Adversarial Machine Learning
A Taxonomy and Terminology of Attacks and Mitigations

Apostol Vassilev
Alina Oprea
Alie Fordyce
Hyrum Anderson

This publication is available free of charge from:
https://doi.org/10.6028/NIST.AI.100-2e2023
NIST Trustworthy and Responsible AI
NIST AI 100-2e2023

Adversarial Machine Learning
A Taxonomy and Terminology of Attacks and Mitigations

Apostol Vassilev
Computer Security Division
Information Technology Laboratory

Alina Oprea
Northeastern University

Alie Fordyce
Hyrum Anderson
Robust Intelligence, Inc.

This publication is available free of charge from:
https://doi.org/10.6028/NIST.AI.100-2e2023

January 2024

U.S. Department of Commerce
Gina M. Raimondo, Secretary

National Institute of Standards and Technology
Laurie E. Locascio, NIST Director and Under Secretary of Commerce for Standards and Technology
Certain commercial equipment, instruments, software, or materials, commercial or non-commercial, are identified in this paper in order to specify the experimental procedure adequately. Such identification does not imply recommendation or endorsement of any product or service by NIST, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

NIST Technical Series Policies
Copyright, Use, and Licensing Statements
NIST Technical Series Publication Identifier Syntax

Publication History
Approved by the NIST Editorial Review Board on 2024-01-02

How to cite this NIST Technical Series Publication:

NIST Author ORCID iDs
Apostol Vassilev: 0000-0002-4979-5292
Alina Oprea: 0000-0002-9081-3042

Submit Comments
ai-100-2@nist.gov

All comments are subject to release under the Freedom of Information Act (FOIA).
Abstract

This NIST Trustworthy and Responsible AI report develops a taxonomy of concepts and defines terminology in the field of adversarial machine learning (AML). The taxonomy is built on surveying the AML literature and is arranged in a conceptual hierarchy that includes key types of ML methods and lifecycle stages of attack, attacker goals and objectives, and attacker capabilities and knowledge of the learning process. The report also provides corresponding methods for mitigating and managing the consequences of attacks and points out relevant open challenges to take into account in the lifecycle of AI systems. The terminology used in the report is consistent with the literature on AML and is complemented by a glossary that defines key terms associated with the security of AI systems and is intended to assist non-expert readers. Taken together, the taxonomy and terminology are meant to inform other standards and future practice guides for assessing and managing the security of AI systems, by establishing a common language and understanding of the rapidly developing AML landscape.

Keywords

artificial intelligence; machine learning; attack taxonomy; evasion; data poisoning; privacy breach; attack mitigation; data modality; trojan attack, backdoor attack; generative models; large language model; chatbot.

NIST Trustworthy and Responsible AI Reports (NIST Trustworthy and Responsible AI)

The National Institute of Standards and Technology (NIST) promotes U.S. innovation and industrial competitiveness by advancing measurement science, standards, and technology in ways that enhance economic security and improve our quality of life. Among its broad range of activities, NIST contributes to the research, standards, evaluations, and data required to advance the development, use, and assurance of trustworthy artificial intelligence (AI).
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audience</td>
<td>iv</td>
</tr>
<tr>
<td>Background</td>
<td>iv</td>
</tr>
<tr>
<td>Trademark Information</td>
<td>iv</td>
</tr>
<tr>
<td>How to read this document</td>
<td>v</td>
</tr>
<tr>
<td>Executive Summary</td>
<td>1</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>3</td>
</tr>
<tr>
<td>2. Predictive AI Taxonomy</td>
<td>6</td>
</tr>
<tr>
<td>2.1. Attack Classification</td>
<td>6</td>
</tr>
<tr>
<td>2.1.1. Stages of Learning</td>
<td>8</td>
</tr>
<tr>
<td>2.1.2. Attacker Goals and Objectives</td>
<td>9</td>
</tr>
<tr>
<td>2.1.3. Attacker Capabilities</td>
<td>10</td>
</tr>
<tr>
<td>2.1.4. Attacker Knowledge</td>
<td>11</td>
</tr>
<tr>
<td>2.1.5. Data Modality</td>
<td>11</td>
</tr>
<tr>
<td>2.2. Evasion Attacks and Mitigations</td>
<td>14</td>
</tr>
<tr>
<td>2.2.1. White-Box Evasion Attacks</td>
<td>15</td>
</tr>
<tr>
<td>2.2.2. Black-Box Evasion Attacks</td>
<td>17</td>
</tr>
<tr>
<td>2.2.3. Transferability of Attacks</td>
<td>18</td>
</tr>
<tr>
<td>2.2.4. Mitigations</td>
<td>19</td>
</tr>
<tr>
<td>2.3. Poisoning Attacks and Mitigations</td>
<td>21</td>
</tr>
<tr>
<td>2.3.1. Availability Poisoning</td>
<td>21</td>
</tr>
<tr>
<td>2.3.2. Targeted Poisoning</td>
<td>23</td>
</tr>
<tr>
<td>2.3.3. Backdoor Poisoning</td>
<td>24</td>
</tr>
<tr>
<td>2.3.4. Model Poisoning</td>
<td>27</td>
</tr>
<tr>
<td>2.4. Privacy Attacks</td>
<td>29</td>
</tr>
<tr>
<td>2.4.1. Data Reconstruction</td>
<td>29</td>
</tr>
<tr>
<td>2.4.2. Membership Inference</td>
<td>30</td>
</tr>
<tr>
<td>2.4.3. Model Extraction</td>
<td>31</td>
</tr>
<tr>
<td>2.4.4. Property Inference</td>
<td>32</td>
</tr>
<tr>
<td>2.4.5. Mitigations</td>
<td>32</td>
</tr>
<tr>
<td>3. Generative AI Taxonomy</td>
<td>35</td>
</tr>
</tbody>
</table>
3.1. Attack Classification ............................................. 35
  3.1.1. GenAI Stages of Learning .......................... 36
  3.1.2. Attacker Goals and Objectives .................. 38
  3.1.3. Attacker Capabilities .............................. 39
3.2. AI Supply Chain Attacks and Mitigations ....................... 39
  3.2.1. Deserialization Vulnerability .................... 39
  3.2.2. Poisoning Attacks ................................ 40
  3.2.3. Mitigations ......................................... 40
3.3. Direct Prompt Injection Attacks and Mitigations ............... 40
  3.3.1. Data Extraction ................................... 43
  3.3.2. Mitigations ........................................ 44
3.4. Indirect Prompt Injection Attacks and Mitigations ............. 44
  3.4.1. Availability Violations ........................... 45
  3.4.2. Integrity Violations ............................... 46
  3.4.3. Privacy Compromises .............................. 46
  3.4.4. Abuse Violations ................................ 47
  3.4.5. Mitigations ........................................ 48
4. Discussion and Remaining Challenges ................................ 50
  4.1. The Scale Challenge ................................... 50
  4.2. Theoretical Limitations on Adversarial Robustness ........ 50
  4.3. The Open vs. Closed Model Dilemma ................ 53
  4.4. Supply chain challenges ............................... 54
  4.5. Tradeoffs Between the Attributes of Trustworthy AI .... 54
  4.7. Quantized models ..................................... 55
Appendix: Glossary .................................................. 92

List of Figures

Figure 1. Taxonomy of attacks on Predictive AI systems .......... 7
Figure 2. Taxonomy of attacks on Generative AI systems .......... 36
Figure 3. Retrieval-augmented generation relies on system instructions, context, and data from third-party sources, often through a vector database, to produce relevant responses for users .......... 38
Audience

The intended primary audience for this document includes individuals and groups who are responsible for designing, developing, deploying, evaluating, and governing AI systems.

Background

This document is a result of an extensive literature review, conversations with experts from the area of adversarial machine learning, and research performed by the authors in adversarial machine learning.

Trademark Information

All trademarks and registered trademarks belong to their respective organizations.

The Information Technology Laboratory (ITL) at NIST develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL’s responsibilities include the development of management, administrative, technical, and physical standards and guidelines.

