

How to Securely Design and Operate Machine Learning

Published 8 May 2023 - ID G00778428 - 45 min read

By Analyst(s): Wilco van Ginkel

Initiatives: [Analytics and Artificial Intelligence for Technical Professionals](#); [Build Trust and Mature D&A Culture](#); [Demonstrate Value and Collaborate With Business Partners](#)

Machine learning (ML) introduces new security risks that target machine learning infrastructure, data and models. Data and analytics technical professionals must use a risk-based approach toward ML security and apply actionable security controls to mitigate these risks.

Overview

Key Findings

- ML adversarial attacks are starting to emerge. Some 27% of organizations that have experienced an AI privacy breach or security incident also indicated that the breach/incident involved intentional malicious attacks on the organization's AI infrastructure.
- ML security is a shared responsibility among teams within the organization (such as D&A and security) because many security controls are cross-departmental.
- The vendor landscape of commercial ML security protection solutions is immature and nascent. However, startups are emerging.
- ML security should be taken into account during the design, implementation and operation of ML systems and not as an afterthought.

Recommendations

Data and analytics (D&A) professionals responsible for the security of ML must:

- Use a risk-based approach to prioritize ML security risks. This approach helps D&A professionals determine which attack vectors are relevant to the ML system and whether the ML system is vulnerable to these attacks.

- Focus on controlling vulnerabilities to mitigate risks, reducing the attacker's ability to exploit the attack (or attacks).
- Work together with different teams across the organization (for example, the security team) to ensure consistency and standardization of the implemented controls.
- Prioritize security controls that cover multiple attacks at once (such as monitoring the model inference requests) and those controls that must be implemented regardless of the stage of the ML system (that is, training, test, quality assurance [QA] or production).

Analysis

Machine Learning Security

What Is ML Security?

You can use ML to improve (or compromise) security, or you can use security to improve ML (that is, ML security). Examples include building resilience against ML adversarial attacks and reducing the risks associated with those attacks. This research is about the latter.

For the purpose of this research, an ML system consists of three components:

- **Data:** This represents the training and inference data used by the ML model.
- **ML Model:** This is the trained model that is put into production.
- **Code:** This represents the code and algorithms to build the model (for example, a Python script using the fast.ai image library to train an image classifier).

Why ML Security Matters

According to a Gartner survey, 41% of organizations had experienced an AI privacy breach or security incident. ¹ Of those reported breaches or incidents, 27% faced intentional malicious attacks on the organization's AI infrastructure. The survey also revealed that, of the organizations that use AI, 73% have deployed hundreds or thousands of models. Given this trend, Gartner has included resistance against adversarial attacks as one of its five pillars in [Top Strategic Technology Trends for 2023: AI Trust, Risk and Security Management](#).

Chief data and analytics officers (CDAOs) and their teams must secure ML models, which are becoming more pervasive. Unsecure ML models are not resistant against ML-specific attacks. Models are at risk, and as a consequence, so are any business or decision-making processes in which they are used.

This research describes a pragmatic, risk-based approach toward ML security with the aim to make it concrete and practical for the professionals responsible for its design and implementation.

Intended Audience

This research is geared toward those who are responsible for ML security in their organization. Other D&A technical professionals (such as data scientists, data engineers and ML engineers) can also benefit from this research because the described techniques and controls will help to improve the robustness of ML models, model data and the monitoring of model usage.

Note that the implementation of ML security is a cross-departmental responsibility. Different teams and roles are involved, such as the data and analytics team and the security team. Within these teams, different roles can be assigned, including ML architect, data engineer and data scientist for the D&A team and security architect and security engineer for the security team. This will be further explained later in this research.

Risk-Based Approach Toward ML Security

This research takes a risk-based approach toward ML security. From a security perspective, a risk is multifaceted and contains the following conditions:

- An attack vector is aimed at the ML system component (or components).
- A successful attack has an impact on the system.
- The ML system component is vulnerable for the attack.
- The attacker (internal or external) is able to exploit the vulnerability.

An example of these conditions is as follows:

- An attack vector is data poisoning, which is the deliberate insertion of malicious data in the training set. The purpose of data poisoning is to cause the ML model to learn from malicious data, leading to wrong results in the model's output.

- The impact is an unreliable model.
- The attack can happen due to, for example, inadequate data access management. This vulnerability allows the attacker to access the training data.
- The attacker is able to access the training data — for instance, via a remote connection to the data storage where the training data is located.

In this example, business processes that use the model are at risk because of the unreliable model outputs. For example, consider a model that classifies financial transactions as fraudulent or benign. If an attacker is able to insert fraudulent transactions that are labeled as “benign” in the training set, the model learns to classify these as benign (in case of supervised learning). When the production model is queried to classify a transaction and then classifies that transaction as “benign,” it becomes ambiguous as to whether this is a true benign transaction.

Risks are mitigated by the implementation of security controls with the aim to reduce and control the vulnerabilities. The focus is on vulnerability control because it limits attackers’ ability to exploit the vulnerability.

Securely Design and Operate ML

To secure an ML system is to control risks, which is to control vulnerabilities. In order to do that, you must:

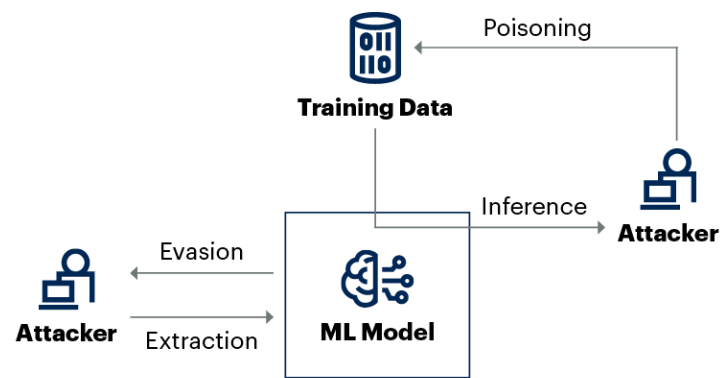
- Take the viewpoint of possible attack vectors for each ML system component (data, model and code).
- Define the vulnerabilities associated with the attack vectors.
- Define which security controls are applicable to control the vulnerabilities.

Attack Vectors

ML system attack vectors are depicted in Figure 1.

Figure 1: ML Adversarial Attack Vectors

ML Adversarial Attack Vectors



Source: Adapted From GitHub
778428_C

Table 1 explains, at a high level, what these attacks entail. The attacks will be described in more detail later in this research.

Table 1: High-Level Explanation of the ML Adversarial Attack Vectors

Attack	Explanation
Poisoning	Deliberate insertion of malicious data into the training set
Inference	Determine whether a data point was in the training set
Extraction	Steal a model by creating a functionally equivalent surrogate of the original model
Evasion	Provide the model a fabricated input in order to manipulate the model's output

Source: Gartner (May 2023)

As shown in Figure 1, the attacks apply to a specific ML system component. For instance, data poisoning is an attack aimed at the training data, while an evasion attack is aimed at the model.

Attacks on a target can be direct or indirect. For example, data theft is a direct attack and targets the data storage with the aim to make an unauthorized copy of the data. By contrast, data poisoning is an indirect attack — the target is not the data itself, but the model's behavior; the attack is used as a steppingstone (hence, indirect) to manipulate the training data and influence the model's behavior.

Attacks can also be white-box or black-box attacks. This is particularly important for the ML system model component. In case of a *white-box attack*, details about the deployed model (such as inputs, model architecture, hyperparameters and the specific model internals like weights) are known to the attacker. For example, the goal of model extraction is to build a surrogate of the target model. In this case, when the attacker knows the internals of the target model (white box), it is much easier to build a surrogate model.

With a *black-box attack*, only the model's inputs are known. The target model must be queried with (fabricated) inputs to get the model's outputs. The input-output pairs are the only information revealed to the attacker, which makes it much harder to extract a model.

