multiple distance metrics. Some notable attacks include:

- DeepFool is an untargeted evasion attack for ℓ_2 norms, which uses a linear approximation of the neural network to construct the adversarial examples [257].
- The Carlini-Wagner attack uses multiple objectives that minimize the loss or logits on the target class and the distance between the adversarial example and original sample. The attack is optimized via the penalty method [65] and considers three distance metrics to measure the perturbations of adversarial examples: ℓ_0, ℓ_2, and ℓ_∞. The attack has been effective against the defensive distillation defense [284].
- The Projected Gradient Descent (PGD) attack [232] minimizes the loss function and projects the adversarial examples to the space of allowed perturbations at each iteration of gradient descent. PGD can be applied to the ℓ_2 and ℓ_∞ distance metrics for measuring the perturbation of adversarial examples.

Universal evasion attacks. Moosavi-Dezfooli et al. [256] showed how to construct small universal perturbations (with respect to some norm) that can be added to most images and induce a misclassification. Their technique relies on successive optimization of the universal perturbation using a set of points sampled from the data distribution. This is a form of FUNCTIONAL ATTACK. An interesting observation is that the universal perturbations generalize across deep network architectures, suggesting similarity in the decision boundaries trained by different models for the same task.

Physically realizable attacks. These are attacks against ML systems that can be implemented feasibly in the physical world [21, 200, 227]. One of the first instances was the attack on facial recognition systems by Sharif et al. [332]. The attack can be realized by printing a pair of eyeglass frames, which misleads facial recognition systems to either evade detection or impersonate another individual. Eykholt et al. [122] proposed an attack to generate robust perturbations under different conditions, resulting in adversarial examples that can evade vision classifiers in various physical environments. The attack is applied to evade a road sign detection classifier by physically applying black and white stickers to the road signs. The ShapeShifter [81] attack was designed to evade object detectors, which is a more challenging problem than attacking image classifiers since the attacker needs to evade the classification in multiple bounding boxes with different scales. This attack also requires the perturbation to be robust enough to survive real-world distortions due to different viewing distances, angles, lighting conditions, and camera limitations.

Other data modalities. In computer vision applications, adversarial examples are often designed to be imperceptible to humans. Therefore, the perturbations introduced by attackers need to be so small that a human correctly recognizes the images, while the ML classifier is tricked into changing its prediction. Alternatively, there may be a trigger object in the image that is still imperceptible or innocuous to humans but causes the model to misclassify. The concept of adversarial examples has been extended to other domains, such as audio, video, NLP, and cybersecurity. In some of these settings, there are additional constraints that need to be respected by adversarial examples, such as text semantics in NLP and the application constraints in cybersecurity. Several representative works include:

- **Audio:** Carlini and Wagner [66] showed a targeted attack on models that generate text from speech. They can generate an audio waveform that is very similar to an existing one but that can be transcribed to any text of the attacker's choice.
- **Video:** Adversarial evasion attacks against video classification models can be split into sparse attacks that perturb a small number of video frames [401] and dense attacks that perturb all of the frames in a video [215]. The goal of the attacker is to change the classification label of the video.
- **Text:** Jia and Liang [185] developed a methodology for generating adversarial text examples. This pioneering work was followed by many advances in developing adversarial attacks on natural language processing (NLP) models (see a comprehensive survey on the topic [438]). La Malfa and Kwiatkowska [202] proposed a method for formalizing perturbation definitions in NLP by introducing the concept of semantic robustness. The main challenges in NLP are that the domain is discrete rather than continuous (e.g., image, audio, and video classification), and adversarial examples need to respect text semantics. These challenges are illustrated by the recent ASCII-art attack [186] against chatbots. An ASCII-art illustration of a forbidden term tricks the chatbot into providing the harmful information even when the chatbot correctly censors the plain English word. The semantic distance between the two prompts is precisely zero, and both of them should have been treated the same.
- **Cybersecurity:** In cybersecurity applications, adversarial examples must respect the constraints imposed by the application semantics and feature representation of cyber data, such as network traffic or program binaries. FENCE is a general framework for crafting white-box evasion attacks using gradient optimization in discrete domains and supports a range of linear and statistical feature dependencies [88]. FENCE has been applied to two network security applications: malicious domain detection and malicious network traffic classification. Sheatsley et al. [334] proposed a method that learns the constraints in feature space using formal logic and crafts adversarial examples by projecting them onto a constraint-compliant space. They applied the technique to network intrusion detection and phishing classifiers. Both papers observed that attacks from continuous domains cannot be readily applied in constrained environments, as they result in infeasible adversarial examples. Pierazzi et al. [295] discussed the difficulty of mounting feasible evasion attacks in cybersecurity due to constraints in feature space and the challenge of mapping attacks from feature space to problem space. They formalized evasion attacks in problem space and constructed feasible adversarial examples for Android malware.

2.2.2. Black-Box Evasion Attacks

[**NISTAML.025**] [Back to Index]

Black-box evasion attacks are designed under a realistic adversarial model in which the attacker has no prior knowledge of the model architecture or training data. Instead, the adversary can interact with a trained ML model by querying it on various data samples and obtaining the model's predictions. Similar APIs are provided by MLaaS offered by public cloud providers, in which users can obtain the model's predictions on selected queries without information about how the model was trained. There are two main classes of black-box evasion attacks in the literature:

- **Score-based attacks:** In this setting, attackers obtain the model's confidence scores or logits and can use various optimization techniques to create the adversarial examples. A popular method is zeroth-order optimization, which estimates the model's gradients without explicitly computing derivatives [80, 173]. Other optimization techniques include discrete optimization [254], natural evolution strategies [172], and random walks [262].
- **Decision-based attacks:** In this more restrictive setting, attackers only obtain the final predicted labels of the model. The first method for generating evasion attacks was the Boundary Attack based on random walks along the decision boundary and rejection sampling [47], which was extended with an improved gradient estimation to reduce the number of queries in the HopSkipJumpAttack [79]. More recently, several optimization methods search for the direction of the nearest decision boundary (e.g., the OPT attack [86]), use sign SGD instead of binary searches (e.g., the Sign-OPT attack [87]), or use Bayesian optimization [344].

The primary challenge in creating adversarial examples in black-box settings is reducing the number of queries to the ML models. Recent techniques can successfully evade the ML classifiers with a relatively small number of queries, typically less than 1000 [344].

2.2.3. Transferability of Attacks

Another method for generating adversarial attacks under restrictive threat models involves transferring an attack crafted on a different ML model. Typically, an attacker trains a substitute ML model, generates white-box adversarial attacks on the substitute model, and transfers the attacks to the target model. Various methods differ in how the substitute models are trained. For example, Papernot et al. [282, 283] train the substitute model with score-based queries to the target model, while several papers train an ensemble of models without explicitly querying the target model [218, 377, 397].

Attack transferability is an intriguing phenomenon, and existing literature attempts to understand the fundamental reasons why adversarial examples transfer across models. Several papers have observed that different models learn intersecting decision boundaries in both benign and adversarial dimensions, which leads to better transferability [144, 256, 377]. Demontis et al. [104] identified two main factors that contribute to attack transferability for both evasion and poisoning: the intrinsic adversarial vulnerability of the target model and the complexity of the surrogate model used to optimize the attack. EXPECTATION OVER TRANSFORMATION aims to make adversarial examples sustain image transformations that occur in the real world, such as angle and viewpoint changes [21].