This NIST Trustworthy and Responsible AI report focuses on identifying, addressing, and managing risks associated with adversarial machine learning. While practical guidance\(^1\) published by NIST may serve as an informative reference, this guidance remains voluntary.

The content of this document reflects recommended practices. This document is not intended to serve as or supersede existing regulations, laws, or other mandatory guidance.

\(^{1}\)The term ‘practice guide,’ ‘guide,’ ‘guidance’ or the like, in the context of this paper, is a consensus-created, informative reference intended for voluntary use; it should not be interpreted as equal to the use of the term ‘guidance’ in a legal or regulatory context. This document does not establish any legal standard or any other legal requirement or defense under any law, nor have the force or effect of law.
How to read this document

This document uses terms such as AI technology, AI system, and AI applications interchangeably. Terms related to the machine learning pipeline, such as ML model or algorithm, are also used interchangeably in this document. Depending on context, the term “system” may refer to the broader organizational and/or social ecosystem within which the technology was designed, developed, deployed, and used instead of the more traditional use related to computational hardware or software.

Important reading notes:

- The document includes a series of blue callout boxes that highlight interesting nuances and important takeaways.
- Terms that are used but not defined/explained in the text are listed and defined in the Glossary. They are displayed in small caps in the text. Clicking on a word shown in small caps (e.g., ADVERSARIAL EXAMPLES) takes the reader directly to the definition of that term in the Glossary. From there, one may click on the page number shown at the end of the definition to return.

Acknowledgments

The authors wish to thank all people and organizations who responded to our call and submitted comments to the draft version of this paper. The received comments and suggested references were essential to improving the paper and the future direction of this work. We also want to thank the many NIST colleagues who assisted in updating the document.

Author Contributions

Authors contributed equally to this work.


Executive Summary

This NIST Trustworthy and Responsible AI report is intended to be a step toward developing a taxonomy and terminology of adversarial machine learning (AML), which in turn may aid in securing applications of artificial intelligence (AI) against adversarial manipulations of AI systems. Broadly, there are two classes of AI systems: Predictive and Generative. The components of an AI system include – at a minimum – the data, model, and processes for training, testing, and deploying the machine learning (ML) models and the infrastructure required for using them. Generative AI systems may also be linked to corporate documents and databases when they are adapted to specific domains and use cases. The data-driven approach of ML introduces additional security and privacy challenges in different phases of ML operations besides the classical security and privacy threats faced by most operational systems. These security and privacy challenges include the potential for adversarial manipulation of training data, adversarial exploitation of model vulnerabilities to adversely affect the performance of the AI system, and even malicious manipulations, modifications or mere interaction with models to exfiltrate sensitive information about people represented in the data, about the model itself, or proprietary enterprise data. Such attacks have been demonstrated under real-world conditions, and their sophistication and potential impact have been increasing steadily. AML is concerned with studying the capabilities of attackers and their goals, as well as the design of attack methods that exploit the vulnerabilities of ML during the development, training, and deployment phase of the ML lifecycle. AML is also concerned with the design of ML algorithms that can withstand these security and privacy challenges. When attacks are launched with malevolent intent, the robustness of ML refers to mitigations intended to manage the consequences of such attacks.

This report adopts the notions of security, resilience, and robustness of ML systems from the NIST AI Risk Management Framework [226]. Security, resilience, and robustness are gauged by risk, which is a measure of the extent to which an entity (e.g., a system) is threatened by a potential circumstance or event (e.g., an attack) and the severity of the outcome should such an event occur. However, this report does not make recommendations on risk tolerance (the level of risk that is acceptable to organizations or society) because it is highly contextual and application/use-case specific. This general notion of risk offers a useful approach for assessing and managing the security, resilience, and robustness of AI system components. Quantifying these likelihoods is beyond the scope of this document. Correspondingly, the taxonomy of AML is defined with respect to the following five dimensions of AML risk assessment: (i) AI system type (Predictive or Generative), (ii) learning method and stage of the ML lifecycle process when the attack is mounted, (iii) attacker goals and objectives, (iv) attacker capabilities, (v) and attacker knowledge of the learning process and beyond.

The spectrum of effective attacks against ML is wide, rapidly evolving, and covers all phases of the ML lifecycle – from design and implementation to training, testing, and finally, to deployment in the real world. The nature and power of these attacks are different
and can exploit not just vulnerabilities of the ML models but also weaknesses of the infrastructure in which the AI systems are deployed. Although AI system components may also be adversely affected by various unintentional factors, such as design and implementation flaws and data or algorithm biases, these factors are not intentional attacks. Even though these factors might be exploited by an adversary, they are not within the scope of the literature on AML or this report.

This document defines a taxonomy of attacks and introduces terminology in the field of AML. The taxonomy is built on a survey of the AML literature and is arranged in a conceptual hierarchy that includes key types of ML methods and lifecycle stages of attack, attacker goals and objectives, and attacker capabilities and knowledge of the learning process. The report also provides corresponding methods for mitigating and managing the consequences of attacks and points out relevant open challenges to take into account in the lifecycle of AI systems. The terminology used in the report is consistent with the literature on AML and is complemented by a glossary that defines key terms associated with the security of AI systems in order to assist non-expert readers. Taken together, the taxonomy and terminology are meant to inform other standards and future practice guides for assessing and managing the security of AI systems by establishing a common language and understanding for the rapidly developing AML landscape. Like the taxonomy, the terminology and definitions are not intended to be exhaustive but rather to aid in understanding key concepts that have emerged in AML literature.
1. Introduction

Artificial intelligence (AI) systems [220] are on a global multi-year accelerating expansion trajectory. These systems are being developed by and widely deployed into the economies of numerous countries, leading to the emergence of AI-based services for people to use in many spheres of their lives, both real and virtual [77]. There are two broad classes of AI systems, based on their capabilities: Predictive AI (PredAI) and Generative AI (GenAI). As these systems permeate the digital economy and become inextricably essential parts of daily life, the need for their secure, robust, and resilient operation grows. These operational attributes are critical elements of Trustworthy AI in the NIST AI Risk Management Framework [226] and in the taxonomy of AI Trustworthiness [223].

However, despite the significant progress that AI and machine learning (ML) have made in a number of different application domains, these technologies are also vulnerable to attacks that can cause spectacular failures with dire consequences.

For example in PredAI computer vision applications for object detection and classification, well-known cases of adversarial perturbations of input images have caused autonomous vehicles to swerve into the opposite direction lane. The misclassification of stop signs as speed limit signs caused critical objects to disappear from images, and even to misidentify people wearing glasses in high-security settings [99, 150, 260, 277]. Similarly, in the medical field where more and more ML models are being deployed to assist doctors, there is the potential for medical record leaks from ML models that can expose deeply personal information [14, 135].

In GenAI, large language models (LLMs) [6, 38, 70, 83, 196, 209, 228, 276, 293, 294, 345] are also becoming an integral part of the Internet infrastructure and software applications. LLMs are being used to create more powerful online search, help software developers write code, and even power chatbots that help with customer service. LLMs are being integrated with corporate databases and documents to enable powerful RETRIEVAL AUGMENTED GENERATION (RAG) [173] scenarios when LLMs are adapted to specific domains and use cases. These scenarios in effect expose a new attack surface to potentially confidential and proprietary enterprise data.

With the exception of BLOOM [209] and LLaMA[293], most of the companies developing such models do not release detailed information about the data sets that have been used to build their language models, but these data sets inevitably include some sensitive personal information, such as addresses, phone numbers, and email addresses. This creates serious risks for user privacy online. The more often a piece of information appears in a dataset, the more likely a model is to leak it in response to random or specifically designed queries or prompts. This could perpetuate wrong and harmful associations with damaging consequences for the people involved and bring additional security and safety concerns [51, 201].

Attackers can also manipulate the training data for both PredAI and GenAI systems, thus
making the AI system trained on it vulnerable to attacks [256]. Scraping of training data from the Internet also opens up the possibility of DATA POISONING at scale [46] by hackers to create vulnerabilities that allow for security breaches down the pipeline.

As ML models continue to grow in size, many organizations rely on pre-trained models that could either be used directly or be fine-tuned with new datasets to enable different tasks. This creates opportunities for malicious modifications of pre-trained models by inserting TROJANS to enable attackers to compromise the model availability, force incorrect processing, or leak the data when instructed [118].