Security Controls

A layered approach to security controls is required because of the diversity of the ML system components at risk and the different types of ML adversarial attacks. These controls can be separated in two groups:

- **Common security controls:** These controls are typically implemented across an organization, such as data access management, security monitoring and encryption. These controls are the responsibility of the security team. Different roles can exist in the security team, where each role is responsible for a set of security controls (for example, security operators for network security monitoring). The security team roles are outside the scope of this research.
- **ML-specific controls:** These are security controls specifically aimed at ML, such as adversarial attack testing. These controls are the D&A team's responsibility, and different roles take on different responsibilities (for example, a data scientist is responsible to test models for model evasion robustness, and a data engineer is responsible for data protection to avoid data poisoning).

The D&A team must work closely with the security team. The D&A team should leverage existing security controls that have been implemented by the security team. This is to ensure organizationwide consistency and compliance with company standards, practices, processes, policies, procedures, technology and solutions. For example, the D&A team should not implement its own identity and access management (IAM) solution. Instead, it should use the IAM solution that has already been implemented by the security team.

Depending on the skills and expertise in the D&A team, the team can be a custodian for the security team. In that case, the D&A team implements the security solutions on behalf of the security team, but with its support. Alternatively, the security team can take care of the security solution implementation for the D&A team. In this case, from the security team's perspective, the D&A team is just another end user, rather than an active participant in the implementation.

For more detailed research on roles and skills to support advanced analytics and AI initiatives, see [Roles and Skills to Support Advanced Analytics and AI Initiatives](#).

ML System Data

This section first describes the possible attack vectors related to the ML system data component. This is followed by a description about where these attacks can happen in the ML pipelines and, finally, which security controls can be applied.

Data Poisoning Attack

As the name of the attack implies, the goal of the attacker is to manipulate the model's output by inserting malicious ("poisoned") data into the training set. During the training phase, the model will learn from this incorrect data, which can lead to unreliable model output. This becomes a risk when this output is used for decision making or as input for other systems.

For example, assume you want to build a classification model for fraud detection using a supervised ML algorithm, such as logistic regression. When an attacker inserts fraudulent transactions with the label "Not Fraudulent" in the training data, the model will learn to classify these types of transactions as "Not Fraudulent." When fraudulent transactions are accepted, this can lead to, for instance, business, legal and financial risks.

Backdoor attacks are a special case of data poisoning. In this case, the attacker taints the training dataset to include examples with visible triggers (e.g., for a facial recognition model). During training, the model will associate the trigger with the target class. During inference, when the model is presented a normal image, it will act as expected. But when the model sees an image that contains the trigger, it will label it as the target class regardless of its contents — for example, a facial recognition model that classifies people wearing a specific type of glasses as the target class.

A data poisoning attack can be either direct or indirect. In a direct attack, the attacker has access to the data due to, for example, the lack of proper data access controls. In the case of an indirect attack, malicious data from an external data source is inserted in the training set when the data source and the data have not been validated before use.

Data poisoning can happen when the data component of the ML system has vulnerabilities, such as:

- Inadequate data access management, which allows the attacker to access the data sources and/or the data storage
- The use of unreliable (external) data sources, which contain malicious data and have not been validated
- Inadequate testing for possible data poisoning, which allows data poisoning to happen without being detected

Membership Inference Attack

A membership inference attack (MIA) tries to discover whether a certain data point was part (that is, a member) of the training set. This is a risk when the training data contains sensitive, private or confidential data, such as health information of individuals.

It is an indirect attack because the data point is not revealed directly from the training data (such as a look-up of that data point in the training data), but rather, via the ML model (called the target model). The attack can be a white-box or black-box attack, depending on how much the attacker knows about the underlying distribution of the training data and the target model (e.g, architecture, algorithm and hyperparameters). In case of a black-box attack, the attacker can only query the target model to get its output(s).

An MIA can be hard to execute because the attacker needs to build an attack model, which has to be trained on a different training set than the target model's training set. This is because the attacker does not have access to the target model's training; otherwise, a simple look-up of the data point would suffice. The attacker needs to query the target model with a considerable number of input queries to get to the necessary input-output pairs to build the attack training set. The MIA has evolved and has eased some of its constraints (e.g., required knowledge of the attack model and the training's data distribution). What has not eased are the queries required to infer the target model. This can trigger the monitoring of the ML model endpoint because these queries indicate abnormal inference behavior.

With the MIA, the attack has a higher probability of success when the target model is overfitted. This results from the fact that, in case of overfitting, the target model does well on training examples, but does not generalize as well on unseen data. This can help attackers when they observe the target model's output (e.g., a classification confidence score of the input, such as "80% confidence that this image is a dog"). When the input has a low confidence score, it is likely that the target model has never seen the example (i.e., the example was not used for training).

Overfitting an ML model is not desirable in general – regardless of possible attacks – because it does not generalize well on not-before-seen (production) data. However, there is no guarantee that ML models in production won't overfit. For example, it is hard to verify overfitting when detailed training, validation and testing metrics are lacking (e.g., in the case of autoML or open-source models). Or, as another example, when the model developer does not have a good understanding of the overfitting concept.

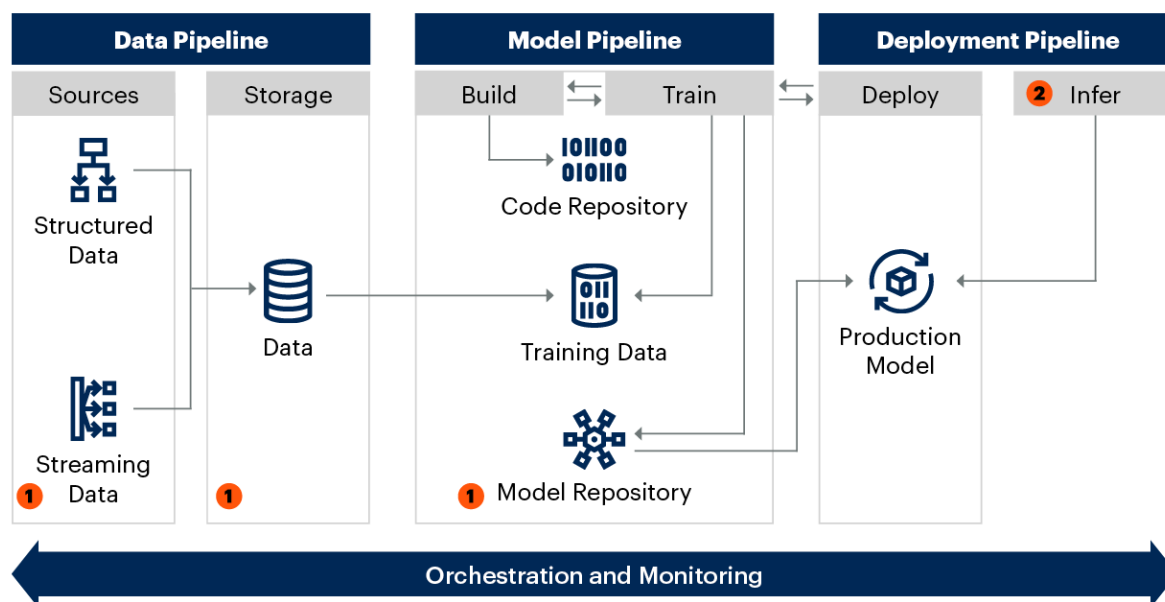
Where Can These Attacks Happen?

Figure 2 shows where the ML system data attack vectors can happen across the ML pipelines. The data poisoning attack is represented by "1," and "2" represents a membership inference attack.

Figure 2: ML System Data Attack Vectors

ML System Data Attack Vectors

❶ Data Poisoning Attack ❷ Membership Inference Attack



Source: Gartner
778428_C

Gartner









A direct data poisoning attack can happen in the data pipeline when attackers have access to the data sources and/or the data storage (for example, a database or data lake). The attack can also occur when attackers have access to the training data in the model pipeline. This is the case when the training data, or a copy of it, is stored outside the data storage. An indirect data poisoning attack can happen when unvalidated external data (which contains malicious data points) is used as a source for all or part of the training set.

