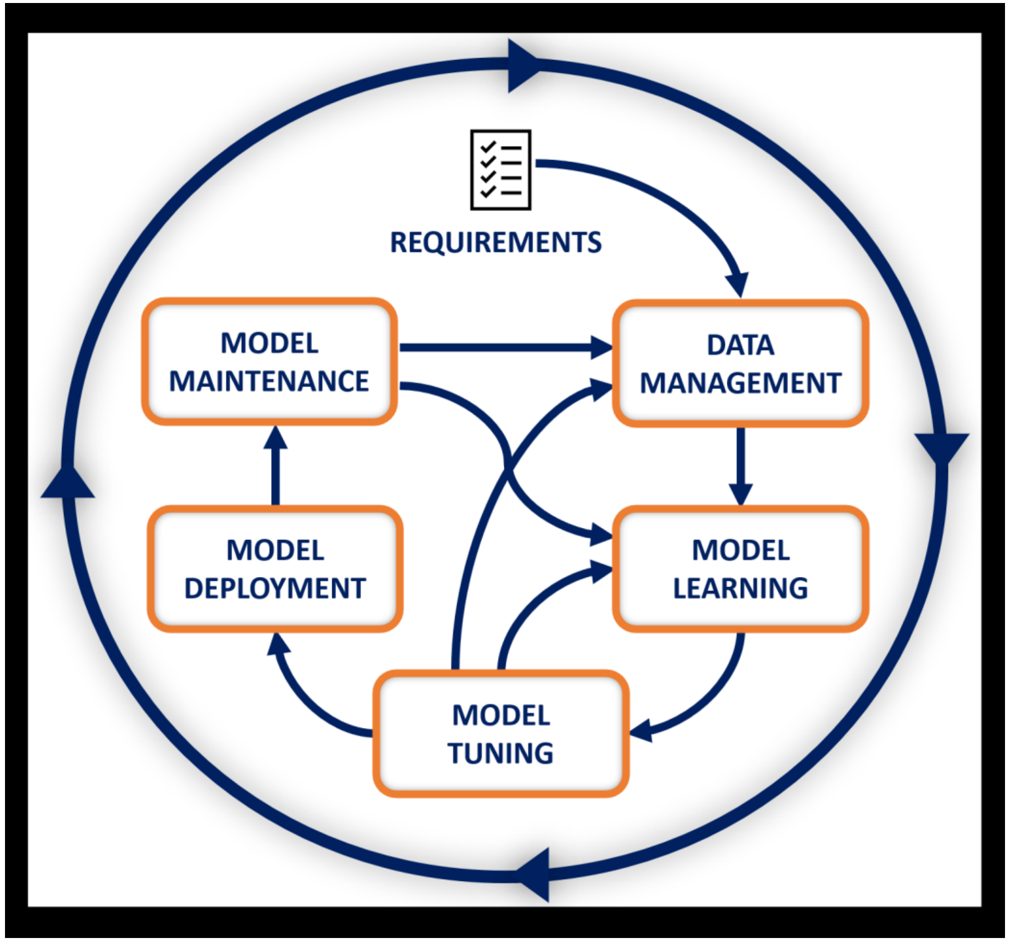
ML Lifecycle



AI Lifecycle stages

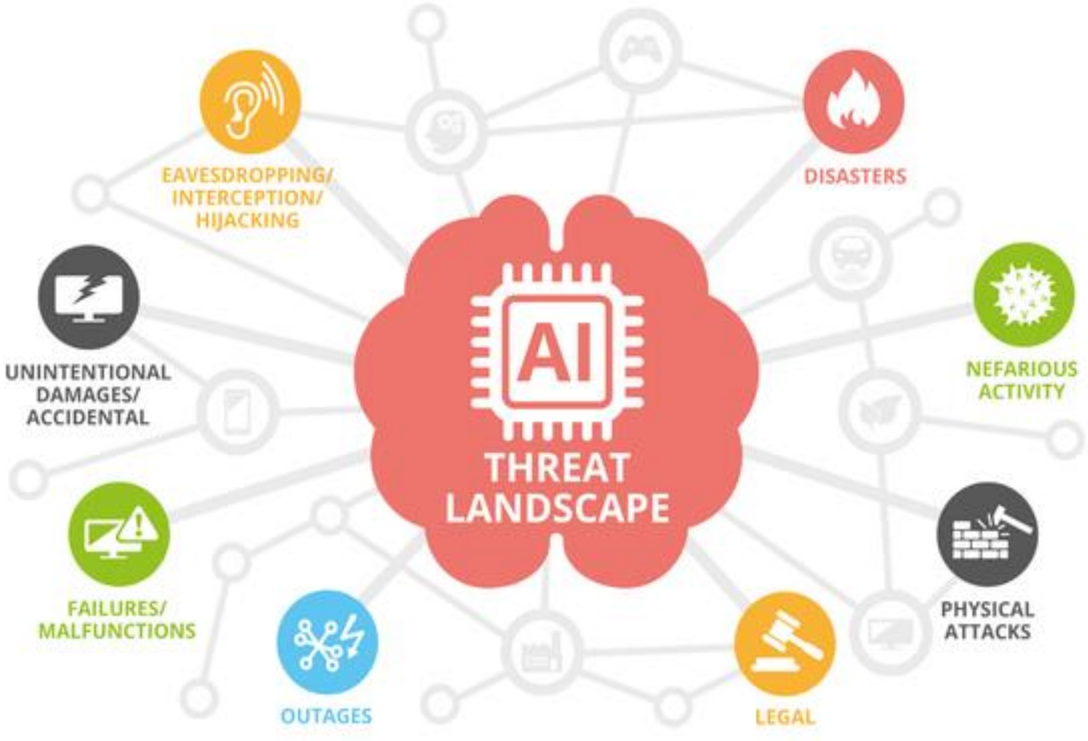
An *activity repertoire*, not a flowchart

Our Focus: *Training to Decommissioning*

- Requirements analysis
- Collect training data
- Prepare training data
- Choose ML method
- Develop classifier
- ***Train classifier***
- ***Improve classifier***
- ***Implement ML system/model***
- ***Deploy ML system/model***
- ***Re-train/feedback loop***
- ***Performance monitoring of ML model***
- ***Maintenance***
- ***Decommissioning***

Threat Modeling for AI

- **Reduce the gap between security practitioners and and AI experts via a structured approach to identifying, quantifying, and addressing threats to AI**
- **Full technique presented here: https://github.com/LaraMauri/STRIDE-AI**