2.2.4. Evasion attacks in the real world

While many of the attacks discussed in this section were demonstrated only in research settings, several evasion attacks have been demonstrated in the real world, and we discuss prominent instances in face recognition systems, phishing webpage detection, and malware classification.

Face recognition systems used for identity verification have been the target of adversarial evasion attacks, as they constitute an entry point to critical systems and enable users to commit financial fraud. During the last half of 2020, the ID.me face recognition service found more than 80,000 attempts of users attempting to fool their ID verification steps used by multiple state workforce agencies [276]. These attacks included people wearing masks, using deepfakes, or using images or videos of other people. The intent was to fraudulently claim unemployment benefits provided during COVID relief efforts. Later in 2022, according to US federal prosecutors, a New Jersey man was able to verify fake driver's licenses through ID.me as part of a US$ 2.5M unemployment-fraud scheme. This time, the suspect used various wigs to evade the face recognition system [156].

Another case study of real-world evasion attacks reported by Apruzzese et al. [17] is an attack against a commercial phishing webpage detector. The ML phishing detector is an ensemble of multiple models that analyze different aspects of the image to determine if it is a phishing attempt. Inputs that are marked uncertain by the model are triaged to security analysts. Out of 4600 samples marked uncertain by the ML image classification system, the authors identified 100 adversarial examples. Interestingly, a manual analysis of these adversarial examples revealed that attackers do not employ optimization-based attacks, but rather utilize relatively simple methods for evasion, such as image cropping, masking, or blurring techniques.

Other examples of evasion attacks demonstrated by researchers in malware classification are cataloged in the MITRE Adversarial Threat Landscape for Artificial-Intelligence Systems (ATLAS) knowledge base [248]. Palo Alto Networks reported evasion attacks against a deep learning detector for malware command-and-control traffic, and a botnet Domain Generation Algorithm (DGA) detector. An instance of a universal evasion attack was discovered against Cylance's AI malware detection model. Researchers also evaded ProofPoint's email protection system by training a shadow ML model and using the insights from that to attack the real system. These are demonstrations of evasion vulnerabilities by researchers, but did not result in attacks in the wild.

2.2.5. Mitigations

Mitigating evasion attacks is challenging because adversarial examples are widespread in a variety of ML model architectures and application domains. Possible explanations for the existence of adversarial examples are that ML models rely on non-robust features that are not aligned with human perception in the computer vision domain [174]. In the last few years, many of the proposed mitigations against adversarial examples have been ineffective against stronger attacks. Furthermore, several papers have performed extensive evaluations and defeated a large number of proposed mitigations:

- Carlini and Wagner showed how to bypass 10 methods for detecting adversarial examples and described several guidelines for evaluating defenses [64]. Recent work shows that detecting adversarial examples is as difficult as building a defense [373]. Therefore, this direction for mitigating adversarial examples is similarly challenging as designing defenses.
- The Obfuscated Gradients attack [20] was specifically designed to defeat several proposed defenses that rely on masking gradients to protect against optimization-based attacks. It relies on a new technique, Backward Pass Differentiable Approximation, which approximates the gradient during the backward pass of backpropagation, and was shown to bypass several proposed defenses based on gradient masking.
- Tramèr et al. [375] described a methodology for designing adaptive attacks against proposed defenses and circumvented 13 existing defenses. They advocate for designing adaptive attacks to test newly proposed defenses rather than merely testing the defenses against well-known attacks.

From the wide range of proposed defenses against adversarial evasion attacks, three main classes have proven to be resilient and have the potential to provide mitigation against evasion attacks:

1. **Adversarial training:** Introduced by Goodfellow et al. [144] and further developed by Madry et al. [232], adversarial training is a general method that augments training data with adversarial examples generated iteratively during training using their correct labels. The stronger the adversarial attacks for generating adversarial examples are, the more resilient the trained model becomes. Adversarial training results in models with more semantic meaning than standard models [379], but this benefit usually comes at the cost of decreased model accuracy on clean data. Additionally, adversarial training is expensive due to the iterative generation of adversarial examples during training.
2. **Randomized smoothing:** Proposed by Lecuyer et al. [207] and further improved by

Cohen et al. [94], randomized smoothing is a method that transforms any classifier into a certifiable robust smooth classifier by producing the most likely predictions under Gaussian noise perturbations. This method results in provable robustness for ℓ_2 evasion attacks, even for classifiers trained on large-scale datasets, such as ImageNet. Randomized smoothing typically provides certified prediction to a subset of testing samples, the exact number of which depends on factors such as the size of the potential perturbations or the characteristics of the training data and model. Recent results have extended the notion of certified adversarial robustness to ℓ_2-norm bounded perturbations by combining a pretrained denoising diffusion probabilistic model and a standard high-accuracy classifier [62]. Li et al. [211] developed a taxonomy for the robustness verification and training of representative algorithms. They also revealed the characteristics, strengths, limitations, and fundamental connections among these approaches, along with theoretical barriers facing the field.

3. **Formal verification:** Another method for certifying the adversarial robustness of a neural network is based on techniques from FORMAL METHODS. Reluplex uses satisfiability modulo theories (SMT) solvers to verify the robustness of small feed-forward neural networks [191]. AI^2 is the first verification method applicable to convolutional neural networks using abstract interpretation techniques [136]. These methods have been extended and scaled up to larger networks in follow-up verification systems, such as DeepPoly [346], ReluVal [394], and Fast Geometric Projections (FGP) [131]. Formal verification techniques have significant potential for certifying neural network robustness but are limited by their lack of scalability, computational cost, and restriction in the type of supported algebraic operations such as addition, multiplication, etc.

All of these proposed mitigations exhibit inherent trade-offs between robustness and accuracy, and they come with additional computational costs during training. Therefore, designing ML models that resist evasion while maintaining accuracy remains an open problem. See Section 4.1.1 for further discussion on these trade-offs.

2.3. Poisoning Attacks and Mitigations

Poisoning attacks are broadly defined as adversarial attacks during the training stage of the ML algorithm. The first known poisoning attack was developed for worm signature generation in 2006 [291]. Since then, poisoning attacks have been studied extensively in several application domains: computer security (for spam detection [269], network intrusion detection [384], vulnerability prediction [318], malware classification [329, 412]), computer vision [137, 148, 330], natural language processing (NLP) [82, 213, 388], and tabular data in healthcare and financial domains [179]. Recently, poisoning attacks have gained more attention in industry applications as well [199]. They can even be orchestrated at scale so that an adversary with limited financial resources could control a fraction of the public datasets used for model training [57].

Poisoning attacks are powerful and can cause availability or integrity violations. Availability poisoning attacks typically cause indiscriminate degradation of the ML model on all samples, while targeted and backdoor poisoning attacks induce integrity violations on a small set of target samples. Poisoning attacks leverage a wide range of adversarial capabilities (e.g., data poisoning, model poisoning, label control, source code control, and test data control), resulting in several subcategories of poisoning attacks. They have been developed in white-box [40, 179, 412], gray-box [179], and black-box settings [39].

This section describes availability poisoning, targeted poisoning, backdoor poisoning, and model poisoning attacks classified according to their adversarial objective. For each poisoning attack category, techniques for mounting the attacks, existing mitigations, and their limitations are also discussed. The classification of poisoning attacks in this document is inspired by the framework developed by Cinà et al. [91], which includes additional references to poisoning attacks and mitigations.