The membership inference attack can happen in the deployment pipeline when attackers query (or infer) the target model.

Data Security Controls**Data Poisoning**

Table 2 shows the security controls to protect against data poisoning.

Table 2: Security Controls to Protect Against Data Poisoning

Control	Common or ML	D&A	Security
Implement data access management	Common		
Implement review governance for third-party (open) data	Common		
Test model for data poisoning	ML		
 = Shared Responsibility  = Teams Responsibility  = No Responsibility			

Source: Gartner (May 2023)

The security controls are a combination of common security controls and ML-specific security controls. As shown in Table 2, for the common security controls, the D&A and the security team have a shared responsibility (as represented by the half-open Harvey balls). Meanwhile, the ML-specific security controls are the sole responsibility of the D&A team (as represented by the closed Harvey ball).

The common security controls to prevent data poisoning are to implement data access management and to review governance.

The implementation of *data access management* reduces unauthorized access and changes to data sources and data storage. The D&A Team's (ML Architect) responsibility is to work together with the security team to determine how the organization's data access management solution can be implemented for the ML System pipelines. The solution implementation can be the responsibility of the Data Engineers, while they work in close partnership with the security team for guidance and support. Alternatively, the security team takes on the full responsibility of the data access implementation for the ML System pipelines and the D&A team is the end user.

For the *governance review process*, the ML architect works together with the security team (and/or the legal team) on the organization's review process for third parties and whether the review process needs to be adapted to review third-party or open data sources. Gartner recommends the following actions:

- Approve list the sources that are considered trusted (e.g., third-party reputation).
- Verify the integrity of the data using secure hash codes or digital signatures. This assures that the data has not been altered and that it is from the claimed source — for example, a digitally signed source from a government agency.
- Limit the accepted data formats to those that can be manually verified, such as CSV or JSON.
- Download the dataset over a secure and verifiable connection (for example, checking the X.509 certificate and Secure Sockets Layer/Transport Layer Security [SSL/TLS] connection). Download the dataset in a separate test environment, and validate the digital signature (if applicable) and the data for inconsistencies or malicious content.

The ML-specific security control to prevent data poisoning is to test the model.

To *test the model* is to detect whether some of the model's output is corrupted or erratic, which may indicate a data poisoning attack. This detection can take place when the model is developed and/or when the model is deployed.

To reduce a data poisoning attack, Gartner recommends the following preventative actions during the build and training of the model:

- Data engineers should validate the data lineage from the data sources to the data storage. In case of third-party or open data sources, the data engineer should verify the data according to the data governance process.

- Data scientists should validate the training data (for example, exploratory data analysis and statistical testing) for inconsistencies before using it for training. Where possible, they should also improve the robustness of the model by incorporating adversarial examples in the training data. Open-source tools such as ART, Armory and Counterfit are good starting points in that regard:
- The [Adversarial Robustness Toolbox \(ART\)](#) is a Python library for machine learning security and provides tools to defend and evaluate machine learning models and applications against the adversarial threats of evasion, poisoning, extraction and inference. ART supports all popular machine learning frameworks (TensorFlow, Keras, PyTorch, Apache MXNet, scikit-learn, XGBoost, LightGBM, CatBoost and GPy), all data types (images, tables, audio and video) and machine learning tasks (classification, object detection, speech recognition, generation and certification).
- [Armory](#) is a testbed for running scalable evaluations of adversarial defenses and runs as a local Docker container or as a cloud instance. Armory standardizes all attacks and defenses as subclasses of their respective implementations in ART.
- [Counterfit](#) is a generic automation layer for assessing the security of machine learning systems. It brings several existing adversarial frameworks (such as ART and [AugLy](#)) under one tool, or allows users to create their own.

To improve the model robustness against a data poisoning attack:

- Train the model (assuming that no data poisoning has been found in the training data), such as a convolutional neural network (CNN; in case of image classification).
- Evaluate the model metric on the test set (e.g., accuracy).
- Create adversarial training and test examples based on the training and test data (e.g., with DeepFool, which is part of ART).
- Evaluate the model metric on these adversarial examples.
- Augment the training data with the adversarial examples.
- Retrain model on augmented training data.

- Evaluate metric of the retrained model on the test set. This should lead to a better accuracy compared with the original model, given that the model has also been trained on adversarial examples.

Improving the model robustness is not an exact science because it depends on the model (or algorithm) used, the training data and the possibility to create adversarial attacks, based on the training data. As such, improving the robustness is a recommended option and not a primary control, because it may not always be feasible. Proper data access management is more effective and should be the first line of defense.

When the model is deployed, it should be monitored for its performance, data and model drift, and erratic behavior. Erratic behavior may be an indication of data poisoning. In particular, when the model is retrained and the training data verification (as explained above) has not been followed. Every time a model is retrained, verification of the training data is key because it contains new or updated data, which may have been poisoned.

Membership Inference Attacks













The security controls to protect against an MIA are twofold:

- Protect the training set on a data level (e.g., synthetic data and differential privacy). This renders an MIA unsuccessful because the training set does not contain the real data.
- The prevention of overfitting in case data-level protection is not possible.

Gartner recommends data-level protection as the first line of defense because it protects the data at its source. The prevention of overfitting becomes important when data-level protection is not possible.

Table 3 shows the security controls to protect against an MIA:

Table 3: Security Controls to Protect Against Membership Inference Attacks

Control	Common or ML	D&A	Security
Use data masking or synthetic data	Common		
Use differential privacy	Common		
Monitor inference attempts	Common		
Prevent overfitting	ML		
Evaluate and test MIA	ML		
 = Shared Responsibility  = Teams Responsibility  = No Responsibility			

Source: Gartner (May 2023)

The common security controls to protect against an MIA are to use data masking or synthetic data, use differential privacy, and monitor model inference attempts.

Data masking protects sensitive data by providing users (such as data scientists) with fictitious yet statistically equivalent data, instead of real and sensitive data, while maintaining users' ability to train ML models. For instance, instead of a real, valid Social Security number (SSN), the training data has an SSN, which resembles or looks like an SSN in terms of format, but is not a real, valid SSN.

Fogger and Presidio are examples of open-source tools. Examples of commercial tools are Accutive and Immuta. For more detailed research regarding data masking, see [Market Guide for Data Masking](#).

Synthetic data (SD) is a class of data that is artificially generated rather than obtained from direct observations of the real world. This data can be generated using different methods, such as statistical sampling from real data distributions or generative AI. SD is critical in removing privacy data, such as personally identifiable information (PII). However, synthetic data can also be used to generate sufficient training data. SD is on the Peak of Inflated Expectations and, although promising, is dependent on the use case to which it applies (for example, it may not always be possible to generate SD based on real data).

Examples of open-source tools are Mimesis and Synthetic Data Vault. Argusa and BetterData are examples of commercial tools. For more information on synthetic data, see [Emerging Tech: Top Use Cases for Tabular Synthetic Data](#).

Differential privacy (DP) is a system for using or sharing a dataset while withholding or distorting certain information elements about individual records in the dataset. The goal of applying DP on the training data is to ensure that:

- The ML model can be trained as if DP was not applied.
- An acceptable model metric can still be achieved (that is, the accuracy is close to the accuracy when trained on non-DP training data).
- No identifiable information has been disclosed.

TensorFlow Privacy Library and Opacus are examples of open-source tools. Examples of commercial tools are LeapYear and Privitar.

Depending on the roles and responsibilities in the organization, the D&A team and security team can have a shared responsibility. One option is to consider data masking, synthetic data generation and/or differential privacy as the responsibility of the security team in terms of design and solution implementation. The D&A team (e.g., the data engineer) implements the solution in the data pipeline with guidance and support of the security team. Alternatively, these roles can be reversed, where the D&A team is responsible and the security team uses/implements the D&A team's solution.

Note that the above-mentioned tools are examples and don't represent an exhaustive list.