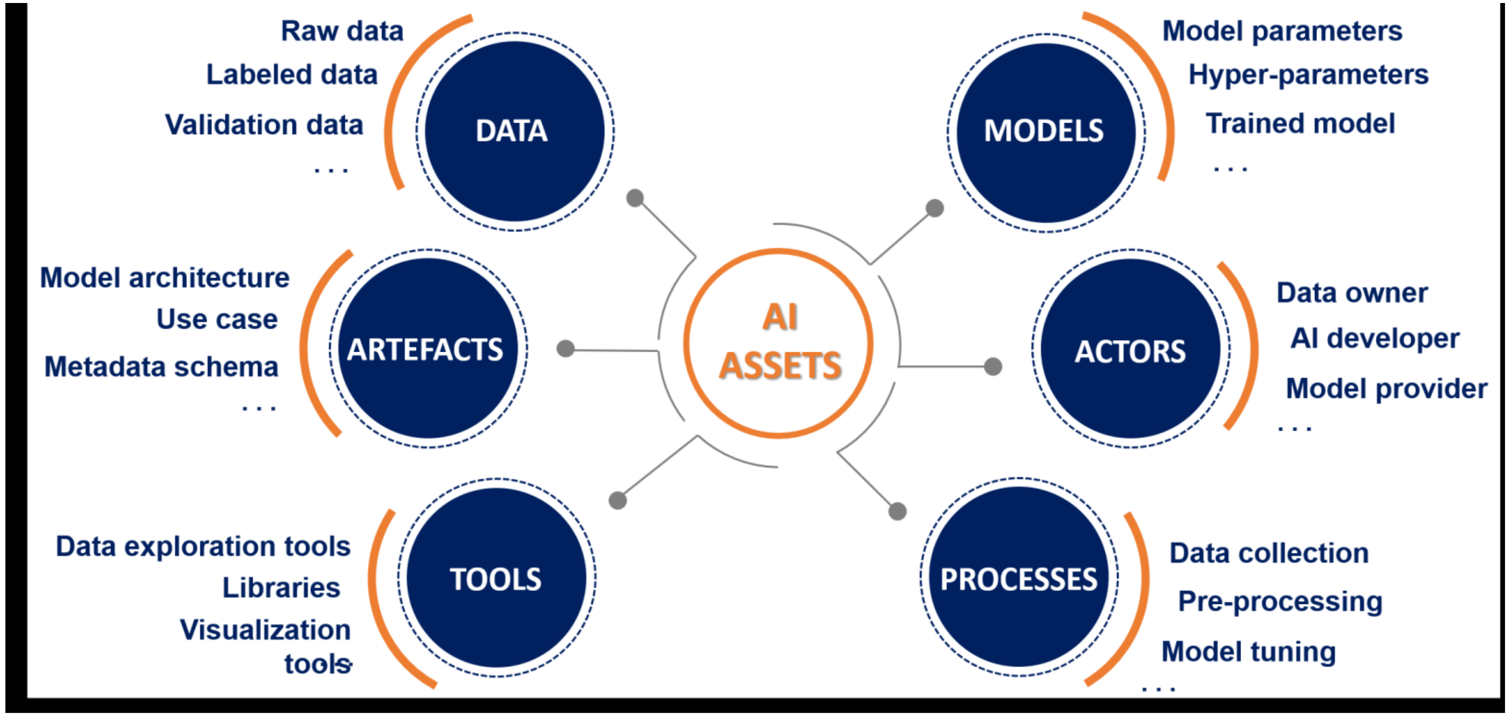
The Threat Modeling Process

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.*

Identifying Assets

- **What is it that you want to protect?**
  - Training/Test data
  - Inference results (e.g., containing intellectual property)
  - Parameters/Hyperparameters (e.g., weigths, loss function)
- **These also count as "assets"**
  - Storage/data lake
  - Machines on the network

ML DATA ASSET MODEL



FAILURE MODES

Failure mode refers to how a device, equipment, or machine can fail. If there are several potential ways that something can go wrong, we say that it has **multiple failure modes**. We can also use the term '**competing risks**.'

Table 1. 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 imply representativeness. 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. Assessment of data representativeness cannot be done *a priori*; it is only possible after identifying the targeted population and the purpose for collecting the data. Selection bias has been described in the scientific literature as due to malpractice [2].
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 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 [41], as this is considered a sign of *over-fitting*.	Validation data fail when labeled data items are modified. Modification of validation data items 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 loose rather than gaining 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.

STRIDE THREATS: GENERIC AND ML-SPECIFIC DEFINITIONS

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)	*Examines identified threats and determines if known attacks show that the overall system is vulnerable to them.*
Elevation of Privilege (EoP)	*This is a threat similar to spoofing, but instead of taking on the ID of another, they elevate their own security level to an administrator.*

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.*

CIA3-R Hexagon

- We map the STRIDE threats to six key security properties

Table 4. 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)

The overall mapping: Assets, Properties, Threats and Attacks

Table 6. 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 [4], 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 [52] 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 [27]. 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 [11]. 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 [17], 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 [39].*
Held-Out Test Cases	Integrity, Availability, Confidentiality	Tampering, DoS, Disclosure	*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 [5], 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 [24]) to distributed ML models involving eavesdropping on inferences.*

Documenting Architecture

- **Define what the system does and how it is used**
  - Ingestor collects data
  - Library trains model (computes parameters)
  - Sensor sends input
- **Diagram the application**
  - Show subsystems
  - Show lifecycle
  - List assets