2.3.1. Availability Poisoning

[**NISTAML.013**] [Back to Index]

The first poisoning attacks discovered in cybersecurity applications were availability attacks against worm signature generation and spam classifiers, which indiscriminately degrade the performance of the entire ML model in order to effectively prevent its use. Perdisci et al. [291] generated suspicious flows with fake invariants that mislead the worm signature generation algorithm in Polygraph [270]. Nelson et al. [269] designed poisoning attacks against Bayes-based spam classifiers by generating training samples of ”spam” emails containing long sequences of words that appear in legitimate emails, degrading the performance of the spam classifier by inducing a higher rate of false positives. Both of these attacks were conducted under the white-box setting in which adversaries were aware of the ML training algorithm, feature representations, training datasets, and ML models. Availability poisoning attacks have also been proposed for ML-based systems that detect cybersecurity attacks against industrial control systems: such detectors are often retrained using data collected during system operation to account for plant operational drift of the monitored signals, creating opportunities for an attacker to mimic the signals of corrupted sensors at training time to poison the detector such that real attacks remain undetected at deployment time [198].

A simple black-box poisoning attack strategy is LABEL FLIPPING, in which an adversary generates training examples with incorrect or altered labels [39]. This method may require a large percentage of poisoning samples to mount an availability attack. These attacks can also be formulated through optimization-based methods, such as by solving a bilevel optimization problem to determine the optimal poisoning samples that will achieve the adversarial objective (i.e., maximize the hinge loss for a SVM [40] or maximize the mean square error [MSE] for regression [179]). Similar optimization-based availability poisoning attacks have been designed against linear regression [179] and neural networks [260], although these optimization-based attacks may require white-box access to the model and training data. In gray-box adversarial settings, the most popular method for generating availability poisoning attacks is transferability, in which poisoning samples are generated for a surrogate model and transferred to the target model [104, 358].

Clean-label poisoning [NISTAML.012] [Back to Index]. A realistic threat model for supervised learning is that of clean-label poisoning attacks, in which adversaries can only control the training examples but not their labels. This may arise in scenarios in which the labeling process is external to the training algorithm, as in malware classification where binary files can be submitted by attackers to threat intelligence platforms and labeling is performed using anti-virus signatures or other external methods. Clean-label availability attacks have been introduced for neural network classifiers by training a generative model and adding noise to training samples to maximize the adversarial objective [128]. A different approach for clean-label poisoning is to use gradient alignment and minimally modify the training data [129].

Availability poisoning attacks have also been designed for unsupervised learning against centroid-based anomaly detection [195] and behavioral clustering for malware [41]. In federated learning, an adversary can mount a model poisoning attack to induce availability violations in the globally trained model [123, 335, 336]. More details on model poisoning attacks are provided in Sec. 2.3.

Mitigations. Availability poisoning attacks are usually detectable by monitoring the standard performance metrics of ML models (e.g., precision, recall, accuracy, F1 scores, and area under the curve) as they cause a large degradation in the classifier metrics. However, detecting these attacks during the testing or deployment stages of ML may be less desirable, and many existing mitigations aim to proactively prevent these attacks during the training stage to generate robust ML models. Existing mitigations for availability poisoning attacks include:

- **Training data sanitization:** These methods leverage the insight that poisoned samples are typically different than regular training samples that are not controlled by adversaries. As such, data sanitization techniques are designed to clean the training set and remove the poisoned samples before the ML training is performed. Cretu et al. [96] proposed the first sanitization procedure for unlabeled datasets that relies on majority voting of multiple models trained on subsets of the training set. They apply the method to anomaly detection on network packets. Nelson et al. [269] introduced the Region of Non-Interest (RONI) method, which examines each sample and excludes it from training if the accuracy of the model decreases when the sample is added. Subsequently proposed sanitization methods improved upon these early approaches by reducing their computational complexity and considering other applications. Paudice et al. [289] introduced a method for label cleaning that was specifically designed for label-flipping attacks. Steinhardt et al. [354] proposed the use of outlier detection methods for identifying poisoned samples. Clustering methods have also been used to detect poisoned samples [203, 363]. Other work has suggested that computing the variance of predictions made by an ensemble of multiple ML models is an effective data sanitization method for network intrusion detection [384]. Once sanitized, datasets may be protected by cybersecurity mechanisms for provenance and integrity attestation [267].

- **Robust training:** An alternative approach to mitigating availability poisoning attacks is to modify the ML training algorithm to increase the robustness of the resulting model. The defender can train an ensemble of multiple models and generate predictions via model voting [37, 209, 395]. Several papers apply techniques from robust optimization, such as using a trimmed loss function [109, 179]. Rosenfeld et al. [314] proposed the use of randomized smoothing to add noise during training to provide protection against label-flipping attacks.

2.3.2. Targeted Poisoning

[**NISTAML.024**] [Back to Index]

In contrast to availability attacks, targeted poisoning attacks induce a change in the ML model's prediction on a small number of targeted samples. If the adversary can control the labeling function of the training data, then label-flipping is an effective targeted poisoning attack: the adversary simply inserts several poisoned samples with the target label, and the model will learn the wrong label. Therefore, targeted poisoning attacks are mostly studied in a clean-label setting in which the attacker does not have control over training data labels.

Several techniques for mounting clean-label targeted attacks have been proposed. Koh and Liang [196] showed how influence functions (i.e., a statistical method that determines the most influential training samples for a prediction) can be leveraged to create poisoned samples in the fine-tuning setting in which a pre-trained model is fine-tuned on new data. Suciu et al. [358] designed StingRay, a targeted poisoning attack that modifies samples in feature space and adds poisoned samples to each mini batch of training. An optimization procedure based on feature collision was crafted by Shafahi et al. [330] to generate clean-label targeted poisoning for fine-tuning and end-to-end learning. ConvexPolytope [444] and BullseyePolytope [4] optimized the poisoning samples against ensemble models, which offers better advantages for attack transferability. MetaPoison [166] uses a meta-learning algorithm to optimize the poisoned samples, while Witches' Brew [137] performs optimization by gradient alignment, resulting in a state-of-the-art targeted poisoning attack.

All of the above attacks impact a small set of targeted samples that are selected by the attacker during training, and they have only been tested for continuous image datasets (with the exception of StingRay, which requires adversarial control of a large fraction of the training set). Subpopulation poisoning attacks [180] were designed to poison samples from an entire subpopulation, defined by matching on a subset of features or creating clusters in representation space. Poisoned samples are generated using label-flipping (for NLP and tabular modalities) or a first-order optimization method (for continuous data, such as images). The attack generalizes to all samples in a subpopulation and requires minimal knowledge about the ML model and a small number of poisoned samples proportional to the subpopulation size.

Targeted poisoning attacks have also been introduced for semi-supervised learning algorithms [53], such as MixMatch [34], FixMatch [347], and Unsupervised Data Augmentation (UDA) [413] in which the adversary poisons a small fraction of the unlabeled training dataset to change the prediction on targeted samples at deployment time.

Mitigations. Targeted poisoning attacks are notoriously challenging to defend against. Jagielski et al. [180] showed an impossibility result for subpopulation poisoning attacks. To mitigate some of the risks associated with such attacks, model developers may protect training data through traditional cybersecurity measures such as access controls, use methods for data sanitization and validation, and use mechanisms for dataset provenance and integrity attestation [267]. Ma et al. [230] proposed the use of differential privacy (DP) as a defense (which follows directly from the definition of differential privacy), but differentially private ML models may also have lower accuracy than standard models, and the trade-off between robustness and accuracy needs to be considered in each application. See Section 4.1.1 for further discussion on the trade-offs between the attributes of Trustworthy AI systems.