A potential indicator of a model inference attack is a spike in *model inference requests*, which is outside its inference pattern. An alert should be triggered for the D&A team (for example, for the data scientist) to further investigate this (unexpected) spike. There are different options to trigger these alerts, such as:

- ML monitoring tool, which is triggered by the spike in inferences: The ML operations (MLOPS) team, the network operations team or the IT infrastructure operations team could operate such a tool.
- A security information and event management (SIEM) tool, which is triggered by internal and/or external spikes in API calls: The SIEM tool can be operated by the security operations team, for instance.

The ML-specific security controls to protect against and MIA are to prevent overfitting and to evaluate and test MIA.

There are different techniques, such as regularization, that add information to a model to *prevent overfitting*. Some techniques are specific to the ML model, while others are model-agnostic. For example:

- Dropout is a well-known regularization technique for (deep) neural networks (NNs). During training of the NN, randomly selected neurons are ignored. As a result, the NN becomes less sensitive to the specific weights of neurons. This, in turn, results in a network capable of better generalization and less likely to overfit the training data.
- Model stacking can be applied to different models. The idea behind model stacking is that the final model is less prone to overfitting. This is achieved by training multiple models on different, disjointed subsets of the training data and stacking them together in a final model. For instance, for a classification model, the first model is trained with an NN on a subset of the training data. The second model is trained on another subset of the training data with a random forest (RF), and the third model combines the output of the NN and the RF and is trained using a logistic regression model.

The prevention of overfitting is not an exact science, but rather trial and error at times. As this depends on the training data used, the ML model and the required model metric (such as accuracy or F1 score), to name a few. In some cases, it may already be a challenge to achieve the right bias-variance balance. This may become more challenging when overfitting prevention techniques have a negative impact on the model's required model metric (for example, too much regularization may reduce the model's accuracy on production data).

There are different open-source libraries available to *evaluate and test membership inference attacks*. These libraries are a great starting point for testing and provide a framework and code examples. Gartner recommends researching these tools to get a better understanding of how these attacks work. The next step we recommend is to apply MIA testing to your own ML models (note that MIA testing is focused on supervised learning). MIA testing will help you address privacy concerns and compliance, and is especially important in case the model is overfitted.

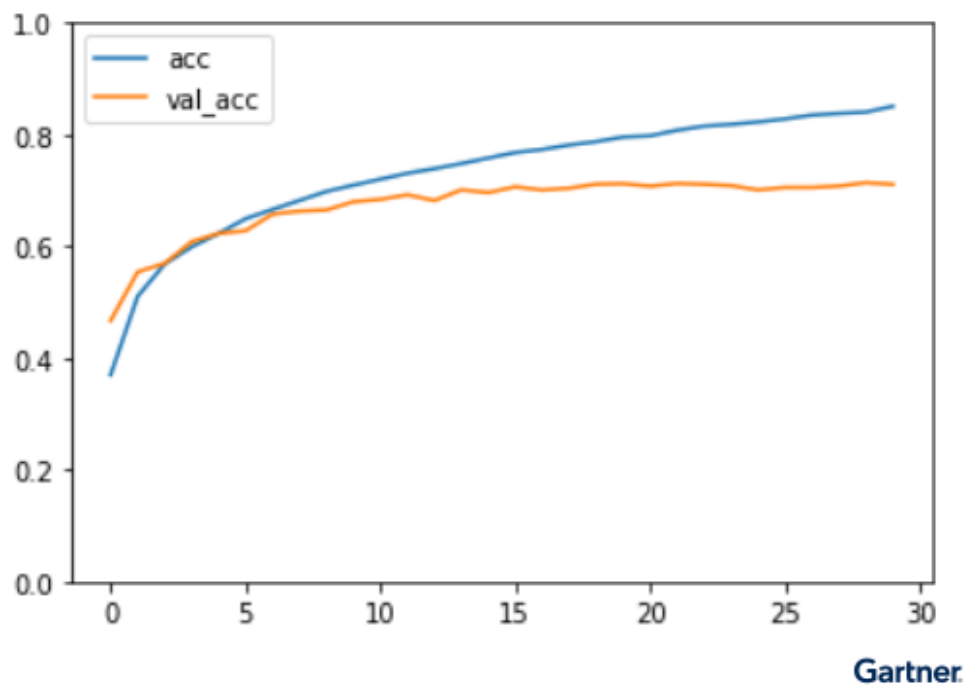
Two examples are the MIA options in:

- ART
- TensorFlow Privacy Library

For example, the MIA tests from the TensorFlow Privacy Library are black-box tests, where only the outputs of the ML model are used (such as losses, logits and predictions). The attack does not need to know about model internals (weights) or input samples. There are many different (configuration) options to choose from, including SlicingSpec, which makes it possible to slice the dataset (for example, to focus the MIA on specific data groups or classes). Another option is AttackType, which determines which MIA is conducted (for example, logistic regression, random forest or k-nearest neighbors). It is also possible to run MIA tests without the need to train an attack model.

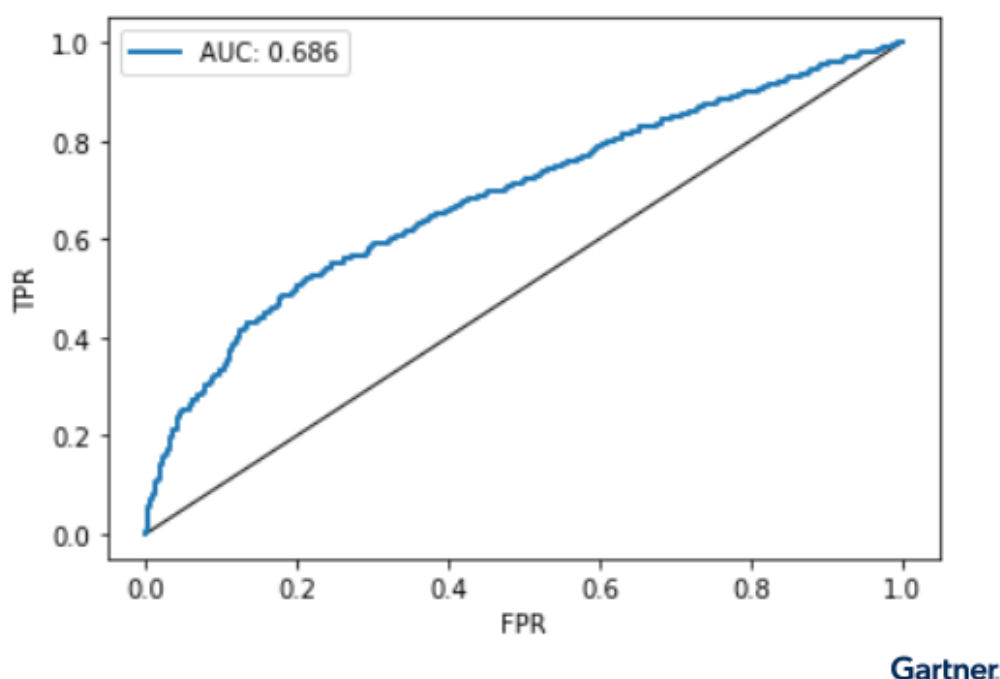
Figure 3 shows the accuracy scores for the training and the validation set for a convolutional neural network (CNN), which was trained on the CIFAR-10 (Canadian Institute for Advanced Research, 10 classes) dataset (30 epochs).

Figure 3: AUC Accuracy Scores



Source: Gartner

As shown in Figure 3, the training accuracy is around 85%, and the validation accuracy is around 70%. The reason is that the model is not trained with regularization, which leads to an overfitted model. Figure 4 shows the AUC (area under the ROC curve) score for the most successful MIA (which is a logistic regression attack on the complete dataset of the CNN).

Figure 4: AUC for the Most Successful MIA

Source: Gartner

Figure 4 shows that the MIA is better than random guessing, or a “coin flip,” in determining whether a specific picture was in the training set or not. This is because the model is overfitted.

