer, 2018, pp. 3–9.

[29] J. Abah, "The Quest for Statistical Significance: Ignorance, Bias and Malpractice of Research Practitioners," *International Journal of Research*, vol. 5, pp. 112–129, 2018.

[30] L. Prechelt, "Early Stopping - But When?" in *Neural Networks: Tricks of the Trade - Second Edition*, ser. Lecture Notes in Computer Science, G. Montavon, G. B. Orr, and K. Müller, Eds. Springer, 2012, vol. 7700, pp. 53–67.

[31] S. Myagmar, A. J. Lee, and W. Yurcik, "Threat modeling as a basis for security requirements," in *in Proceedings of the IEEE Symposium on Requirements Engineering for Information Security*, 2005.

[32] T. UcedaVelez and M. M. Morana, *Risk centric threat modeling*. Wiley Online Library, 2015.

[33] E. A. Oladimeji, S. Supakkul, and L. Chung, "Security threat modeling and analysis: A goal-oriented approach," in *ICSE 2006*, 2006.

[34] A. Shostack, "Experiences Threat Modeling at Microsoft," in *MODSEC@MoDELS*, ser. CEUR Workshop Proceedings, vol. 413. CEUR-WS.org, 2008. [Online]. Available: http://ceur-ws.org/Vol-413/paper12.pdf

[35] ——, *Threat Modeling: Designing for Security*. Wiley, 2014.

[36] G. Martins, S. Bhatia, X. Koutsoukos, K. Stouffer, C. Tang, and R. Candell, "Towards a Systematic Threat Modeling Approach for Cyber-physical Systems," in *2015 Resilience Week (RWS)*, 2015, pp. 1–6.

[37] R. Khan, K. McLaughlin, D. Laverty, and S. Sezer, "Stride-based Threat Modeling for Cyber-Physical Systems," in *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, 2017, pp. 1–6.

[38] M. Cagnazzo, M. Hertlein, T. Holz, and N. Pohlmann, "Threat Modeling for Mobile Health Systems," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2018, pp. 314–319.

[39] V. E. Urias, B. Van Leeuwen, W. M. S. Stout, and H. Lin, "Applying a Threat Model to Cloud Computing," in *2018 International Carnahan Conference on Security Technology (ICCST)*, 2018, pp. 1–5.

[40] B. Jelacic, D. Rosic, I. Lendak, M. Stanojevic, and S. Stoja, "STRIDE to a Secure Smart Grid in a Hybrid Cloud," in *Computer Security*. Springer International Publishing, 2018, pp. 77–90.

[41] R. Hasan, S. Myagmar, A. J. Lee, and W. Yurcik, "Toward a threat model for storage systems," in *Proceedings of the 2005 ACM Workshop On Storage Security And Survivability, StorageSS 2005, Fairfax, VA, USA, November 11, 2005*. ACM, 2005, pp. 94–102.

[42] M. Abomhara, G. Køien, and M. Gerdes, "A STRIDE-Based Threat Model for Telehealth Systems," in *NISK Journal*, 2015, pp. 82–96.

[43] S. Cimato and E. Damiani, *Some Ideas on Privacy-Aware Data Analytics in the Internet-of-Everything*. Cham: Springer International Publishing, 2018, pp. 113–124.

[44] C. Dwork, "Differential Privacy," in *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, ser. Lecture Notes in Computer Science, vol. 4052. Springer, 2006, pp. 1–12.

[45] J. S. Allain, "From Jeopardy! to Jaundice: The Medical Liability Implications of Dr. Watson and Other Artificial Intelligence Systems," *Louisiana Law Review*, vol. 73, 2013.

[46] D. Yeung, "When AI Misjudgment Is Not an Accident," 2018. [Online]. Available: https://blogs.scientificamerican.com/observations/when-ai-misjudgment-is-not-an-accident/

[47] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning," *CoRR*, vol. abs/1712.05526, 2017. [Online]. Available: http://arxiv.org/abs/1712.05526

[48] H. Eghbal-zadeh, K. Koutini, P. Primus, V. Haunschmid, M. Lewandowski, W. Zellinger, B. A. Moser, and G. Widmer, "On Data Augmentation and Adversarial Risk: An Empirical Analysis," *CoRR*, vol. abs/2007.02650, 2020. [Online]. Available: https://arxiv.org/abs/2007.02650

[49] A. Paudice, L. Muñoz-González, A. György, and E. C. Lupu, "Detection of Adversarial Training Examples in Poisoning Attacks through Anomaly Detection," *CoRR*, vol. abs/1802.03041, 2018. [Online]. Available: http://arxiv.org/abs/1802.03041

[50] E. Ameisen, *Building Machine Learning Powered Applications: Going from Idea to Product*. O'Reilly, 2020.

[51] B. Kailkhura, V. S. Siddhardh Nadendla, and P. K. Varshney, "Distributed inference in the presence of eavesdroppers: a survey," *IEEE Communications Magazine*, vol. 53, no. 6, pp. 40–46, 2015.

[52] A. Shoufan and E. Damiani, "On inter-rater reliability of information security experts," *J. Inf. Sec. Appl.*, vol. 37, pp. 101–111, 2017.

[53] M. H. u. Rehman, A. M. Dirir, K. Salah, E. Damiani, and D. Svetinovic, "TrustFed: A Framework for Fair and Trustworthy Cross-Device Federated Learning in IIoT," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2021.

[54] L. Mauri, E. Damiani, and S. Cimato, "Be Your Neighbor's Miner: Building Trust in Ledger Content via Reciprocally Useful Work," in *13th IEEE International Conference on Cloud Computing, CLOUD 2020*. IEEE Computer Society, 2020.