Historically, modality-specific AI technology has emerged for each input modality (e.g., text, images, speech, tabular data) in PredAI and GenAI systems, each of which is susceptible to domain-specific attacks. For example, the attack approaches for image classification tasks do not directly translate to attacks against natural language processing (NLP) models. Recently, transformer architectures that are used extensively in NLP have shown to have applications in the computer vision domain [90]. In addition, multimodal ML has made exciting progress in many tasks, including generative and classification tasks, and there have been attempts to use multimodal learning as a potential mitigation of single-modality attacks [328]. However, powerful simultaneous attacks against all modalities in a multimodal model have also emerged [63, 261, 326].

Fundamentally, the machine learning methodology used in modern AI systems is susceptible to attacks through the public APIs that expose the model, and against the platforms on which they are deployed. This report focuses on the former and considers the latter to be the scope of traditional cybersecurity taxonomies. For attacks against models, attackers can breach the confidentiality and privacy protections of the data and model by simply exercising the public interfaces of the model and supplying data inputs that are within the acceptable range. In this sense, the challenges facing AML are similar to those facing cryptography. Modern cryptography relies on algorithms that are secure in an information-theoretic sense. Thus, people need to focus only on implementing them robustly and securely—no small task. Unlike cryptography, there are no information-theoretic security proofs for the widely used machine learning algorithms. Moreover, information-theoretic impossibility results have started to appear in the literature [102, 116] that set limits on the effectiveness of widely-used mitigation techniques. As a result, many of the advances in developing mitigations against different classes of attacks tend to be empirical and limited in nature.

This report offers guidance for the development of the following:

- Standardized terminology in AML to be used by the ML and cybersecurity communities;
- A taxonomy of the most widely studied and effective attacks in AML, including
  - evasion, poisoning, and privacy attacks for PredAI systems,
  - evasion, poisoning, privacy, and abuse attacks for GenAI systems;
- attacks against all viable learning methods (e.g., supervised, unsupervised, semi-supervised, federated learning, reinforcement learning) across multiple data modalities.

- A discussion of potential mitigations in AML and limitations of some of the existing mitigation techniques.

As ML is a fast evolving field, we envision the need to update the report regularly as new developments emerge on both the attack and mitigation fronts.

The goal of this report is not to provide an exhaustive survey of all literature on AML. In fact, this by itself is an almost impossible task as a search on arXiv for AML articles in 2021 and 2022 yielded more than 5000 references. Rather, this report provides a categorization of attacks and their mitigations for PredAI and GenAI systems, starting with the main types of attacks: 1) evasion, 2) data and model poisoning, 3) data and model privacy, and 4) abuse (GenAI only).

This report is organized into three sections. In Section 2 we consider PredAI systems. Section 2.1 introduces the taxonomy of attacks for PredAI systems. The taxonomy is organized by first defining the broad categories of attacker objectives/goals. Based on that, we define the categories of capabilities the adversary must be able to leverage to achieve the corresponding objectives. Then, we introduce specific attack classes for each type of capability. Sections 2.2, 2.3, and 2.4 discuss the major classes of attacks: evasion, poisoning, and privacy, respectively. A corresponding set of mitigations for each class of attacks is provided in the attack class sections. In Section 3 we consider GenAI systems. Section 3.1 introduces the taxonomy of attacks for GenAI systems. Similarly to the PredAI case, we define the categories of capabilities the adversary must be able to leverage to achieve the corresponding objectives with GenAI systems. Then, we introduce specific attack classes for each type of capability. Section 4 discusses the remaining challenges in the field.
2. **Predictive AI Taxonomy**

2.1. **Attack Classification**

Figure 1 introduces a taxonomy of attacks in adversarial machine learning for PredAI systems. The attacker’s objectives are shown as disjointed circles with the attacker’s goal at the center of each circle: **Availability** breakdown, **Integrity** violations, and **Privacy** compromise. 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. Multiple attack classes that requiring same capabilities for reaching the same objective are shown in a single callout. Related attack classes that require different capabilities for reaching the same objective are connected with dotted lines.
These attacks are classified according to the following dimensions: 1) learning method and stage of the learning process when the attack is mounted, 2) attacker goals and objectives, 3) attacker capabilities, and 4) attacker knowledge of the learning process. Several adversarial attack classification frameworks have been introduced in prior works [30, 283], and the goal here is to create a standard terminology for adversarial attacks on ML that unifies existing work.
2.1.1. Stages of Learning

Machine learning involves a TRAINING STAGE, in which a model is learned, and a DEPLOYMENT STAGE, in which the model is deployed on new, unlabeled data samples to generate predictions. In the case of SUPERVISED LEARNING labeled training data is given as input to a training algorithm in the training stage and the ML model is optimized to minimize a specific loss function. Validation and testing of the ML model is usually performed before the model is deployed in the real world. Common supervised learning techniques include CLASSIFICATION, in which the predicted labels or classes are discrete, and REGRESSION, in which the predicted labels or response variables are continuous.

ML models may be GENERATIVE (i.e., learn the distribution of training data and generate similar examples, such as generative adversarial networks [GAN] and large language models [LLM]), cf. Section 3, or DISCRIMINATIVE (i.e., learn only a decision boundary, such as LOGISTIC REGRESSION, SUPPORT VECTOR MACHINES, and CONVOLUTIONAL NEURAL NETWORKS). Most PredAI models are DISCRIMINATIVE.

Other learning paradigms in the ML literature are UNSUPERVISED LEARNING, which trains models using unlabeled data at training time; SEMI-SUPERVISED LEARNING, in which a small set of examples have labels, while the majority of samples are unlabeled; REINFORCEMENT LEARNING, in which an agent interacts with an environment and learns an optimal policy to maximize its reward; FEDERATED LEARNING, in which a set of clients jointly train an ML model by communicating with a server, which performs an aggregation of model updates; ENSEMBLE LEARNING which is an approach in machine learning that seeks better predictive performance by combining the predictions from multiple models.

Adversarial machine learning literature predominantly considers adversarial attacks against AI systems that could occur at either the training stage or the ML deployment stage. During the ML training stage, the attacker might control part of the training data, their labels, the model parameters, or the code of ML algorithms, resulting in different types of poisoning attacks. During the ML deployment stage, the ML model is already trained, and the adversary could mount evasion attacks to create integrity violations and change the ML model’s predictions, as well as privacy attacks to infer sensitive information about the training data or the ML model.

Training-time attacks. Attacks during the ML training stage are called POISONING ATTACKS [28]. In a DATA POISONING attack [28, 124], an adversary controls a subset of the training data by either inserting or modifying training samples. In a MODEL POISONING attack [185], the adversary controls the model and its parameters. Data poisoning attacks are applicable to all learning paradigms, while model poisoning attacks are most prevalent in federated learning [152], where clients send local model updates to the aggregating server, and in supply-chain attacks where malicious code may be added to the model by suppliers of model technology.

Deployment-time attacks. Two different types of attacks can be mounted at inference or
deployment time. First, evasion attacks modify testing samples to create ADVERSARIAL EXAMPLES [26, 120, 287], which are similar to the original sample (according to certain distance metrics) but alter the model predictions to the attacker’s choices. Second, privacy attacks, such as membership inference [269] and data reconstruction [89], are typically mounted by attackers with query access to an ML model. They could be further divided into data privacy attacks and model privacy attacks.

2.1.2. Attacker Goals and Objectives

The attacker’s objectives are classified along three dimensions according to the three main types of security violations considered when analyzing the security of a system (i.e., availability, integrity, confidentiality): availability breakdown, integrity violations, and privacy compromise. Correspondingly, ADVERSARIAL SUCCESS indicates achieving one or more of these objectives. Figure 1 separates attacks into three disjointed circles according to their objective, and the attacker’s objective is shown at the center of each circle.