2.3.3. Backdoor Poisoning

[**NISTAML.021, NISTAML.023**] [Back to Index]

Backdoor poisoning attacks are poisoning attacks that cause the targeted model to misclassify samples containing a particular BACKDOOR PATTERN or trigger. In 2017, Gu et al. [148] proposed BadNets, the first backdoor poisoning attack. They observed that image classifiers can be poisoned by adding a small patch trigger in a subset of images at training time and changing their label to a target class. The classifier learns to associate the trigger with the target class, and any image that includes the trigger or backdoor pattern will be misclassified to the target class at testing time. Concurrently, Chen et al. [84] introduced backdoor attacks in which the trigger is blended into the training data. Follow-up work introduced the concept of clean-label backdoor attacks [380] in which the adversary cannot change the label of the poisoned examples. Clean-label attacks typically require more poisoning samples to be effective, but the attack model is more realistic.

In the last few years, backdoor attacks have become more sophisticated and stealthy, making them harder to detect and mitigate. Latent backdoor attacks were designed to survive even upon model fine-tuning of the last few layers using clean data [420]. Backdoor Generating Network (BaN) [322] is a dynamic backdoor attack in which the location of the trigger changes in the poisoned samples so that the model learns the trigger in a location-invariant manner. Functional triggers (i.e., FUNCTIONAL ATTACK) are embedded throughout the image or change according to the input. Li et al. used steganography algorithms to hide the trigger in the training data [214] and introduced a clean-label attack that uses natural reflection on images as a backdoor trigger [223]. Wenger et al. [404] poisoned facial recognition systems by using physical objects as triggers, such as sunglasses and earrings. Architectural backdoor attacks [205] perform malicious modifications to the structure of an ML model during its training phase, which allows an attacker to manipulate the model's behavior when presented with specific triggers. These attacks require adversarial access to the model design or training environment and are applicable when model training is outsourced to a more powerful entity, such as a cloud service.

Other data modalities. While the majority of backdoor poisoning attacks are designed for computer vision applications, this attack vector has been effective in other application domains with different data modalities, such as audio, NLP, and cybersecurity settings.

- **Audio:** In audio domains, Shi et al. [341] showed how an adversary can inject an unnoticeable audio trigger into live speech, which is jointly optimized with the target model during training.
- **NLP:** In NLP, the construction of meaningful poisoning samples is more challenging as the text data is discrete, and the semantic meaning of sentences would ideally be preserved for the attack to remain unnoticeable. Recent work has shown that backdoor attacks in NLP domains are becoming feasible. For instance, Chen et al. [82] introduced semantic-preserving backdoors at the character, word, and sentence level for sentiment analysis and neural machine translation applications. Li et al. [213] generated hidden backdoors against transformer models using generative language models in three NLP tasks: toxic comment detection, neural machine translation, and question answering.
- **Cybersecurity:** Following early work on poisoning in cybersecurity [269, 291], Severi et al. [329] showed how AI explainability techniques can be leveraged to generate clean-label poisoning attacks with small triggers against malware classifiers. They attacked multiple models (i.e., neural networks, gradient boosting, random forests,

and SVMs) using three malware datasets: Ember for Windows PE file classification, Contagio for PDF file classification, and DREBIN for Android app classification. Jigsaw Puzzle [418] designed a backdoor poisoning attack for Android malware classifiers that uses realizable software triggers harvested from benign code.

Mitigations. The literature on backdoor attack mitigation is vast compared to other poisoning attacks. Below we discuss several classes of defenses, including data sanitization, trigger reconstruction, and model inspection and sanitization, and we also mention their limitations.

- **Training data sanitization:** Similar to poisoning availability attacks, training data sanitization can be applied to detecting backdoor poisoning attacks. For example, outlier detection in the latent feature space [157, 293, 378] has been effective for convolutional neural networks used for computer vision applications. Activation Clustering [76] clusters training data in representation space to isolate the backdoored samples in a separate cluster. Data sanitization achieves better results when the poisoning attack controls a relatively large fraction of training data but is not as effective against stealthy poisoning attacks. Overall, this leads to a trade-off between attack success and the detectability of malicious samples.
- **Trigger reconstruction:** This class of mitigations aims to reconstruct the backdoor trigger, assuming that it is at a fixed location in the poisoned training samples. NeuralCleanse by Wang et al. [390] developed the first trigger reconstruction approach and used optimization to determine the most likely backdoor pattern that reliably misclassifies the test samples. The initial technique has been improved to reduce performance time on several classes and simultaneously support multiple triggers inserted into the model [163, 411]. A representative system in this class is Artificial Brain Simulation (ABS) by Liu et al. [221], which stimulates multiple neurons and measures the activations to reconstruct the trigger patterns. Khaddaj et al. [193] developed a new primitive for detecting backdoor attacks and a corresponding effective detection algorithm with theoretical guarantees.
- **Model inspection and sanitization:** Model inspection analyzes the trained ML model before its deployment to determine whether it was poisoned. An early work in this space is NeuronInspect [168], which is based on explainability methods to determine different features between clean and backdoored models that are subsequently used for outlier detection. DeepInspect [78] uses a conditional generative model to learn the probability distribution of trigger patterns and performs model patching to remove the trigger. Xu et al. [416] proposed the Meta Neural Trojan Detection (MNTD) framework, which trains a meta-classifier to predict whether a given ML model is backdoored (or “Trojaned,” in the authors’ terminology). This technique is general and can be applied to multiple data modalities, such as vision, speech, tabular data, and NLP. Once a backdoor is detected, model sanitization can be performed via pruning [407], retraining [429], or fine-tuning [217] to restore the

model's accuracy.

- **Certified defenses:** Several methods for achieving certified defenses against data poisoning attacks have been proposed in the literature. BagFlip [440] is a model-agnostic defense that extends randomized smoothing [94] and combines training data bagging with adding noise to both training and testing samples. Deep Partition Aggregation [209] and Deep Finite Aggregation [396] are certified defenses that partition the training data into disjointed subsets and train an ensemble method on each partition to reduce the impact of poisoned samples. Recently, FCert [398] provides a certified defense against data poisoning in few-shot classification settings used for both vision and text data.

Most of these mitigations have been designed against computer vision classifiers based on convolutional neural networks using backdoors with fixed trigger patterns. Severi et al. [329] showed that some of the data sanitization techniques (e.g., spectral signatures [378] and Activation Clustering [76]) are ineffective against clean-label backdoor poisoning on malware classifiers. More recent semantic and functional backdoor triggers would also pose challenges to approaches based on trigger reconstruction or model inspection, which generally assume fixed backdoor patterns. The limitation of using meta classifiers for predicting a Trojaned model [416] is the high computational complexity of the training stage of the meta classifier, which requires training thousands of SHADOW MODEL. Additional research is required to design strong backdoor mitigation strategies that can protect ML models against this important attack vector without suffering from these limitations.

In cybersecurity, Rubinstein et al. [315] proposed an approach based on principal component analysis (PCA) to mitigate poisoning attacks against PCA subspace anomaly detection methods in backbone networks. It maximized median absolute deviation (MAD) instead of variance to compute principal components and used a threshold value based on Laplace distribution instead of Gaussian. Madani and Vlajic [231] built an autoencoder-based intrusion detection system, assuming that malicious poisoning attack instances were under 2%.