The TOREADOR-Light Source Case Study

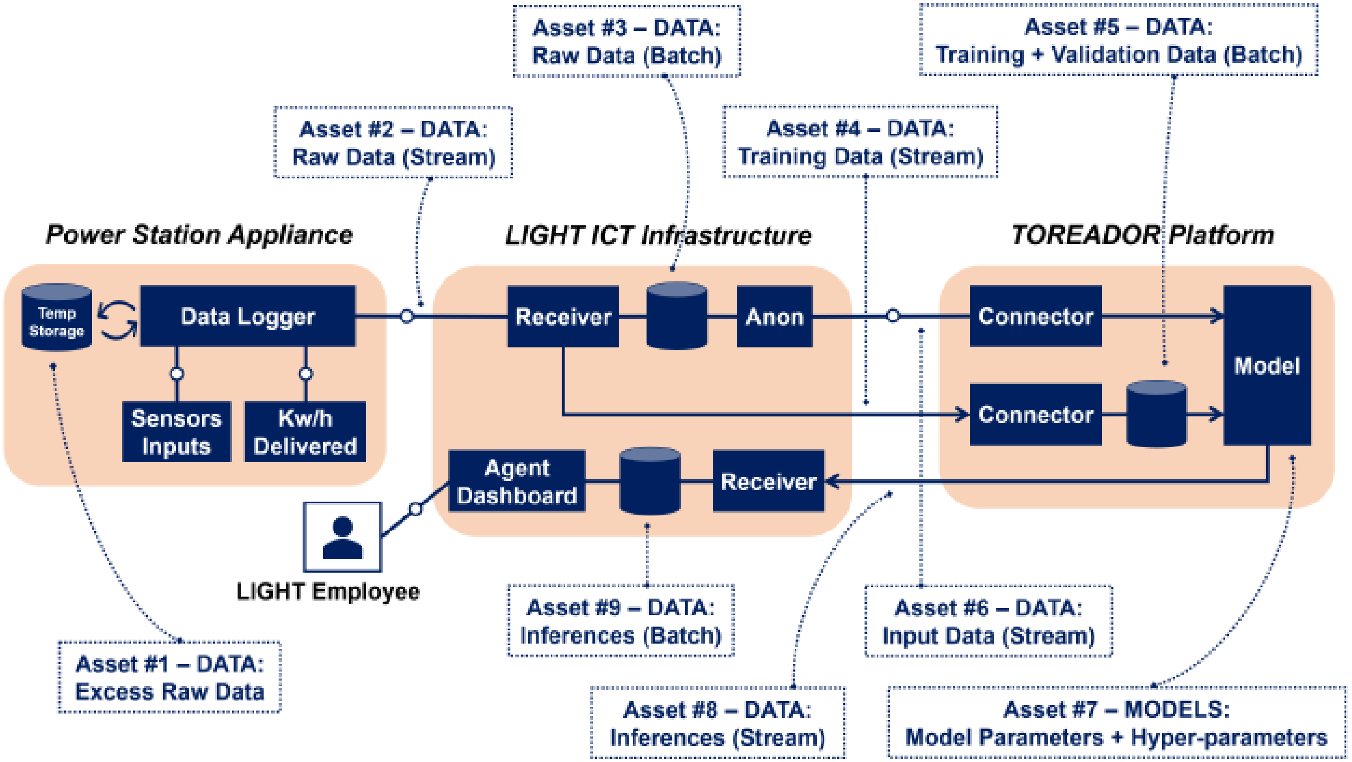


Table 8. Complete mapping for the assets identified in the use case.