Availability Breakdown. An AVAILABILITY ATTACK is an indiscriminate attack against ML in which the attacker attempts to break down the performance of the model at deployment time. Availability attacks can be mounted via data poisoning, when the attacker controls a fraction of the training set; via model poisoning, when the attacker controls the model parameters; or as ENERGY-LATENCY ATTACKS via query access. Data poisoning availability attacks have been proposed for SUPPORT VECTOR MACHINES [28], linear regression [143], and even neural networks [190, 215], while model poisoning attacks have been designed for neural networks [185] and federated learning [12]. Recently, ENERGY-LATENCY ATTACKS that require only black-box access to the model have been developed for neural networks across many different tasks in computer vision and NLP [273].

Integrity Violations. An INTEGRITY ATTACK targets the integrity of an ML model’s output, resulting in incorrect predictions performed by an ML model. An attacker can cause an integrity violation by mounting an evasion attack at deployment time or a poisoning attack at training time. Evasion attacks require the modification of testing samples to create adversarial examples that are mis-classified by the model to a different class, while remaining stealthy and imperceptible to humans [26, 120, 287]. Integrity attacks via poisoning can be classified as TARGETED POISONING ATTACKS [113, 258], BACKDOOR POISONING ATTACKS [124], and MODEL POISONING [12, 24, 101]. Targeted poisoning tries to violate the integrity of a few targeted samples and assumes that the attacker has training data control to insert the poisoned samples. Backdoor poisoning attacks require the generation of a BACKDOOR PATTERN, which is added to both the poisoned samples and the testing samples to cause misclassification. Backdoor attacks are the only attacks in the literature that require both training and testing data control. Model poisoning attacks could result in either targeted or backdoor attacks, and the attacker modifies model parameters to cause an integrity violation. They have been designed for centralized learning [185] and federated learning [12, 24].
Privacy Compromise. Attackers might be interested in learning information about the training data (resulting in DATA PRIVACY attacks) or about the ML model (resulting in MODEL PRIVACY attacks). The attacker could have different objectives for compromising the privacy of training data, such as DATA RECONSTRUCTION [89] (inferring content or features of training data), MEMBERSHIP-INFERENCE ATTACKS [130, 270] (inferring the presence of data in the training set), data EXTRACTION [48, 51] (ability to extract training data from generative models), and PROPERTY INFERENCE [110] (inferring properties about the training data distribution). MODEL EXTRACTION is a model privacy attack in which attackers aim to extract information about the model [141].

2.1.3. Attacker Capabilities

An adversary might leverage six types of capabilities to achieve their objectives, as shown in the outer layer of the objective circles in Figure 1:

- **TRAINING DATA CONTROL**: The attacker might take control of a subset of the training data by inserting or modifying training samples. This capability is used in data poisoning attacks (e.g., availability poisoning, targeted or backdoor poisoning).

- **MODEL CONTROL**: The attacker might take control of the model parameters by either generating a Trojan trigger and inserting it in the model or by sending malicious local model updates in federated learning.

- **TESTING DATA CONTROL**: The attacker may utilize this to add perturbations to testing samples at model deployment time, as performed in evasion attacks to generate adversarial examples or in backdoor poisoning attacks.

- **LABEL LIMIT**: This capability is relevant to restrict the adversarial control over the labels of training samples in supervised learning. Clean-label poisoning attacks assume that the attacker does not control the label of the poisoned samples – a realistic poisoning scenario, while regular poisoning attacks assume label control over the poisoned samples.

- **SOURCE CODE CONTROL**: The attacker might modify the source code of the ML algorithm, such as the random number generator or any third-party libraries, which are often open source.

- **QUERY ACCESS**: When the ML model is managed by a cloud provider (using Machine Learning as a Service – MLaaS), the attacker might submit queries to the model and receive predictions (either labels or model confidences). This capability is used by black-box evasion attacks, ENERGY-LATENCY ATTACKS, and all privacy attacks.

Note that even if an attacker does not have the ability to modify training/testing data, source code, or model parameters, access to these are still crucial for mounting white-box attacks. See Section 2.1.4 for more details on attacker knowledge.
Figure 1 connects each attack class with the capabilities required to mount the attack. For instance, backdoor attacks that cause integrity violations require control of training data and testing data to insert the backdoor pattern. Backdoor attacks can also be mounted via source code control, particularly when training is outsourced to a more powerful entity. Clean-label backdoor attacks do not allow label control on the poisoned samples, in addition to the capabilities needed for backdoor attacks.

2.1.4. Attacker Knowledge

Another dimension for attack classification is how much knowledge the attacker has about the ML system. There are three main types of attacks: white-box, black-box, and gray-box.

**White-box attacks.** These assume that the attacker operates with full knowledge about the ML system, including the training data, model architecture, and model hyper-parameters. While these attacks operate under very strong assumptions, the main reason for analyzing them is to test the vulnerability of a system against worst-case adversaries and to evaluate potential mitigations. Note that this definition is more general and encompasses the notion of adaptive attacks where the knowledge of the mitigations applied to the model or the system is explicitly tracked.

**Black-box attacks.** These attacks assume minimal knowledge about the ML system. An adversary might get query access to the model, but they have no other information about how the model is trained. These attacks are the most practical since they assume that the attacker has no knowledge of the AI system and utilize system interfaces readily available for normal use.

**Gray-box attacks.** There are a range of gray-box attacks that capture adversarial knowledge between black-box and white-box attacks. Suciu et al. [283] introduced a framework to classify gray-box attacks. An attacker might know the model architecture but not its parameters, or the attacker might know the model and its parameters but not the training data. Other common assumptions for gray-box attacks are that the attacker has access to data distributed identically to the training data and knows the feature representation. The latter assumption is important in applications where feature extraction is used before training an ML model, such as cybersecurity, finance, and healthcare.

2.1.5. Data Modality

Adversarial attacks against ML have been discovered in a range of data modalities used in many application domains. Until recently, most attacks and defenses have operated under a single modality, but a new ML trend is to use multimodal data. The taxonomy of attacks defined in Figure 1 is independent of the modality of the data in specific applications.

The most common data modalities in the adversarial ML literature include:

1. **Image:** Adversarial examples of image data modality [120, 287] have the advantage
of a continuous domain, and gradient-based methods can be applied directly for optimization. Backdoor poisoning attacks were first invented for images [124], and many privacy attacks are run on image datasets (e.g., [269]). The image modality includes other types of imaging (e.g., LIDAR, SAR, IR, ‘hyperspectral’).

2. **Text:** Natural language processing (NLP) is a popular modality, and all classes of attacks have been proposed for NLP applications, including evasion [126], poisoning [68, 175], and privacy [337].

3. **Audio:** Audio systems and text generated from audio signals have also been attacked [54].

4. **Video:** Video comprehension models have shown increasing capabilities on vision-and-language tasks [339], but such models are also vulnerable to attacks [318].

5. **Cybersecurity²:** The first poisoning attacks were discovered in cybersecurity for worm signature generation (2006) [236] and spam email classification (2008) [222]. Since then, poisoning attacks have been shown for malware classification, malicious PDF detection, and Android malicious app classification [257]. Evasion attacks against the same data modalities have been proposed as well: malware classification [84, 282], PDF malware classification [279, 325], and Android malicious app detection [239]. Clements et al. [78] developed a mechanism for effective generation of evasion attacks on small, weak routers in network intrusion detection. Poisoning unsupervised learning models has been shown for clustering used in malware classification [29] and network traffic anomaly detection [249].

Industrial Control Systems (ICS) and Supervisory Control and Data Acquisition (SCADA) systems are part of modern Critical Infrastructure (CI) such as power grids, power plants (nuclear, fossil fuel, renewable energy), water treatment plants, oil refineries, etc. ICS are an attractive target for adversaries because of the potential for highly consequential disruptions of CI [55, 167]. The existence of targeted stealth attacks has led to the development of defense-in-depth mechanisms for their detection and mitigation. Anomaly detection based on data-centric approaches allows automated feature learning through ML algorithms. However, the application of ML to such problems comes with specific challenges related to the need for a very low false negative and low false positive rates, ability to catch zero-day attacks, account for plant operational drift, etc. This challenge is compounded by the fact that trying to accommodate all these together makes ML models susceptible to adversarial attacks [161, 243, 353].