ML System Model

This section describes the possible attack vectors related to the ML system model component. This is followed by a description of where these attacks can happen in the ML pipelines and, finally, the security controls that can be applied.

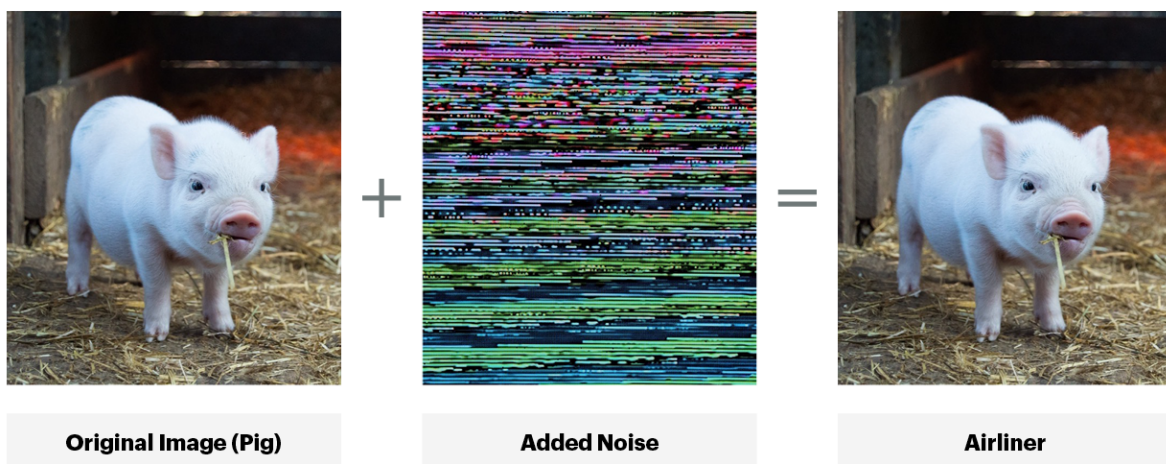
Evasion

The purpose of an evasion attack is for the attackers to evade the model’s output by querying the model with manipulated input. This is a risk because this leads to unreliable model output. For example, in case of a fraud detection model, the model might classify a manipulated fraudulent transaction as benign instead of fraudulent. Another example is the manipulation of physical objects, which can throw off an image classification model (see stop sign example in Figure 6).

It can be a white-box or black-box attack, which happens at inference time. It is a direct attack because it is directly aimed at the model. The attackers infer (query) the model by sending slightly manipulated input vectors to analyze the model's response to those inputs. The goal is to find a manipulated input, which tricks the model. For example, Figure 5 shows how a computer vision model could be tricked in its classification. The model correctly classified the original image as a pig. However, by adding noise to the picture (which is invisible to the human eye), the model classifies the altered image as an airliner.

Figure 5: Example of Model Evasion of an Altered Image

Example of Model Evasion of an Altered Image

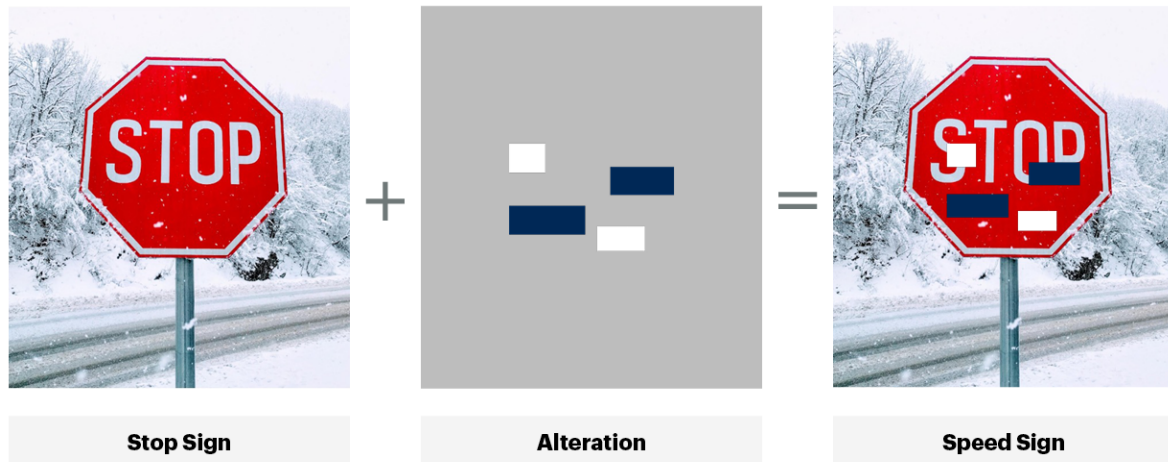


Source: Gartner
778428_C

Gartner

Another example of model evasion is due to the perturbation of a physical object. In Figure 6, the model misclassified the altered stop sign (that is, the stop sign with the stickers) as a speed sign instead of a stop sign.

Figure 6: Example of model evasion of an altered stop sign

Example of Model Evasion of an Altered Stop Sign

Source: Gartner
778428_C

Gartner

Evasion attacks can happen when the model is not robust enough. Without testing, it is hard to determine whether a model is resistant to these attacks because there are many different evasion attacks. Some are focused on just images (such as spatial transformation), while others are aimed at multiple data types, such as images and tabular data (for example, the HopSkipJump evasion attack). Depending on the data distribution, an attack might not be feasible, simply because the attack requires too many iterations during execution (for example, the number of pixels in high resolution images in a pixel-based attack).

Gartner recommends that you evaluate the evasion robustness for high-risk models. These models (and subsequently, the business process or system in which they are used) are impacted when a publicly known evasion attack is successful. Examples:

- Fraudulent transactions, which the model classifies as benign
- Image recognition models, which misinterpret altered signs (such as the stop sign example)

Extraction

The purpose of an extraction attack is for the attackers to obtain detailed information about the target ML model. They then use this information to re-create a close approximation (or “copy”) of the target model. This is a risk because the target model is copied (theft of intellectual property). Another risk is that the target model can be studied (using the copied model) as preparation for another attack, such as model evasion of the target model (two-step model attack).

The attackers use extraction techniques, such as querying the model with specific input, to observe the model’s output. Based on these input-output observations, they try to reconstruct the model. The attack can be a black-box or white-box attack, which happens at inference time. As with model evasion, model extraction is a direct attack because it is directly aimed at the model.

An example of a white-box scenario is when the model is deployed on an ML-as-a-service (MLaaS) platform and the platform provides insights in its (autoML) models. The attacker can study those models (algorithms, setup, hyperparameters tuning, training and deployment APIs) by simply using them. This provides additional information on top of the input-output observations.

Note that stealing a model (that is, copying the model’s code from a code or model repository) is not considered an extraction attack because model evasion techniques are not required. The result is still not desired, because this gives the attacker an identical copy of the model.

Model extraction can be difficult. The following factors have an impact on a successful model extraction:

- The algorithm used in the model: For instance, it is easier to extract a logistic regression model than a random forest model. A random forest model has more complexity in its algorithm (randomness of building the forest being one of them) than logistic regression.
- The information that is exposed as part of the model’s output: For example, the model provides detailed information on confidence levels instead of a (simple) label. An example of confidence levels — in case of a classification model for fraudulent transactions — is [fraudulent (0.78), benign (0.22)]. Instead, a simple label would be “1” (in case of fraudulent) or “0” (in case of benign), and the model’s output is just the label “1” or “0,” depending on the classification.