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

Decomposing the Architecture

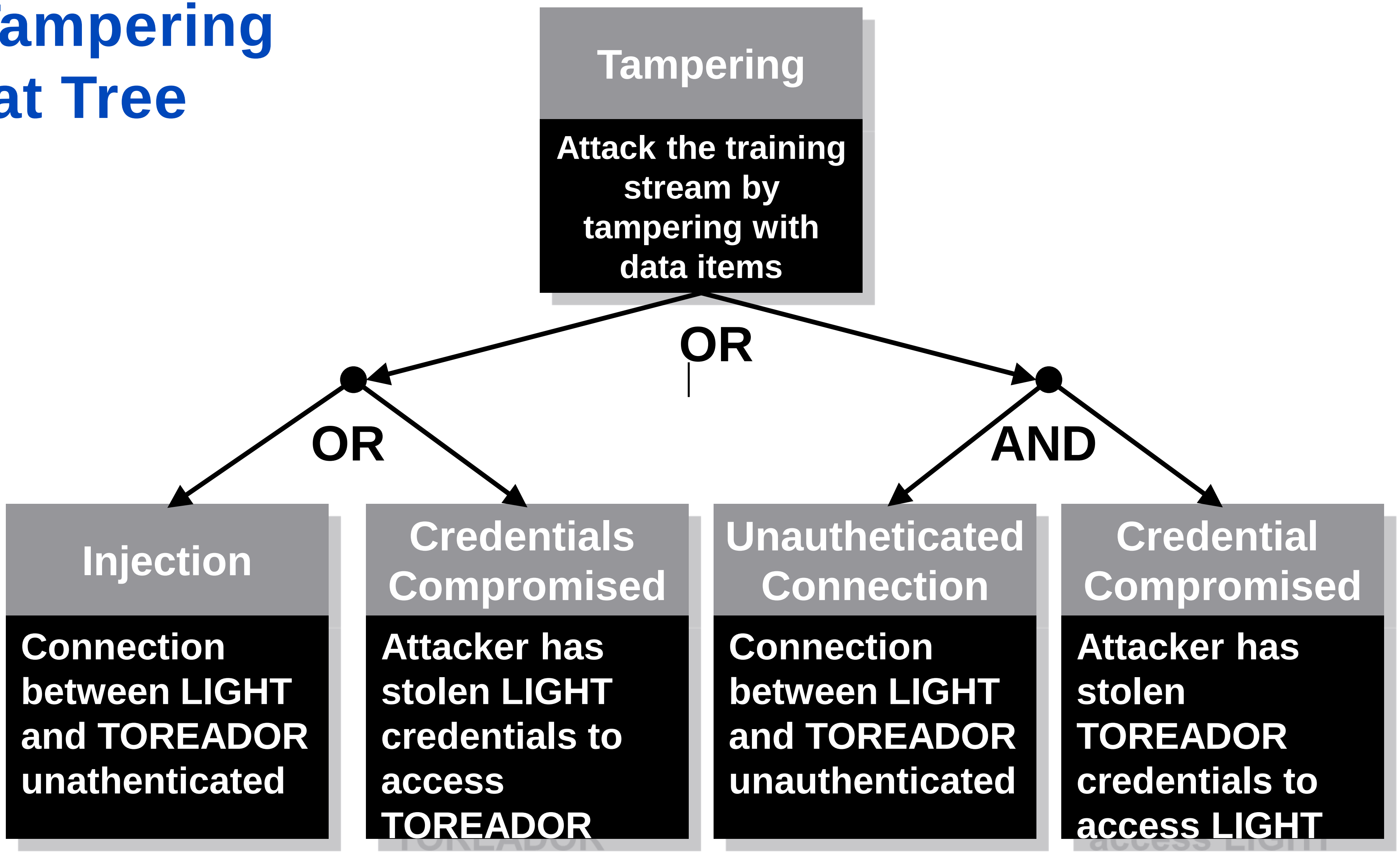
- **Refine the architecture diagram**
  - Show authentication mechanisms
  - Show authorization mechanisms
  - Show technologies (e.g., Tensorflow)
  - Diagram trust boundaries
  - Identify entry points
- **Begin to think like an attacker**