6. **Tabular data:** Numerous attacks against ML models working on tabular data in finance, business, and healthcare applications have been demonstrated. For example, poisoning availability attacks have been shown against healthcare and business ap-

---

²Strictly speaking, cybersecurity data may not include a single modality, but rather multiple modalities such as network-level, host-level, or program-level data.
Applications [143]; privacy attacks have been shown against healthcare data [333]; and evasion attacks have been shown against financial applications [117].

Recently, the use of ML models trained on multimodal data has gained traction, particularly the combination of image and text data modalities. Several papers have shown that multimodal models may provide some resilience against attacks [328], but other papers show that multimodal models themselves could be vulnerable to attacks mounted on all modalities at the same time [63, 261, 326]. See Section 4.6 for additional discussion.

An interesting open challenge is to test and characterize the resilience of a variety of multimodal ML against evasion, poisoning, and privacy attacks.
2.2. Evasion Attacks and Mitigations

The discovery of evasion attacks against machine learning models has generated increased interest in adversarial machine learning, leading to significant growth in this research space over the last decade. In an evasion attack, the adversary’s goal is to generate adversarial examples, which are defined as testing samples whose classification can be changed at deployment time to an arbitrary class of the attacker’s choice with only minimal perturbation [287]. Early known instances of evasion attacks date back to 1988 with the work of Kearns and Li [155], and to 2004, when Dalvi et al. [82], and Lowd and Meek [188] demonstrated the existence of adversarial examples for linear classifiers used in spam filters. Adversarial examples became even more intriguing to the research community when Szegedy et al. [287] showed that deep neural networks used for image classification can be easily manipulated, and adversarial examples were visualized. In the context of image classification, the perturbation of the original sample must be small so that a human cannot observe the transformation of the input. Therefore, while the ML model can be tricked to classify the adversarial example in the target class selected by the attacker, humans still recognize it as part of the original class.

In 2013, Szegedy et al. [287] and Biggio et al. [26] independently discovered an effective method for generating adversarial examples against linear models and neural networks by applying gradient optimization to an adversarial objective function. Both of these techniques require white-box access to the model and were improved by subsequent methods that generated adversarial examples with even smaller perturbations [10, 53, 194]. Adversarial examples are also applicable in more realistic black-box settings in which attackers only obtain query access capabilities to the trained model. Even in the more challenging black-box setting in which attackers obtain the model’s predicted labels or confidence scores, deep neural networks are still vulnerable to adversarial examples. Methods for creating adversarial examples in black-box settings include zeroth-order optimization [66], discrete optimization [210], and Bayesian optimization [271], as well as transferability, which involves the white-box generation of adversarial examples on a different model architecture before transferring them to the target model [232, 233, 299]. Cybersecurity and image classifications were the first application domains that showcased evasion attacks. However, with the increasing interest in adversarial machine learning, ML technology used in many other application domains went under scrutiny, including speech recognition [54], natural language processing [149], and video classification [177, 317].

Mitigating adversarial examples is a well-known challenge in the community and deserves additional research and investigation. The field has a history of publishing defenses evaluated under relatively weak adversarial models that are subsequently broken by more powerful attacks, a process that appears to iterate in perpetuity. Mitigations need to be evaluated against strong adaptive attacks, and guidelines for the rigorous evaluation of newly proposed mitigation techniques have been established [81, 297]. The most promising directions for mitigating the critical threat of evasion attacks are adversarial training [120, 194] (iteratively generating and inserting adversarial examples with their correct labels at train-
ing time); certified techniques, such as randomized smoothing [79] (evaluating ML prediction under noise); and formal verification techniques [112, 154] (applying formal method techniques to verify the model’s output). Nevertheless, these methods come with different limitations, such as decreased accuracy for adversarial training and randomized smoothing, and computational complexity for formal methods. There is an inherent trade-off between robustness and accuracy [296, 301, 342]. Similarly, there are trade-offs between a model’s robustness and fairness guarantees [59].

This section discusses white-box and black-box evasion attack techniques, attack transferability, and the potential mitigation of adversarial examples in more detail.

### 2.2.1. White-Box Evasion Attacks

There are several optimization-based methods for designing evasion attacks that generate adversarial examples at small distances from the original testing samples. There are also several choices for distance metrics, universal evasion attacks, and physically realizable attacks, as well as examples of evasion attacks developed for multiple data modalities, including NLP, audio, video, and cybersecurity domains.

**Optimization-based methods.** Szedegy et al. [287] and Biggio et al. [26] independently proposed the use of optimization techniques to generate adversarial examples. In their threat models, the adversary is allowed to inspect the entirety of the ML model and compute gradients relative to the model’s loss function. These attacks can be targeted, in which the adversarial example’s class is selected by the attacker, or untargeted, in which the adversarial examples are misclassified to any other incorrect class.

Szedegy et al. [287] coined the widely used term *adversarial examples*. They considered an objective that minimized the $\ell_2$ norm of the perturbation, subject to the model prediction changing to the target class. The optimization is solved using the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) method. Biggio et al. [26] considered the setting of a binary classifier with malicious and benign classes with continuous and differentiable discriminant function. The objective of the optimization is to minimize the discriminant function in order to generate adversarial examples of maximum confidence.

While Biggio et al. [26] apply their method to linear classifiers, kernel SVM, and multi-layer perceptrons, Szedegy et al. [287] show the existence of adversarial examples on deep learning models used for image classification. Goodfellow et al. [120] introduced an efficient method for generating adversarial examples for deep learning: the Fast Gradient Sign Method (FGSM), which performs a single iteration of gradient descent for solving the optimization. This method has been extended to an iterative FGSM attack by Kurakin et al. [163].

Subsequent work on generating adversarial examples have proposed new objectives and methods for optimizing the generation of adversarial examples with the goals of minimizing the perturbations and supporting multiple distance metrics. Some notable attacks include:
1. DeepFool is an untargeted evasion attack for $\ell_2$ norms, which uses a linear approximation of the neural network to construct the adversarial examples [212].

2. 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 [53] and considers three distance metrics to measure the perturbations of adversarial examples: $\ell_0$, $\ell_2$, and $\ell_\infty$. The attack has been effective against the defensive distillation defense [234].

3. The Projected Gradient Descent (PGD) attack [194] 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 $\ell_2$ and $\ell_\infty$ distance metrics for measuring the perturbation of adversarial examples.

**Universal evasion attacks.** Moosavi-Dezfooli et al. [211] showed how to construct small universal perturbations (with respect to some norm), which 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 ATTACKS. 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 machine learning systems that become feasible in the physical world [11, 163, 189]. One of the first physically realizable attacks in the literature is the attack on facial recognition systems by Sharif et al. [260]. 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. [100] 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 [67] attack is 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. In addition, this attack requires the perturbation to be robust enough to survive real-world distortions due to different viewing distances and angles, lighting conditions, and camera limitations.

**Other data modalities.** In computer vision applications, adversarial examples must 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 to humans but causes the model to misclassify. The concept of adversarial examples has been extended to other domains, such as audio, video, natural language processing (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 are discussed below:

- **Audio:** Carlini and Wagner [54] 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 [317] and dense attacks that perturb all of the frames in a video [177]. The goal of the attacker is to change the classification label of the video.

- **NLP:** Jia and Liang [149] developed a methodology for generating adversarial NLP examples. This pioneering work was followed by many advances in developing adversarial attacks on NLP models (see a comprehensive survey on the topic [347]). Recently, La Malfa and Kwiatkowska [164] 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.

- **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 [73]. FENCE has been applied to two network security applications: malicious domain detection and malicious network traffic classification. Sheatsley et al. [262] propose 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 apply the technique to network intrusion detection and phishing classifiers. Both papers observe that attacks from continuous domains cannot be readily applied in constrained environments, as they result in infeasible adversarial examples. Pierazzi et al. [239] discuss the difficulty of mounting feasible evasion attacks in cyber security due to constraints in feature space and the challenge of mapping attacks from feature space to problem space. They formalize evasion attacks in problem space and construct feasible adversarial examples for Android malware.