[193] provided a different perspective on backdoor mitigation by showing that backdoors are indistinguishable from naturally occurring features in the data if no additional assumptions are made about the attack. However, assuming that the backdoor creates the strongest feature in the data, the paper proposed an optimization technique to identify and remove the training samples that correspond to the backdoor.

Poison forensics [331] is a technique for root cause analysis that identifies malicious training samples and complements existing mitigations that are not always resilient in the face of evolving attacks. Poison forensics adds another layer of defense in an ML system: once a poisoning attack is detected at deployment time, poison forensics can trace back to the source of the attack in the training set.

2.3.4. Model Poisoning

[**NISTAML.011**, **NISTAML.026**] [Back to Index]

Model poisoning attacks attempt to directly modify the trained ML model to inject malicious functionality into it. In centralized learning, TrojNN [222] reverse engineers the trigger from a trained neural network and then retrains the model by embedding the trigger in external data to poison it. Most model poisoning attacks have been designed in the federated learning setting in which clients send local model updates to a server that aggregates them into a global model. Compromised clients can send malicious updates to poison the global model. Model poisoning attacks can cause both availability and integrity violation in federated models:

- Poisoning availability attacks that degrade the global model's accuracy have been effective, but they usually require a large percentage of clients to be under the control of the adversary [123, 335].
- Targeted model poisoning attacks induce integrity violations on a small set of samples at testing time. They can be mounted by a model replacement or model boosting attack in which the compromised client replaces the local model update according to the targeted objective [23, 35, 360].
- Backdoor model poisoning attacks introduce a trigger via malicious client updates to induce the misclassification of all samples with the trigger at testing time [23, 35, 360, 392]. Most of these backdoors are forgotten if the compromised clients do not regularly participate in training, but the backdoor becomes more durable if injected in the lowest utilized model parameters [441].

Supply chain model poisoning. [**NISTAML.05**] [**NISTAML.051**] [Back to Index] Model poisoning attacks are also possible in supply-chain scenarios in which models or components of the model provided by suppliers are poisoned with malicious code. Dropout Attack [425] is a recent supply-chain attack that shows how an adversary who manipulates the randomness used in neural network training (particularly in dropout regularization) might poison the model to decrease accuracy, precision, or recall on a set of targeted classes. See Supply Chain Attacks and Mitigations for additional discussion of supply-chain risks to GenAI models that are applicable to PredAI models too.

Mitigations. A variety of Byzantine-resilient aggregation rules have been designed and evaluated to defend federated learning from model poisoning attacks. Most of them attempt to identify and exclude the malicious updates when performing the aggregation at the server [8, 43, 51, 149, 242–244, 359, 423]. However, motivated adversaries can bypass these defenses by adding constraints to the attack generation optimization problem [23, 123, 335]. Gradient clipping and differential privacy have the potential to mitigate model poisoning attacks to some extent [23, 271, 360], but they usually decrease accuracy and do not provide complete mitigation.

For specific model poisoning vulnerabilities, such as backdoor attacks, there are some techniques for model inspection and sanitization (see Sec. 2.3.3). However, mitigating supply-chain attacks in which adversaries might control the source code of the training algorithm or the ML hyperparameters remains challenging. Program verification techniques used in other domains (e.g., cryptographic protocol verification [299]) might be adapted to this setting, but ML algorithms have intrinsic randomness and non-deterministic behavior, which enhances the difficulty of verification.

Designing ML models that are robust in the face of supply-chain model poisoning vulnerabilities is a critical open problem.

2.3.5. Poisoning Attacks in the Real World

As poisoning attacks require adversarial control over the ML training process, they are difficult to mount in the real world. Still, there are several examples of documented cases of real poisoning attacks targeting early AI chatbots, email spam filters, and malware classification services.

The first example of a real-world poisoning attack is the Tay.AI chatbot, a chatbot released by Microsoft on Twitter in 2016 [272]. After online interaction with users for less than 24 hours, the chatbot was poisoned and immediately taken down. At about the same time, there were several large-scale efforts to compromise Google's Gmail spam filter, in which attackers sent millions of emails to attempt to poison the Gmail spam classifier algorithm, enabling them to send other malicious emails without being detected [272]. MITRE ATLAS reported a poisoning incident on the VirusTotal threat intelligence service, in which similar, but not identical samples of a ransomware family were submitted through a popular virus sharing platform to cause the mis-classification of that particular ransomware family [248].

These incidents highlight the risks associated with online learning, as the Tay.AI chatbot was updated in real-time based on user interactions, and the Gmail spam filter and the VirusTotal malware classification system were continuously updated based on newly received samples. In all these incidents, attackers crafted poisoned samples after an initial model release, counting on the fact that models are continuously updated.

2.4. Privacy Attacks and Mitigations

[**NISTAML.03**] [Back to Index]

The seminal work of Dinur and Nissim [110] introduced DATA RECONSTRUCTION attacks, which seek to reverse-engineer private information about an individual user record or other sensitive input data from access to a trained model. More recently, data reconstruction attacks have been designed for binary and multi-class neural network classifiers [50, 152]. With MEMBERSHIP-INFERENCE ATTACK, an adversary can determine whether a particular record was included in the dataset used for training an ML model. Membership inference attacks were first introduced by Homer et al. [162] for genomic data. Recent literature focuses on membership attacks against ML models in mostly black-box settings in which adversaries have query access to a trained ML model [54, 342, 422]. Property inference attacks [19, 74, 134, 233, 361, 437] aim to extract global information about a training dataset, such as the fraction of training examples with a certain sensitive attribute. A different privacy violation for MLaaS is MODEL EXTRACTION attacks, which are designed to extract information about an ML model, such as its architecture or model parameters[3] [58, 70, 177, 376].

This section discusses privacy attacks related to data reconstruction, the memorization of training data, membership inference, property inference, and model extraction, as well as mitigations for some of these attacks and open problems in designing general mitigation strategies.

2.4.1. Data Reconstruction

[**NISTAML.032**] [Back to Index]

Data reconstruction attacks have the ability to recover an individual's data from released aggregate information. Dinur and Nissim [110] were the first to introduce reconstruction attacks that recover user data from linear statistics. Their original attack required an exponential number of queries for reconstruction, but subsequent work has shown how to perform reconstruction with a polynomial number of queries [116]. A survey of privacy attacks, including reconstruction attacks, is given by Dwork et al. [114]. More recently, the U.S. Census Bureau performed a large-scale study on the risk of data reconstruction attacks on census data [135], which motivated the use of differential privacy in the decennial release of the U.S. Census in 2020.

In the context of ML classifiers, Fredrickson et al. [130] introduced model inversion attacks that reconstruct class representatives from the training data of an ML model. While

[3] A privacy violation in this context describes a loss of confidential information about an ML model. If ML model leakage leads to further privacy violations for individuals (e.g., identity theft, sensitive data extraction), it may also be viewed as a cybersecurity-related privacy event. For further discussion on the relationship between privacy and cybersecurity risks, see NIST Privacy Framework, Version 1.0.

model inversion generates semantically similar images as those in the training set, it cannot directly reconstruct the training data of the model. Recently, Balle et al. [26] trained a reconstructor network that can recover a data sample from a neural network model, assuming that a powerful adversary has information about all other training samples. Haim et al. [152] showed how the training data of a binary neural network classifier can be reconstructed from access to the model parameters by leveraging theoretical insights about implicit bias in neural networks. This work has recently been extended to reconstruct training samples of multi-class multi-layer perceptron classifiers [50]. Attribute inference is another relevant privacy attack in which the attacker extracts a sensitive attribute of the training set, assuming partial knowledge about other features in the training data [184].