  - Where are the AI model vulnerabilities?
  - What am I going to do about them?

Documenting Threats

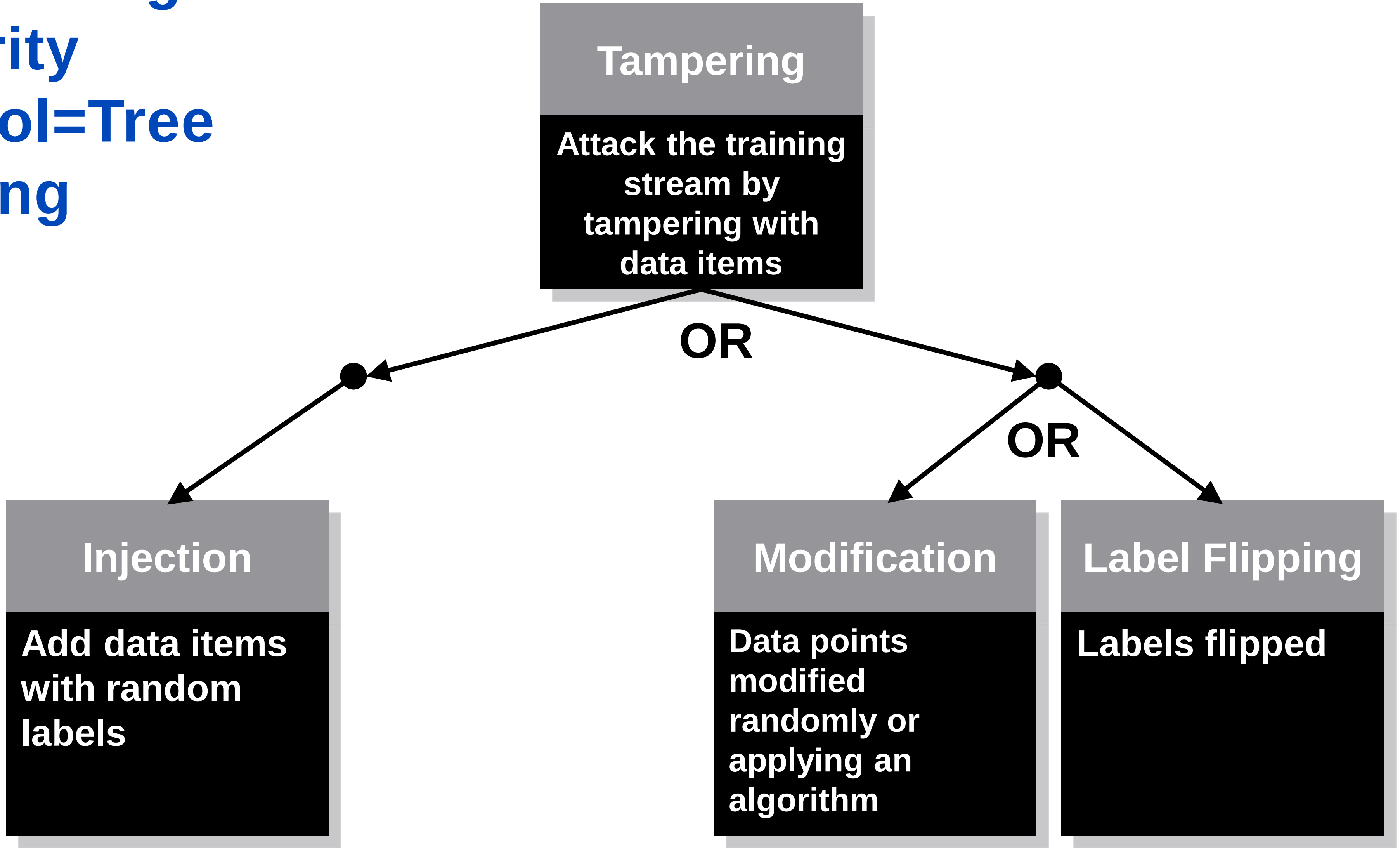
Theft of Training Data by Eavesdropping on Connection	
Threat target	Connections between sensors and ingestor
Risk	
Attack techniques	Attacker uses sniffer to monitor traffic
Countermeasures	Use SSL/TLS to encrypt traffic