### 2.2.2. Black-Box Evasion Attacks

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 machine learning as a ser-
vice (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 [66, 137]. Other optimization techniques include discrete optimization [210], natural evolution strategies [136], and random walks [216].

- **Decision-based attacks**: In this more restrictive setting, attackers obtain only 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 [35], which was extended with an improved gradient estimation to reduce the number of queries in the HopSkipJumpAttack [65]. More recently, several optimization methods search for the direction of the nearest decision boundary (the OPT attack [71]), use sign SGD instead of binary searches (the Sign-OPT attack [72]), or use Bayesian optimization [271].

The main 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 [271].

### 2.2.3. Transferability of Attacks

Another method for generating adversarial attacks under restrictive threat models is via transferability of 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. [232, 233] 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 [181, 299, 315].

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 [120, 211, 299]. Demontis et al. [85] 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 im-
Mitigating evasion attacks is challenging because adversarial examples are widespread in a variety of ML model architectures and application domains, as discussed above. 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 [138]. 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 [52]. Recent work shows that detecting adversarial examples is as difficult as building a defense [295]. Therefore, this direction for mitigating adversarial examples is similarly challenging when designing defenses.

- The Obfuscated Gradients attack [10] was specifically designed to defeat several proposed defenses that mask the gradients using the $\ell_0$ and $\ell_\infty$ distance metrics. It relies on a new technique, Backward Pass Differentiable Approximation, which approximates the gradient during the backward pass of backpropagation. It bypasses seven proposed defenses.

- Tramèr et al. [297] described a methodology for designing adaptive attacks against proposed defenses and circumvented 13 existing defenses. They advocate 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 proved resilient and have the potential to provide mitigation against evasion attacks:

1. **Adversarial training**: Introduced by Goodfellow et al. [120] and further developed by Madry et al. [194], adversarial training is a general method that augments the 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. Interestingly, adversarial training results in models with more semantic meaning than standard models [301], 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. [169] and further improved by Cohen et al. [79], 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 $\ell_2$ 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 depends on the radius of the $\ell_2$ ball and the characteristics of the training data and model). Recent results have extended the notion of certified adversarial robustness to $\ell_2$-norm bounded perturbations by combining a pretrained denoising diffusion probabilistic model and a standard high-accuracy classifier [50].

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 [154]. AI$^2$ is the first verification method applicable to convolutional neural networks using abstract interpretation techniques [112]. These methods have been extended and scaled up to larger networks in follow-up verification systems, such as DeepPoly [274], ReluVal [313], and Fast Geometric Projections (FGP) [108]. Formal verification techniques have significant potential for certifying neural network robustness, but their main limitations are their lack of scalability, computational cost, and restriction in the type of supported operations.

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.
2.3. Poisoning Attacks and Mitigations

Another relevant threat against machine learning systems is the risk of adversaries mounting poisoning attacks, which are broadly defined as adversarial attacks during the training stage of the ML algorithm. Poisoning attacks have a long history in cybersecurity, as the first known poisoning attack was developed for worm signature generation in 2006 [236]. Since then, poisoning attacks have been studied extensively in several application domains: computer security (for spam detection [222]), network intrusion detection [305], vulnerability prediction [251], malware classification [257, 323], computer vision [113, 124, 258], natural language processing [68, 175, 309], and tabular data in healthcare and financial domains [143]. Recently, poisoning attacks have gained more attention in industrial applications as well. A Microsoft report revealed that they are considered to be the most critical vulnerability of machine learning systems deployed in production [162]. Recently, it has been shown how poisoning could be orchestrated at scale so that an adversary with limited financial resources can control a fraction of public datasets used for model training [46].

Poisoning attacks are very powerful and can cause either an availability violation or an integrity violation. In particular, availability poisoning attacks cause indiscriminate degradation of the machine learning model on all samples, while targeted and backdoor poisoning attacks are stealthier and induce integrity violations on a small set of target samples. Poisoning attacks leverage a wide range of adversarial capabilities, such as 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 adversarial scenarios [28, 143, 323], gray-box settings [143], and black-box models [27]. This section discusses the threat of 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 as well as existing mitigations and their limitations are also discussed. Our classification of poisoning attacks is inspired by the framework developed by Cinà et al. [76], which includes additional references to poisoning attacks and mitigations.

2.3.1. Availability Poisoning

The first poisoning attacks discovered in cybersecurity applications were availability attacks against worm signature generation and spam classifiers, which indiscriminately impact the entire machine learning model and, in essence, cause a denial-of-service attack on users of the AI system. Perdisci et al. [236] generated suspicious flows with fake invariants that mislead the worm signature generation algorithm in Polygraph [224]. Nelson et al. [222] designed poisoning attacks against Bayes-based spam classifiers, which generate spam emails that contain long sequences of words appearing in legitimate emails to induce the misclassification of spam emails. Both of these attacks were conducted under the white-box setting in which adversaries are aware of the ML training algorithm, feature representations, training datasets, and ML models. ML-based methods have been proposed
for the detection of cybersecurity attacks targeting ICS. Such detectors are often retrained using data collected during system operation to account for plant operational drift of the monitored signals. This retraining procedure creates opportunities for an attacker to mimic the signals of corrupted sensors at training time and poison the learning process of the detector such that attacks remain undetected at deployment time [161].

A simple black-box poisoning attack strategy is LABEL FLIPPING, which generates training examples with a victim label selected by the adversary [27]. This method requires a large percentage of poisoning samples for mounting an availability attack, and it has been improved via optimization-based poisoning attacks introduced for the first time against SUPPORT VECTOR MACHINES (SVM) [28]. In this approach, the attacker solves a bilevel optimization problem to determine the optimal poisoning samples that will achieve the adversarial objective (i.e., maximize the hinge loss for SVM [28] or maximize the mean square error [MSE] for regression [143]). These optimization-based poisoning attacks have been subsequently designed against linear regression [143] and neural networks [215], and they 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 [85, 283].

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 case models 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 [105]. A different approach for clean-label poisoning is to use gradient alignment and minimally modify the training data [106].

Availability poisoning attacks have also been designed for unsupervised learning against centroid-based anomaly detection [159] and behavioral clustering for malware [29]. In federated learning, an adversary can mount a model poisoning attack to induce availability violations in the globally trained model [101, 263, 264]. More details on model poisoning attacks are provided in Section 2.3.4.

Mitigations. Availability poisoning attacks are usually detectable by monitoring the standard performance metrics of ML models – such as precision, recall, accuracy, F1 scores, and area under the curve – as they cause a large degradation in the classifier metrics. Nevertheless, detecting these attacks during the testing or deployment stages of ML is less desirable, and existing mitigations aim to proactively prevent these attacks during the training stage to generate robust ML models. Among the existing mitigations, some generally promising techniques include:

- **Training data sanitization:** These methods leverage the insight that poisoned sam-
ples are typically different than regular training samples not controlled by adversaries. As such, data sanitization techniques are designed to clean the training set and remove the poisoned samples before the machine learning training is performed. Nelson et al. [222] propose 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 this early approach by reducing its computational complexity. Paudice et al. [235] introduced a method for label cleaning that was specifically designed for label flipping attacks. Steinhardt et al. [280] propose the use of outlier detection methods for identifying poisoned samples. Clustering methods have also been used for detecting poisoned samples [165, 288]. In the context of network intrusion detection, computing the variance of predictions made by an ensemble of multiple ML models has proven to be an effective data sanitization method [305]. Once sanitized, the datasets should be protected by cybersecurity mechanisms for provenance and integrity attestation [220].

- **Robust training**: An alternative approach to mitigating availability poisoning attacks is to modify the ML training algorithm and perform robust training instead of regular training. The defender can train an ensemble of multiple models and generate predictions via model voting [25, 172, 314]. Several papers apply techniques from robust optimization, such as using a trimmed loss function [88, 143]. Rosenfeld et al. [248] proposed the use of randomized smoothing for adding noise during training and obtaining certification against label flipping attacks.

2.3.2. **Targeted Poisoning**

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 the clean-label setting in which the attacker does not have access to the labeling function.