The ability to reconstruct training samples is partially explained by the tendency of neural networks to memorize their training data. Zhang et al. [431] discussed how neural networks can memorize randomly selected datasets. Feldman [126] showed that the memorization of training labels is necessary to achieve an almost optimal generalization error in ML. Brown et al. [48] constructed two learning tasks based on next-symbol prediction and cluster labeling in which memorization is required for high-accuracy learning. Feldman and Zhang empirically evaluated the benefit of memorization for generalization using an influence estimation method [127]. Data reconstruction attacks and their connection to memorization for generative AI are discussed in Sec. 3.3.2.

2.4.2. Membership Inference

[**NISTAML.033**] [Back to Index]

Membership inference attacks may expose private information about an individual, like reconstruction or memorization attacks, and are of great concern when releasing aggregate information or ML models trained on user data. In certain situations, determining that an individual is part of the training set already has privacy implications, such as in a medical study of patients with a rare disease. Moreover, membership inference can be used as a building block for mounting data extraction attacks [59, 63].

In membership inference, the attacker's goal is to determine whether a particular record or data sample was part of the training dataset used for the statistical or ML algorithm. These attacks were introduced by Homer et al. [162] for statistical computations on genomic data under the name *tracing attacks*. Robust tracing attacks have been analyzed when an adversary gains access to noisy statistical information about the dataset [115]. In the last five years, the literature has used the terminology *membership inference* for attacks against ML models. Most of the attacks in the literature are performed against deep neural networks that are used for classification [54, 89, 208, 342, 421, 422]. Similar to other attacks in AML, membership inference can be performed in white-box settings [208, 264, 317] in which attackers have knowledge of the model's architecture and parameters, but most of the attacks have been developed for black-box settings in which the adversary generates queries to the trained ML model [54, 89, 342, 421, 422].

The attacker's success in membership inference has been formally defined using a cryptographically inspired privacy game in which the attacker interacts with a challenger and needs to determine whether a target sample was used in training the queried ML model [183, 321, 422]. In terms of techniques for mounting membership inference attacks, the loss-based attack by Yeom et al. [422] is one of the most efficient and widely used method. Using the knowledge that the ML model minimizes the loss on training samples, the attack determines that a target sample is part of training if its loss is lower than a fixed threshold (selected as the average loss of training examples). Sablayrolles et al. [317] refined the loss-based attack by scaling the loss using a per-example threshold. Another popular technique introduced by Shokri et al. [342] is *shadow models*, which trains a meta-classifier on examples in and out of the training set obtained by training thousands of shadow ML models on the same task as the original model. This technique is generally expensive, and while it might improve upon the simple loss-based attack, its computational cost is high and requires access to many samples from the distribution to train the shadow models. These two techniques are at opposite ends of the spectrum in terms of their complexity, but they perform similarly in terms of precision at low false positive rates [54].

An intermediary method that obtains good performance in terms of the AREA UNDER THE CURVE (AUC) metric is the LiRA attack by Carlini et al. [54], which trains a smaller number of shadow models to learn the distribution of model logits on examples in and out of the training set. Using the assumption that the model logit distributions are Gaussian, LiRA performs a hypothesis test for membership inference by estimating the mean and standard deviation of the Gaussian distributions. Ye et al. [421] designed a similar attack that performs a one-sided hypothesis test, which does not make any assumptions on the loss distribution but achieves slightly lower performance than LiRA. Recently, Lopez et al. [225] proposed a more efficient membership inference attack that requires training a single model to predict the quantiles of the confidence score distribution of the model under attack. Membership inference attacks have also been designed under the stricter label-only threat model in which the adversary only has access to the predicted labels of the queried samples [89].

There are several public privacy libraries that offer implementations of membership inference attacks: the TensorFlow Privacy library [350] and the ML Privacy Meter [259].

2.4.3. Property Inference

[**NISTAML.034**] [Back to Index]

In property inference attacks (also called distribution inference), the attacker tries to learn global information about the training data distribution by interacting with an ML model. For example, an attacker can determine the fraction of the training set with a certain sensitive attribute (e.g., demographic information) that might reveal potentially confidential information about the training set that is not intended to be released.

Property inference attacks were introduced by Ateniese et al. [19] and formalized as a distinguishing game between the attacker and the challenger training two models with different fractions of the sensitive data [361]. Property inference attacks were designed in white-box settings in which the attacker has access to the full ML model [19, 134, 361] and black-box settings in which the attacker issues queries to the model and learns either the predicted labels [233] or the class probabilities [74, 437]. These attacks have been demonstrated for HIDDEN MARKOV MODEL, SUPPORT VECTOR MACHINES [19], FEEDFORWARD NEURAL NETWORKS [134, 233, 437], CONVOLUTIONAL NEURAL NETWORKS [361], FEDERATED LEARNING [240], GENERATIVE ADVERSARIAL NETWORKS [443], and GRAPH NEURAL NETWORK [442]. Mahloujifar et al. [233] and Chaudhauri et al. [74] showed that poisoning the property of interest can help design a more effective distinguishing test for property inference. Moreover, Chaudhauri et al. [74] designed an efficient property size estimation attack that recovers the exact fraction of the population of interest.

The relationship between different training set inference attacks, such as membership inference, attribute inference, and property inference, has been explored by Salem et al.[321] under a unified definitional framework.

2.4.4. Model Extraction

[**NISTAML.031**] [Back to Index]

In MLaaS scenarios, cloud providers typically train large ML models using proprietary data and would like to keep the model architecture and parameters confidential. The goal of an attacker performing a MODEL EXTRACTION attack is to extract information about the model architecture and parameters by submitting queries to the ML model trained by an MLaaS provider. The first model stealing attacks were shown by Tramer at al. [376] on several online ML services for different ML models, including logistic regression, decision trees, and neural networks. However, Jagielski et al. [177] have shown the exact extraction of ML models to be impossible. Instead, a functionally equivalent model can be reconstructed that is different than the original model but achieves similar performance at the prediction task. Jagielski et al. [177] have shown that even the weaker task of extracting functionally equivalent models is computationally prohibitive (*NP*-hard).

Several techniques for mounting model extraction attacks have been introduced in the literature. The first method is that of direct extraction based on the mathematical formulation of the operations performed in deep neural networks, which allows the adversary to compute model weights algebraically [58, 177, 376]. A second technique is to use learning methods for extraction. For example, active learning [70] can guide the queries to the ML model for more efficient extraction of model weights, and reinforcement learning can train an adaptive strategy that reduces the number of queries [280]. A third technique uses SIDE CHANNEL information for model extraction. Batina et al. [29] used electromagnetic side channels to recover simple neural network models, while Rakin et al. [303] showed how ROWHAMMER ATTACK can be used for model extraction of more complex convolutional neural network architectures.

Model extraction is often not an end goal but a step toward other attacks. As the model weights and architecture become known, attackers can launch more powerful attacks that are typical for the white-box or gray-box settings. Therefore, preventing model extraction can mitigate downstream attacks that depend on the attacker having knowledge of the model architecture and weights.