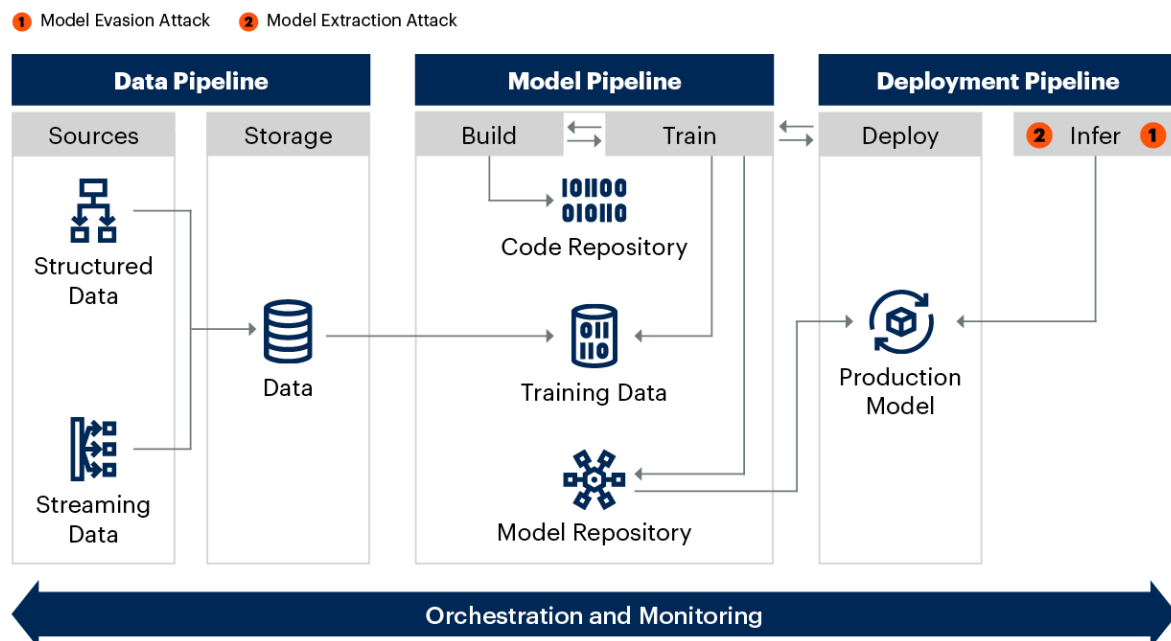
- Model inferences unnoticed or spikes not triggered: The model needs to be queried in order to get the input-output observations. Depending on the model, the number of queries can run into the hundreds of thousands, using the model's deployment/production API. Not monitoring or detecting this spike allows the attacker to continuously gather input-output observations.

Where Can These Attacks Happen?

Figure 7 shows where the ML system model attack vectors can happen across the ML pipelines. The model evasion and extraction attacks are represented by "1" and "2," respectively.

Figure 7: Model Evasion and Extraction Attacks

Model Evasion and Extraction Attacks



Source: Gartner
778428_C

Gartner

Model Security Controls

Evasion

The security controls to protect against evasion attacks are twofold, as described in Table 4.

Table 4: Security Controls to Protect Against Evasion Attacks

Control	Common or ML	D&A	Security
Improve model robustness	ML		
Monitor evasion attempts	Common		
 = Shared Responsibility  = Teams Responsibility  = No Responsibility			

Source: Gartner (May 2023)

The ML-specific security control to protect against evasion attacks is to improve model robustness.

This is a two-step approach:

- Evaluate the model for evasion robustness (for instance, for high-risk models). This is done by attacking the model with known evasion attacks.
- Update the model training to make it more robust by implementing attack-related measures for which the model is vulnerable. Examples are feature scaling, regularization and adding adversarial examples to the training data. A potential challenge might be that these measures impact accuracy of the model’s output, rendering the model useless for production. In that case, the monitoring of evasion attempts is even more important as a line of defense because measures to make the model more robust could not be applied.

Figure 8 provides an example of this two-step approach. It uses ART, but other tools such as Counterfit or Armory can also be used.

The model in this example is an image classification model (e.g., a CNN). The evasion attack is the Fast Gradient Sign Method (FGSM) attack. This attack adds noise to the picture (a car wheel in Figure 8), where the “noisy” picture is invisible to the human eye. Due to the noise, the model classified the altered picture as a disc brake.

Figure 8 shows the attack and the picture. (Note that this is not the complete code, but only the attack-related code.)

Figure 8: Example of a FGSM Attack Using ART

Example of a FGSM Attack Using ART

```
# import necessary libs
from tensorflow import keras
from art.attacks.evasion import FastGradientMethod

# Load image and model
image = load_image('car_wheel.jpg')
model = load_model('wheel_classifier.h5')
classifier = KerasClassifier(model=model, clip_values=(0,255))

# Specify FGSM attack
attacker = FastGradientMethod(eps=2)
adversarial_image = attacker.generate(image)

# Get predictions of original and adversarial images
prediction_org = classifier.predict(image)
prediction_adv = classifier.predict(adversarial_image)
```

Original Picture



prediction_org
95% car wheel

Altered Picture



prediction_adv
31% disc brake

Source: Gartner
778428_C

Gartner

There are different defenses against FGSM, such as spatial smoothing and feature scaling. Spatial smoothing averages the values of the neighboring pixels, which reduces the impact of small changes (perturbations) in the individual pixels. Feature scaling helps to reduce large variations in feature values (which is important as larger variations provide more possibilities to alter the picture). For example, the code snippet in Figure 9 shows how to apply spatial smoothing and feature scaling to the adversarial picture. This makes the picture look more blurry. The model's prediction of the smoothed adversarial picture shows pretty much the same accuracy (94%) as the original picture (95%).

Figure 9: Example of FGSM Defense

Example of FGSM Defense

```

filtered_adversarial_image = SpatialSmoothing(window_size=7)(adversarial_image)
filtered_adversarial_image = FeatureSqueezing(bit_depth=5)(filtered_adversarial_image, clip_values=(0, 255))
prediction = classifier.predict(filtered_adversarial_image)

```

The Model's Prediction of the Smoothed Adversarial Picture

prediction
94% car wheel

Source: Gartner
778428_C

Gartner

The common security control to protect against evasion attacks is to monitor evasion attempts.

As with a membership inference attack, a potential evasion attack indicator is a spike in the ML model inference requests, where the spike is outside the normal model inference pattern. There are different options to monitor a spike, such as:

- ML monitoring tool, which is triggered by the spike in inferences: Such a tool can be operated by, for example, the ML operations (MLOps) team, the network operations team or the IT infrastructure operations team.
- SIEM tool, which is triggered by the spike in API calls. The SIEM tool can be operated by the security operations team, for instance.

The monitoring team should alert the D&A team (e.g., the data scientist) when a spike is triggered. The data scientist should further investigate the underlying cause of the spike.

Extraction

As with model evasion, the security controls to protect against extraction attacks are twofold, as shown in Table 5.

Table 5: Security Controls to Protect Against Extraction Attacks

Control	Common or ML	D&A	Security
Improve model robustness	ML		
Monitor extraction attempts	Common		
 = Shared Responsibility  = Teams Responsibility  = No Responsibility			

Source: Gartner (May 2023)

The ML-specific security control for protecting against extractions is to improve model robustness.

As described earlier, there are factors which contribute to successful model extraction. Reducing those factors can improve the model’s robustness against model extraction, such as:

- Some algorithms are easier to extract than others: Replace the easier algorithm with a more complex algorithm, if feasible, without reducing the model’s output metric (such as accuracy). As an example, there are different classification algorithms, such as logistic regression, decision tree, random forest and neural networks. Logistic regression and decision trees are less complex than a random forest or a neural network.
- Limit the model’s output details in production: These details might be required during training in order to tune the model. However, in production, limit the model details in production to the bare minimum (for example, labels instead of detailed confidence levels).

- Don't expose model information (e.g., algorithm, architecture, hyperparameters) to the outside world, if not required: Note that just obscuring this information as the only line of defense may be not enough to protect against extraction attacks. This is because an attacker may still be able to get information about the model. An example would be when the model is built and/or runs on an MLaaS platform and it provides information to the outside world how the ML services work (for example, which algorithms are used, the default hyperparameter settings and the model's output details during inference).

The common security control for protecting against extractions is to monitor extraction attempts.

As with a membership inference and model evasion attacks, a spike in the ML model requests could be an indicator of such an attack — in particular, when the spike is outside the normal model inference pattern. As described in the model evasion section, there are different options to monitor a spike, such as ML monitoring or SIEM tools.