Several techniques for mounting clean-label targeted attacks have been proposed. Koh and Liang [160] showed how influence functions – a statistical method that determines the most influential training samples for a prediction – can be leveraged for creating poisoned samples in the fine-tuning setting in which a pre-trained model is fine-tuned on new data. Suciu et al. [283] 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. [258] to generate clean-label targeted poisoning for fine-tuning and end-to-end learning. ConvexPolytope [352] and BullseyePolytope [2] optimized the poisoning samples against ensemble models, which offers better advantages for attack transferability. MetaPoison [133] uses a meta-learning
algorithm to optimize the poisoned samples, while Witches’ Brew [113] 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 [144] 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 [42], such as MixMatch [22], FixMatch [275], and Unsupervised Data Augmentation (UDA) [324] 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. [144] showed an impossibility result for subpopulation poisoning attacks. To mitigate some of the risks associated with such attacks, cybersecurity mechanisms for dataset provenance and integrity attestation [220] should be used judiciously. Ma et al. [192] proposed the use of differential privacy (DP) as a defense (which follows directly from the definition of differential privacy), but it is well known that differentially private ML models have lower accuracy than standard models. The trade-off between robustness and accuracy needs to be considered in each application. If the application has strong data privacy requirements, and differentially private training is used for privacy, then an additional benefit is protection against targeted poisoning attacks. However, the robustness offered by DP starts to fade once the targeted attack requires multiple poisoning samples (as in subpopulation poisoning attacks) because the group privacy bound will not provide meaningful guarantees for large poisoned sets.

2.3.3. Backdoor Poisoning

In 2017, Gu et al. [124] 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 – including the trigger or backdoor pattern – will be misclassified to the target class at testing time. Concurrently, Chen et al. [69] introduced backdoor attacks in which the trigger is blended into the training data. Follow-up work introduced the concept of clean-label backdoor attacks [302] in which the adversary is restricted in preserving the label of the poisoned examples. Clean-label attacks typically require more poisoning samples to be effective, but the attack model is
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 [331]. Backdoor Generating Network (BaN) [253] 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, a.k.a. FUNCTIONAL ATTACKS, are embedded throughout the image or change according to the input. For instance, Li et al. [176] used steganography algorithms to hide the trigger in the training data. Liu et al. [186] introduced a clean-label attack that uses natural reflection on images as a backdoor trigger. Wenger et al. [320] poisoned facial recognition systems by using physical objects as triggers, such as sunglasses and earrings.

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. [268] 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 natural language processing, 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. [68] introduced semantic-preserving backdoors at the character, word, and sentence level for sentiment analysis and neural machine translation applications. Li et al. [175] generated hidden backdoors against transformer models using generative language models in three NLP tasks: toxic comment detection, neural machine translation, and question answering.

• Cybersecurity: Early poisoning attacks in cybersecurity were designed against worm signature generation in 2006 [236] and spam detectors in 2008 [222], well before rising interest in adversarial machine learning. More recently, Severi et al. [257] 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 [329] 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, model inspection and sanitization, and also their limitations.

- **Training Data Sanitization:** Similar to poisoning availability attacks, training data sanitization can be applied to detecting backdoor poisoning attacks. For instance, outlier detection in the latent feature space [129, 238, 300] has been effective for convolutional neural networks used for computer vision applications. Activation Clustering [62] performs clustering of training data in representation space with the goal of isolating 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 that effective against stealthy poisoning attacks. Overall, this leads to a trade-off between attack success and 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. [310] 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 [131, 322]. A representative system in this class is Artificial Brain Simulation (ABS) by Liu et al. [184], which stimulates multiple neurons and measures the activations to reconstruct the trigger patterns. Khaddaj et al. [156] 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 [134], which is based on explainability methods to determine different features between clean and backdoored models that are subsequently used for outlier detection. DeepInspect [64] uses a conditional generative model to learn the probability distribution of trigger patterns and performs model patching to remove the trigger. Xu et al. [327] 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 [321], retraining [340], or fine-tuning [180] to restore the model’s accuracy.

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. [257] showed that some of the data sanitization techniques (e.g., spectral signatures [300] and Activation Clustering [62]) are ineffective against clean-label backdoor poisoning on malware classifiers. Most 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 pre-
dicting a Trojaned model [327] is the high computational complexity of the training stage of the meta classifier, which requires training thousands of SHADOW MODELS. 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. [249] proposed a principal component analysis (PCA)-based approach to mitigate poisoning attacks against PCA subspace anomaly detection method 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 [193] built an autoencoder-based intrusion detection system, assuming malicious poisoning attack instances were under 2%.

A recent paper [156] provides 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 proposes an optimization technique to identify and remove the training samples corresponding to the backdoor.

To complement existing mitigations that are not always resilient in face of evolving attacks, poison forensics [259] is a technique for root cause analysis that identifies the malicious training samples. 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 the source of attack in the training set.

2.3.4. Model Poisoning

Model poisoning attacks attempt to directly modify the trained ML model to inject malicious functionality into the model. In centralized learning, TrojNN [185] 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 [101, 263].

- 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 [13, 23, 285].

- Backdoor model poisoning attacks introduce a trigger via malicious client updates
to induce the misclassification of all samples with the trigger at testing time [13, 23, 285, 312]. 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 [349].

Model poisoning attacks are also possible in supply-chain scenarios where models or components of the model provided by suppliers are poisoned with malicious code. A recent supply-chain attack, Dropout Attack [336], shows how an adversary who manipulates the randomness used in neural network training (in particular in dropout regularization), might poison the model to decrease accuracy, precision, or recall on a set of targeted classes.

**Mitigations.** To defend federated learning from model poisoning attacks, a variety of Byzantine-resilient aggregation rules have been designed and evaluated. Most of them attempt to identify and exclude the malicious updates when performing the aggregation at the server [3, 31, 40, 125, 203–205, 284, 334]. However, motivated adversaries can bypass these defenses by adding constraints in the attack generation optimization problem [13, 101, 263]. Gradient clipping and differential privacy have the potential to mitigate model poisoning attacks to some extent [13, 225, 285], 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, as discussed in Section 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 (such as cryptographic protocol verification [241]) 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 robust in face of supply-chain vulnerabilities is a critical open problem that needs to be addressed by the community.
2.4. Privacy Attacks

Although privacy issues have long been a concern, privacy attacks against aggregate information collected from user records started with the seminal work of Dinur and Nissim [89] on data reconstruction attacks. The goal of reconstruction attacks is to reverse engineer private information about an individual user record or sensitive critical infrastructure data from access to aggregate information. More recently, data reconstruction attacks have been designed for binary and multi-class neural network classifiers [39, 128]. Another privacy attack is that of membership-inference attacks in which an adversary can determine whether a particular record was included in the dataset used for computing statistical information or training a machine learning model. Membership inference attacks were first introduced by Homer et al. [130] 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 [43, 269, 333]. 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 [47, 58, 141, 298].

Property inference attacks [9, 61, 110, 195, 286, 346] aim to extract global information about a training dataset, such as the fraction of training examples with a certain sensitive attribute.

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

2.4.1. Data Reconstruction

Data reconstruction attacks are the most concerning privacy attacks as they have the ability to recover an individual’s data from released aggregate information. Dinur and Nissim [89] were the first to introduce reconstruction attacks that recover user data from linear statistics. Their original attack requires an exponential number of queries for reconstruction, but subsequent work has shown how to perform reconstruction with a polynomial number of queries [96]. A survey of privacy attacks, including reconstruction attacks, is given by Dwork et al. [94]. More recently, the U.S. Census Bureau performed a large-scale study on the risk of data reconstruction attacks on census data [111], 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. [107] introduced model inversion attacks that reconstruct class representatives from the training data of an ML model. While model inversion generates semantically similar images with those in the training set, it cannot directly reconstruct the training data of the model. Recently, Balle et al. [15] trained a reconstructor network that can recover a data sample from a neural network model, assuming a powerful adversary with information about all other training samples. Haim et al. [128] 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 been recently extended to reconstruct training samples of multi-class multi-layer perceptron classifiers [39]. In another relevant privacy attack, attribute inference, the attacker extracts a sensitive attribute of the training set, assuming partial knowledge about other features in the training data [147].