2.4.5. Mitigations

The discovery of reconstruction attacks against aggregate information motivated the rigorous definition of *differential privacy* (DP) [112, 113], an extremely strong definition of privacy that guarantees a bound on how much an attacker with access to the algorithm output can learn about each individual record in the dataset. The original *pure* definition of DP has a privacy parameter ε (i.e., privacy budget), which bounds the probability that the attacker with access to the algorithm's output can determine whether a particular record was included in the dataset. DP has been extended to the notions of approximate DP, which includes a second parameter δ that is interpreted as the probability of information accidentally being leaked in addition to ε and Rènyi DP [246].

DP has been widely adopted due to several useful properties: group privacy (i.e., the extension of the definition to two datasets that differ in k records), post-processing (i.e., privacy is preserved even after processing the output), and composition (i.e., privacy is composed if multiple computations are performed on the dataset). DP mechanisms for statistical computations include the Gaussian mechanism [113], the Laplace mechanism [113], and the Exponential mechanism [238]. The most widely used DP algorithm for training ML models is DP-SGD [1], and recent improvements include DP-FTRL [189] and DP matrix factorization [105].

By definition, DP provides mitigation against data reconstruction and membership inference attacks. In fact, the definition of DP immediately implies an upper bound on the success of an adversary in mounting a membership inference attack. Tight bounds on the success of membership inference have been derived by Thudi et al. [369]. However, DP does not provide guarantees against model extraction attacks, as this method is designed to protect the training data, not the model. Several papers have reported negative results after using differential privacy to protect against property inference attacks that aim to extract the properties of subpopulations in the training set [74, 233].

One of the main challenges of using DP in practice is setting up the privacy parameters to achieve a trade-off between the level of privacy and the achieved utility, which is typically measured in terms of accuracy for ML models. Analysis of privacy-preserving algorithms (e.g., DP-SGD) is often worst-case and not tight, and selecting privacy parameters based purely on theoretical analysis results in utility loss. Therefore, large privacy parameters are often used in practice (e.g., the 2020 U.S. Census release used $\varepsilon = 19.61$), and the exact privacy obtained in practice is difficult to estimate. Jagielski et al. [181] introduced *privacy auditing* with the goal of empirically measuring the actual privacy guarantees of an algorithm and determining privacy lower bounds by mounting privacy attacks. Many privacy auditing techniques are based on inserting *canaries* – synthetic and easy-to-identify out-of-distribution examples – into the training set, and then measuring the canary presence in model output. Auditing can also be performed with membership inference attacks [183, 427], but intentional insertion of strong canaries may result in better estimates of privacy leakage [181, 265]. Recent advances in privacy auditing include tighter bounds for the Gaussian mechanism [263] and rigorous statistical methods that allow for the use of multiple canaries to reduce the sample complexity of auditing [297]. Additionally, two efficient methods for privacy auditing with training a single model have been proposed: Steinke et al. [355] use, multiple random data canaries without incurring the cost of group privacy; and Andrew et al. [10] used multiple random client canaries and cosine similarity test statistics to audit user-level private federated learning.

Differential privacy provides a rigorous notion of privacy and protects against membership inference and data reconstruction attacks. To achieve the best balance between privacy and utility, empirical privacy auditing is recommended to complement the theoretical analysis of private training algorithms.

There are other mitigation techniques against model extraction, such as limiting user queries to the model, detecting suspicious queries to the model, or creating more robust architectures to prevent side-channel attacks. However, these techniques can be circumvented by motivated and well-resourced attackers and should be used with caution. There are practice guides available for securing ML deployments [69, 274]. A completely different approach to potentially mitigating privacy leakage of a user's data is to perform MACHINE UNLEARNING, a technique that enables a user to request the removal of their data from a trained ML model. Existing techniques for machine unlearning are either exact (i.e., retraining the model from scratch or from a certain checkpoint) [45, 52] or approximate (i.e., updating the model parameters to remove the influence of the unlearned records) [139, 175, 268]. They offer different tradeoffs between computation and privacy guarantees, with exact unlearning methods offering stronger privacy, at additional computational cost.

3. Generative AI Taxonomy

GenAI is a branch of AI that develops models that can generate content (e.g., images, text, and other media) with similar properties to their training data. GenAI includes several different types of AI technologies with distinct origins, modeling approaches, and related properties, including: GENERATIVE ADVERSARIAL NETWORKS, GENERATIVE PRE-TRAINED TRANSFORMER (GPT), and DIFFUSION MODELS, among others. Recently, GenAI systems have emerged with multi-modal content generation or comprehension capabilities [119], sometimes through combining two or more model types.

3.1. Attack Classification

While many attack types in the PredAI taxonomy apply to GenAI (e.g., data poisoning, model poisoning, and model extraction), recent work has also introduced novel AML attacks specific to GenAI systems.

Figure 2 shows a taxonomy of attacks in AML for GenAI systems. Similar to the PredAI taxonomy in Fig. 1, this taxonomy is first categorized by the system properties that attackers seek to compromise in each case, including **availability** breakdowns, **integrity** violations, and **privacy** compromises, as well as the additional category of AML attack relevant to GenAI of **misuse** enablement, in which attackers seek to circumvent restrictions placed on the outputs of GenAI systems (see Sec. 2.1.2). The capabilities that an adversary must leverage to achieve their objectives are shown in the outer layer of the objective circles. Attack classes are shown as callouts connected to the capabilities required to mount each attack. Where there are specific types of a more general class of attack (for example, a jailbreak is a specific kind of direct prompting attack attack), the specific attack is linked to the more general attack class through an additional callout. Certain attack classes are listed multiple times because the same attack technique can be used to achieve different attacker goals.