The D&A team (such as the data scientist) should be alerted by the monitoring team in case of a spike and then further investigate its underlying cause.

ML System Code

This section first describes the possible attack vectors related to the ML system code component. It then describes where these attacks can happen in the ML pipelines and, finally, which security controls can be applied.

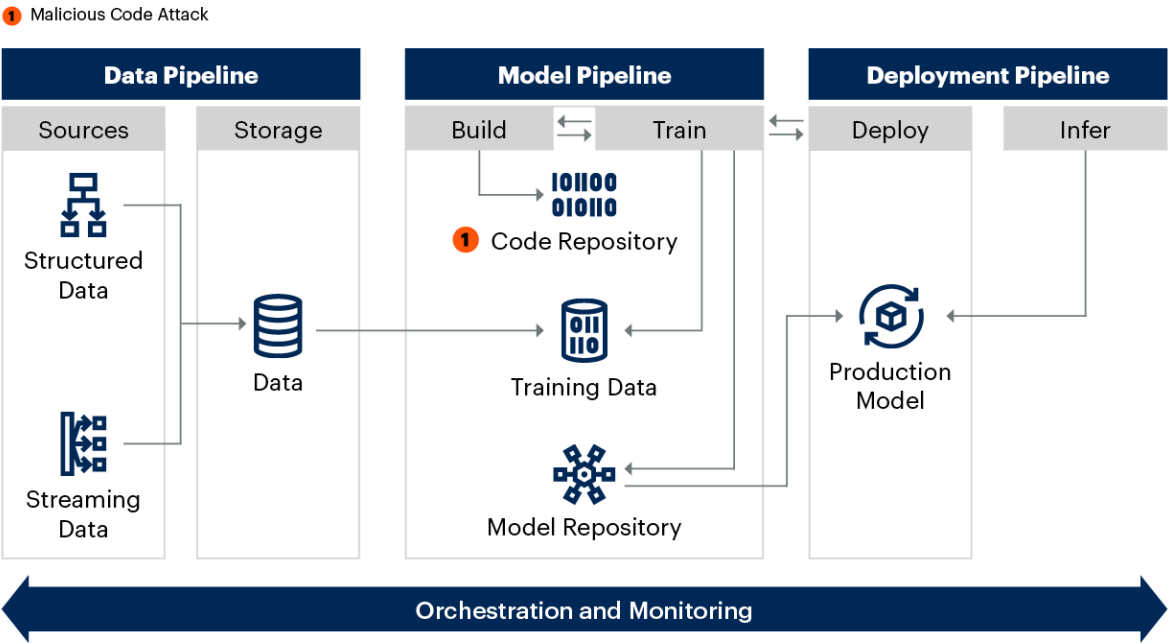
An ML system can be vulnerable when part of its code contains vulnerabilities or malicious content. For instance, the lack of secure coding standards can lead to unsecure code, such as plain text credentials in source code files. Another example is the use of unvalidated open-source or third-party components, such as pretrained models, code libraries or frameworks. This becomes more challenging when these components are in binary format only (for example, Python code that is serialized in a pickle file or HDF5/h5 file). Malicious components can leak user credentials or contain code injections to open a backdoor connection as soon as the serialized code is loaded.

Where Can These Attacks Happen?

Figure 10 shows that the ML system code attacks can happen in the code repository (part of the model pipeline), as represented by "1."

Figure 10: Malicious Code Attacks

Malicious Code Attacks



Source: Gartner
778428_C

Code Security Controls

The security controls to protect against malicious code are described in Table 6.

Table 6: Security Controls to Protect Against Code Attacks

Control	Common or ML	D&A	Security	SW Dev
Code review	Common	◐	○	◐
Code testing	Common	◐	◐	◐
◐ = Shared Responsibility ◑ = Teams Responsibility ○ = No Responsibility				

Source: Gartner (May 2023)

The common security controls for protecting against code attacks are code review and code testing.

The *code review control* is important to ensure that the ML system code does not contain vulnerabilities, which can be exploited — for example, user credentials (user ID and password) for a database connection string or the secrets of API keys as plain text in the code. Open-source scanners (e.g., GitHub secret scanning or NB Defense) or commercial code scanners can be used to scan the code for these types of vulnerabilities.

Depending on the software development expertise and experience level, the D&A team (data scientist) can carry out the code reviews within the team. Alternatively, the team may want to work with the software development (SW) team for this code review. Gartner recommends working with the SW team regardless — in particular when the model is part of an end-to-end workflow and the SW team is responsible for the back-end and front-end integration of the model's code.

For *code testing*, Gartner recommends:

- Only use open-source or third-party components that your organization trusts (e.g., the third-party is trusted, and as such, their code is trusted).
- In case unknown open-source or third-party components need to be used, review the actual code.

However, this may not always be possible in case of binary files or when the code is too large (for example, ML frameworks), which takes considerable effort and skills. In that case, run the code in a separate (virtual) test environment (such as a virtual machine or container). The D&A team should work with the security team to verify whether the code has indications of malicious intent. For instance, are there any inbound and/or outbound connections initiated by the code, and if so, are they legitimate? A connection could be an indication of a backdoor connection to leak information. Or — as another example — does the code execute malicious instructions to encrypt the files as a ransomware attack?

Note that the recommendations are from a security perspective and do not describe other code-testing practices, such as unit and integration testing. To ensure consistency and standardization of code testing across the organization, the D&A team should work with the SW team. This is especially important in case the D&A team lacks the expertise to carry out code tests.

Strengths

When designing and operating the security of ML systems, D&A technical professionals should consider the following benefits:

- **Take an ML system approach:** Taking the ML system approach ensures that all components (data, model and code) are covered from the ML risk perspective. Mapping the ML system components across the ML pipeline also ensures that security controls are implemented at the right stage in the pipeline.
- **Use a risk-based approach:** This will help to focus on the ML risks, which are relevant to the context in which the ML system is used and justifies the investments in ML security. Adversarial attacks can be ignored when they don't apply to the ML system or when the ML system is not vulnerable.
- **Incorporate ML security into shared security responsibilities:** This will spread out the work, help with prioritization and reduce duplication of efforts. Shared security controls cover multiple attacks, such as the monitoring implementation for model inference requests, which captures indicators of a possible model inference, model evasion or model extraction attack.
- **Practice cross-departmental responsibility:** This will create opportunities for peer learning and collaboration as well as more robust defense. The shared security controls are cross-departmental and make ML security a shared responsibility across the organization, or at least for the teams involved. This ensures that the roles and responsibilities are assigned to the right team with the right skills, knowledge and experience. It will also lead to better standardization and consistency across the organization in terms of solutions and processes.

Weaknesses

When designing and operating the security of ML systems, D&A technical professionals should consider the following cautions:

- **Tools and practices are immature:** For the last 5 years, ML adversarial attacks have been mainly studied in academia and in R&D departments. Most of these attacks are now available in open-source toolkits (such as ART and Counterfit). These toolkits can be used to test or detect adversarial attacks. However, the market of commercial solutions to detect and protect against ML adversarial attacks is immature and nascent. Startups are emerging (see list of Representative Vendors), but it is still early days.

- **Challenges with MLOps integration:** Most of the tools and solutions are point solutions and integration options with other (commercial) MLOps/DSML platforms are lacking. This makes it challenging to standardize, for example, ML system monitoring using one platform. This will become more challenging when new attack vectors emerge and/or more complex models are being used (for example, large language models [e.g., ChatGPT]). It also makes it harder to implement automated workflows to handle triggered alerts.
- **Steep learning curve:** ML adversarial attacks are complex and new territory for most D&A technical professionals (such as data scientists, data engineers and ML engineers). Yet, D&A professionals need an understanding of the intricacies of these attacks to assess whether their ML systems are vulnerable and which controls can be implemented to offset these vulnerabilities. This may require a steep learning curve, depending on the existing knowledge of the D&A technical professional.