Theft of Inference Data via Key Steal	
Threat target	Sensors
Risk	
Attack techniques	Attacker physically accesses Root-of-Trust
Countermeasures	Surveillance on sensor site

Analyzing Threats: the Tampering Threat Tree

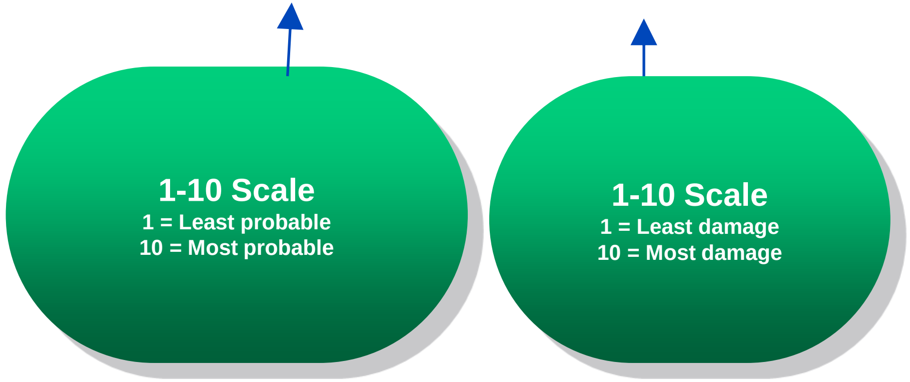


Introducing a security control=Tree pruning



Rating Threats

Risk = Probability * Damage Potential



- **DREAD model**
  - Greater granularization of threat potential
  - Rates (prioritizes) each threat on scale of 1-15
  - Developed and abandoned ☺ by Microsoft, still used by OpenStack
  - Simple model, does not directly take into account whether the attack requires a timing window

DREAD

D **Damage potential**
What are the consequences of a successful exploit?

R **Reproducibility**
Would an exploit work every time or only under certain circumstances?

E **Exploitability**
How skilled must an attacker be to exploit the vulnerability?

A **Affected users**
How many users would be affected by a successful exploit?

D **Discoverability**
How likely is it that an attacker will know the vulnerability exists?

Example

Threat	D	R	E	A	D	Sum
Substitution of Training Data by Man-in-the-Middle	3	2	4	2	5	16

Potential for damage is high (permanent backdoor , etc.)

Data can be substituted any time, but is only useful once system is deployed

Inserting data in non-authenticated connection requires moderate skill

All sensors could be affected, but in reality most won't be unauthenticated

Difficult to discover by testing the model