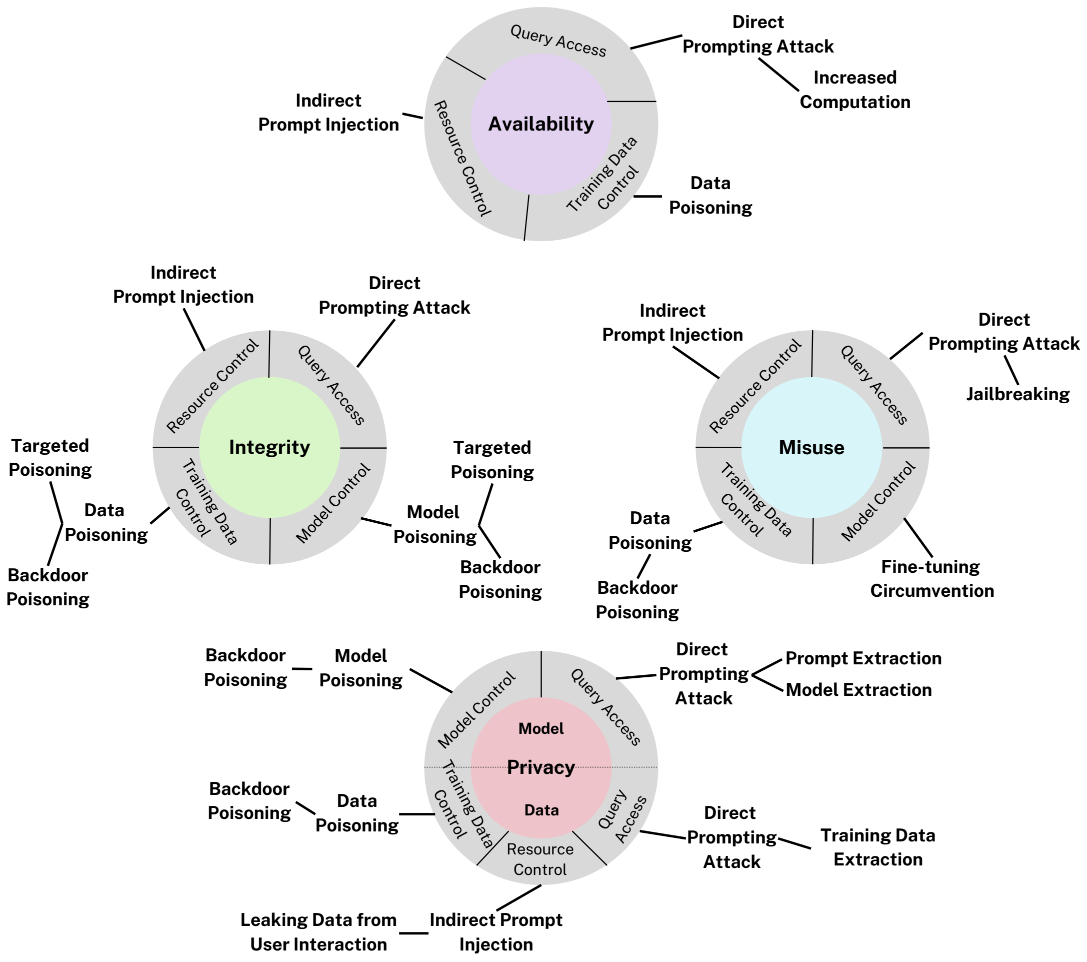

Figure 2. Taxonomy of attacks on GenAI systems

An attack can be further categorized by the learning stage to which it applies and by the attacker's knowledge and access. These are reviewed in the following sections. Where possible, the discussion broadly applies to GenAI models, though some attacks may be most relevant to particular kinds of GenAI models or model-based systems such as RETRIEVAL-AUGMENTED GENERATION (RAG) [RAG] systems, chatbots, or AGENT systems.

3.1.1. GenAI Stages of Learning

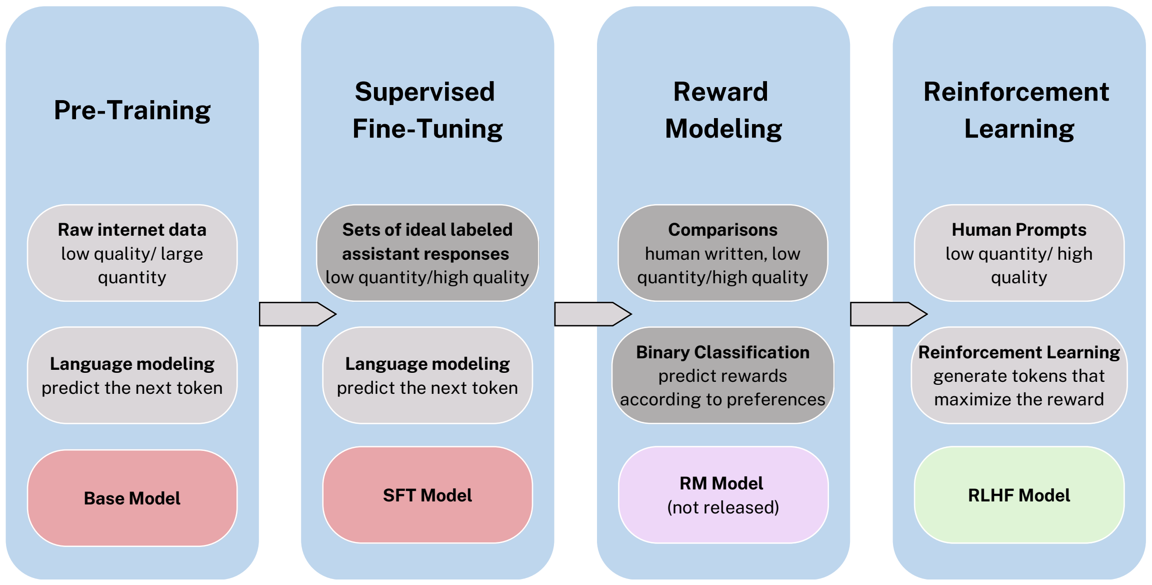

Figure 3. Example LLM Training Pipeline used for InstructGPT [281]

The GenAI development pipeline shapes the space of possible AML attacks against GenAI models and systems. In GenAI moreso than in PredAI, different activities such as data collection, model training, model deployment, and application development are often carried out by multiple different organizations or actors.

For example, a common paradigm in GenAI is the use of a smaller number of foundation models to support a diverse range of downstream applications. Foundation models are pre-trained on large-scale data using self-supervised learning in order to encode general patterns in text, images, or other data that may be relevant for many different applications [311]. Data at the scale used in foundation models is often collected from a variety of internet sources (which attackers can target, such as in DATA POISONING attacks).

This generalist learning paradigm equips foundation models with a variety of capabilities and tendencies — many of which are desirable, but some of which may be harmful or unwanted by the model developer. Techniques such as supervised fine-tuning (SFT) and reinforcement learning from human feedback (RLHF) can be used after initial pre-training to better align the base model with human preferences and to curb undesirable or harmful model outputs [281] (see Fig. 3). However, these interventions can later be targeted using AML techniques by attackers seeking to recover or re-enable potentially harmful capabilities.

Developers can make trained foundation models available to downstream users and developers in a variety of ways, including openly releasing the model's weights for re-use and

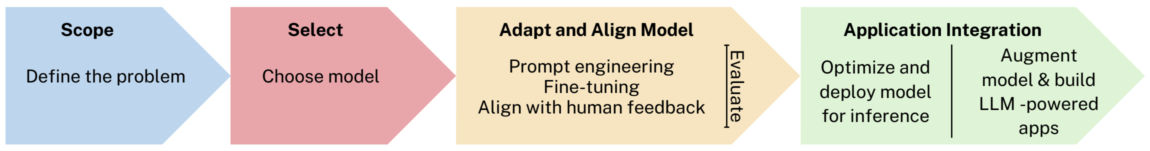

Figure 4. LLM enterprise adoption pipeline

modification, or hosting the model and offering access as a service through an API. These release decisions impact attacker capabilities that shape the space of possible AML attacks, such as whether attackers possess MODEL CONTROL.

Depending on how a foundation model has been made available, downstream developers can customize and build upon the model to create new applications, such as by further fine-tuning the model for a specific use case, or by integrating a foundation model with a software system, such as to build a retrieval-augmented generation (RAG) or agent (see Figure 4). Thus, a foundation model's vulnerabilities to AML attacks can potentially impact a wide range of downstream applications and end users. At the same time, the specific application context in which a foundation model is integrated can create additional vectors for and risks from AML attacks, such as the potential exposure of application-specific data.

AML attacks differ and depend on different phases of the GenAI development lifecycle. One major division is between attacks that target the training stage and those that target model inference during the deployment stage.

Training-time attacks. [**NISTAML.037**] [Back to Index] The TRAINING STAGE for GenAI often consists of foundation model PRE-TRAINING and model FINE-TUNING. This pattern exists for generative image models, text models, audio models, and multimodal models, among others. Since foundation models are most effective when trained on large datasets, it has become common to scrape data from a wide range of public sources, increasing the vulnerability of these models to DATA POISONING attacks. Additionally, GenAI systems trained or fine-tuned by third parties are often used in downstream applications, leading to the risk of MODEL POISONING attacks from maliciously constructed models.