Guidance

Organizations keen on implementing ML security should take a risk-based approach and work cross-departmental. In addition, the D&A team should ensure that its resources, such as data scientists and data engineers, gain the necessary knowledge and experience regarding ML security.

Follow this guidance to implement ML security successfully.

Take a Risk-Based Approach

Prioritize ML risks.

As described earlier in this document, a risk can materialize when all of the following conditions are true:

- An attack vector is aimed at the ML system component (or components).
- There is an impact when the attack is successful.
- This component is vulnerable for the attack.
- The attacker (internal or external) is able to exploit the vulnerability.

To prioritize the risks, focus on the attack vectors that are applicable to your ML system and those to which the ML system is vulnerable.

For example, a model evasion attack aimed at sound evasion can be ignored if the ML system is not built to process sounds. Alternatively, the ML system might be vulnerable for an adversarial attack, but the impact (in case of a successful attack) is assessed as low. Therefore, no security controls will be implemented because this may not outweigh the costs (compared with the low impact). This is still a risk, but a residual risk, and is accepted as such.

Focus on Vulnerability Control

Focus on controlling vulnerabilities to mitigate risks.

Attack vectors and attackers cannot be controlled. Controlling the impact of an attack can be difficult, as is defining the impact in terms of quantity or even qualitatively (for example, the impact/price of reputational loss).

Vulnerabilities can be assessed and reduced by means of security controls, which in turn reduce the ability for attackers to exploit the vulnerabilities. As a result, they cannot execute the attack, and the risk is mitigated.

Prioritize Security Controls

Prioritize security controls to effectively mitigate risks from the start.

Start with:

- Shared security controls (such as data access management and model monitoring) because they cover multiple attacks: For instance, indicators of a possible model inference, model evasion or model extraction attack can be captured by the implementation of a model monitoring control.
- Security controls that are relevant regardless of the stage, such as data protection for privacy data: This data needs to be protected (for example, because of legislation), regardless of whether the ML model is developed, trained, tested, in Q&A or put into production.

Teamwork Is Key

ML security is a shared responsibility and requires cross-departmental involvement.

ML security is not one team's responsibility. Security controls are cross-departmental and a shared responsibility across the organization, or at least for the teams involved. This ensures that the roles and responsibilities are assigned to the right team with the right skills, knowledge and experience. It will also lead to better standardization and consistency across the organization in terms of solutions and processes.

For the shared security controls, define the RACI (responsible, accountable, consulted and informed) matrix with roles and responsibilities for each involved team. Define a similar RACI matrix for the D&A team with regard to the team's ML-specific controls (for example, testing the model for data poisoning).

Climb the Learning Curve

ML adversarial attacks are complex and must be understood to detect and protect against them.

ML adversarial attacks are complex and new territory for most D&A technical professionals (such as data scientists, data engineers and ML engineers). An understanding of the intricacies of these attacks is necessary to assess whether ML systems are vulnerable and to determine which controls can be implemented to offset these vulnerabilities. For most D&A technical professionals, this knowledge must be acquired and may require a steep learning curve.

The D&A team members responsible to assess and protect against adversarial attacks should use open-source tools to kick-start their learning curve.

The D&A team should start by engaging with commercial vendors to get a better understanding of their solution and how it can help the team.

Representative Vendors

The market for vendors with a specific focus on the detection and protection of ML adversarial attacks is nascent. Startups are emerging with commercial solutions to address ML adversarial attacks, such as detection and protection. Some vendors provide part of their solution as an open-source toolkit, incorporate existing open-source toolkits (such as ART) or integrate their solution with commercial data science and machine learning (DSML) platform providers.

Representative vendors are [HiddenLayer](#), [Protect AI](#), [TrojAI](#), and [Robust Intelligence](#).

Evidence

¹ 2021 Gartner AI in Organizations Survey. This survey was conducted to understand the keys to successful AI implementations and the barriers to the operationalization of AI. The research was conducted online from October through December 2021 among 699 respondents from organizations in the U.S., Germany and the U.K. Quotas were established for company size and for industries to ensure a good representation across the sample. Organizations were required to have developed AI or intended to deploy AI within the next three years. Respondents were required to be part of the organization's corporate leadership or report into corporate leadership roles, and have a high level of involvement with at least one AI initiative. Respondents were also required to have one of the following roles when related to AI in their organizations. determine AI business objectives, measure the value derived from AI initiatives or manage AI initiatives development and implementation. The survey was developed collaboratively by a team of Gartner analysts and Gartner's Research Data, Analytics and Tools team. Disclaimer: Results of this survey do not represent global findings or the market as a whole, but reflect the sentiments of the respondents and companies surveyed.

Recommended by the Author

Some documents may not be available as part of your current Gartner subscription.

[Top Strategic Technology Trends for 2023: AI Trust, Risk and Security Management](#)

[Market Guide for AI Trust, Risk and Security Management](#)

[Guide to Data Security Concepts](#)

[Roles and Skills to Support Advanced Analytics and AI Initiatives](#)

[Data Engineering Essentials, Patterns and Best Practices](#)

© 2023 Gartner, Inc. and/or its affiliates. All rights reserved. Gartner is a registered trademark of Gartner, Inc. and its affiliates. This publication may not be reproduced or distributed in any form without Gartner's prior written permission. It consists of the opinions of Gartner's research organization, which should not be construed as statements of fact. While the information contained in this publication has been obtained from sources believed to be reliable, Gartner disclaims all warranties as to the accuracy, completeness or adequacy of such information. Although Gartner research may address legal and financial issues, Gartner does not provide legal or investment advice and its research should not be construed or used as such. Your access and use of this publication are governed by [Gartner's Usage Policy](#). Gartner prides itself on its reputation for independence and objectivity. Its research is produced independently by its research organization without input or influence from any third party. For further information, see "[Guiding Principles on Independence and Objectivity](#)." Gartner research may not be used as input into or for the training or development of generative artificial intelligence, machine learning, algorithms, software, or related technologies.

Table 1: High-Level Explanation of the ML Adversarial Attack Vectors

Attack	Explanation
Poisoning	Deliberate insertion of malicious data into the training set
Inference	Determine whether a data point was in the training set
Extraction	Steal a model by creating a functionally equivalent surrogate of the original model
Evasion	Provide the model a fabricated input in order to manipulate the model's output














Source: Gartner (May 2023)

Table 2: Security Controls to Protect Against Data Poisoning

Control	Common or ML	D&A		Security	
Implement data access management	Common		<div></div>		<div></div>
Implement review governance for third-party (open) data	Common		<div></div>		<div></div>
Test model for data poisoning	ML		<div></div>		<div></div>
<div></div> = Shared Responsibility <div></div> = Teams Responsibility <div></div> = No Responsibility					








Source: Gartner (May 2023)

Table 3: Security Controls to Protect Against Membership Inference Attacks

Control	Common or ML	D&A	Security
Use data masking or synthetic data	Common		
Use differential privacy	Common		
Monitor inference attempts	Common		
Prevent overfitting	ML		
Evaluate and test MIA	ML		
 = Shared Responsibility  = Teams Responsibility  = No Responsibility			

Source: Gartner (May 2023)

Table 4: Security Controls to Protect Against Evasion Attacks

Control	Common or ML	D&A	Security
Improve model robustness	ML		
Monitor evasion attempts	Common		
 = Shared Responsibility  = Teams Responsibility  = No Responsibility			

Source: Gartner (May 2023)

Table 5: Security Controls to Protect Against Extraction Attacks

Control	Common or ML	D&A	Security
Improve model robustness	ML	●	○
Monitor extraction attempts	Common	◐	◐
◐ = Shared Responsibility ● = Teams Responsibility ○ = No Responsibility			

Source: Gartner (May 2023)

Table 6: Security Controls to Protect Against Code Attacks

Control	Common or ML	D&A	Security	SW Dev
Code review	Common	◐	○	◐
Code testing	Common	◐	◐	◐
◐ = Shared Responsibility ● = Teams Responsibility ○ = No Responsibility				

Source: Gartner (May 2023)