The ability to reconstruct training samples is partially explained by the tendency of neural networks to memorize their training data. Zhang et al. [341] discussed how neural networks can memorize randomly selected datasets. Feldman [103] showed that the memorization of training labels is necessary to achieving almost optimal generalization error in ML. Brown et al. [36] 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 [104]. We will discuss data reconstruction attacks and their connection to memorization for generative AI in Section 3.3.1.

2.4.2. Membership Inference

Membership inference attacks also expose private information about an individual, like reconstruction or memorization attacks, and are still 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 [48, 51].

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. [130] 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 [95]. 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 used for classification [43, 74, 171, 269, 332, 333]. Similar to other attacks in adversarial machine learning, membership inference can be performed in white-box settings [171, 218, 250] 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 [43, 74, 269, 332, 333].

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 [146, 252, 333]. In terms of techniques for mounting membership inference attacks, the loss-based attack by Yeom et al. [333] 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. [250] refined the loss-based attack by scaling the loss using a per-example threshold. Another popular technique introduced by Shokri et al. [269] is that of shadow models, which trains a meta-classifier on examples in and out of the training set obtained from 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 [43].

An intermediary method that is obtains good performance in terms of the AREA UNDER THE CURVE (AUC) metric is the LiRA attack by Carlini et al. [43], 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. [332] 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. [187] propose 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 [74].

There are several public privacy libraries that offer implementations of membership inference attacks: the TensorFlow Privacy library [278] and the ML Privacy Meter [214].

2.4.3. Model Extraction

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. [298] on several online ML services for different ML models, including logistic regression, decision trees, and neural networks. However, Jagielski et al. [141] 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. [141] have shown that even the weaker task of extracting functionally equivalent models is 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 com-
pute model weights algebraically [47, 141, 298]. A second technique explored in a series of papers is to use learning methods for extraction. For instance, active learning [58] 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 [231]. A third technique is the use of side channel information for model extraction. Batina et al. [18] used electromagnetic side channels to recover simple neural network models, while Rakin et al. [245] showed how rowhammer attacks can be used for model extraction of more complex convolutional neural network architectures.

Note that model extraction is often not an end goal but a step towards other attacks. As the model weights and architecture become known, attackers can launch more powerful attacks 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.

2.4.4. Property Inference

In property inference attacks, the attacker tries to learn global information about the training data distribution by interacting with an ML model. For instance, an attacker can determine the fraction of the training set with a certain sensitive attribute, such as 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. [9] and formalized as a distinguishing game between the attacker and the challenger training two models with different fractions of the sensitive data [286]. Property inference attacks were designed in white-box settings in which the attacker has access to the full ML model [9, 110, 286] and black-box settings in which the attacker issues queries to the model and learns either the predicted labels [195] or the class probabilities [61, 346]. These attacks have been demonstrated for hidden Markov models, support vector machines [9], feed-forward neural networks [110, 195, 346], convolutional neural networks [286], federated learning models [200], generative adversarial networks [351], and graph neural networks [350]. Mahloujifar et al. [195] and Chaudhauri et al. [61] showed that poisoning the property of interest can help design a more effective distinguishing test for property inference. Moreover, Chaudhauri et al. [61] designed an efficient property size estimation attack that recovers the exact fraction of the population of interest.

2.4.5. Mitigations

The discovery of reconstruction attacks against aggregate information motivated the rigorous definition of differential privacy (DP) [92, 93]. Differential privacy is 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 $\varepsilon$ (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 $\delta$ that is interpreted as the probability of information accidentally being leaked in addition to $\varepsilon$ and Rényi DP [208].

DP has been widely adopted due to several useful properties: group privacy (i.e., the extension of the definition to two datasets differing 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 that are performed on the dataset). DP mechanisms for statistical computations include the Gaussian mechanism [93], the Laplace mechanism [93], and the Exponential mechanism [198]. The most widely used DP algorithm for training ML models is DP-SGD [1], with recent improvements such as DP-FTRL [151] and DP matrix factorization [86].

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. [291]. 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 reported negative results on using differential privacy to protect against property inference attacks which aim to extract properties of subpopulations in the training set [61, 195].

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, such as 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. Recently, a promising line of work is that of privacy auditing introduced by Jagielski et al. [145] with the goal of empirically measuring the actual privacy guarantees of an algorithm and determining privacy lower bounds by mounting privacy attacks. Auditing can be performed with membership inference attacks [146, 338], but poisoning attacks are much more effective and result in better estimates of the privacy leakage [145, 219]. Recent advances in privacy auditing include tighter bounds for the Gaussian mechanism [217], as well as rigorous statistical methods that allow the use of multiple canaries to reduce the sample complexity of auditing [240]. Additionally, two efficient methods for privacy auditing with training a single model have been proposed: Steinke et al. [281] use multiple random data canaries without incurring the cost of group privacy; and Andrew et al. [4] use multiple random client canaries and a cosine similarity test statistics to audit user-level private federated learning.
Differential privacy provides a rigorous notion of privacy, protecting 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.

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 exist in the literature. However, these techniques can be circumvented by motivated and well-resourced attackers and should be used with caution.

We refer the reader to available practice guides for securing machine learning deployments [57, 226]. 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 removal of their data from a trained ML model. Existing techniques for machine unlearning are either exact (e.g., retraining the model from scratch or from a certain checkpoint) [34, 41] or approximate (updating the model parameters to remove the influence of the unlearned records) [115, 139, 221].
3. Generative AI Taxonomy

Generative AI is a branch of AI that develops generative models with the capability of learning to generate content such as images, text, and other media with similar properties as their training data. Generative AI includes several different types of AI technologies with distinct origins, modeling approaches and related properties: GENERATIVE ADVERSARIAL NETWORKS, GENERATIVE PRE-TRAINED TRANSFORMER, and DIFFUSION MODEL. Recently, multi-modal AI systems have started to combine two or more technologies to enable multi-modal content generation capabilities [289].

3.1. Attack Classification

While many attack types in the PredAI taxonomy apply to GenAI (e.g., model poisoning, data poisoning, evasion, model extraction, etc.), a substantial body of recent work on the security of GenAI merits particular focus on novel security violations.

Figure 2 introduces a taxonomy of attacks in adversarial machine learning for GenAI systems. Similar to the PredAI taxonomy in Figure 1, this taxonomy is first categorized by the attacker’s objectives, which include availability breakdowns, integrity violations, and privacy compromise. For GenAI systems, violations of abuse are also especially relevant. 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. Multiple attack classes that require the same capabilities to reach the same objective are shown in a single callout.
An attack can be further categorized by the learning stage to which it applies and, subsequently, by the attacker’s knowledge and access. These are reviewed in the following sections. Where possible, the discussion broadly applies to GenAI with some specific areas that apply to LLMs (e.g., RETRIEVAL AUGMENTED GENERATION [RAG], which dominates many of the deployment stage attacks described below).

3.1.1. GenAI Stages of Learning

Due to the size of the models and training sets, predominant patterns in GenAI model development have departed from historical processes in which the entire process of data collection, labeling, model training, model validation, and model deployment are accomplished in a single pipeline by a single organization. Instead, foundation models are created during a pre-training stage that makes heavy use of unsupervised learning. The foundation
model encodes patterns (e.g., in text, images, etc.) that are useful for downstream tasks. The foundation models themselves are then the basis for creating task-specific applications via fine-tuning. In many cases, application developers begin with a foundation model developed by a third party and fine-tune it for their specific application. Attacks that correspond to the stages of GenAI application development are described in more detail below.

**Training-time attacks.** The training stage for GenAI often consists of two distinct stages: 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. This makes foundation models especially susceptible to poisoning attacks, in which an adversary controls a subset of the training data. Researchers have demonstrated that an attacker can induce targeted failures in models by arbitrarily poisoning only 0.001% of uncurated web-scale training datasets [42]. Executing web-scale dataset poisoning can be as simple as purchasing a small fraction of expired domains from known data sources [46]. Model fine-tuning may also be susceptible to poisoning attacks under the more common attacker knowledge and capabilities outlined in Section 2.1.

**Inference-time attacks.** The deployment stage for GenAI also differs from PredAI. How a model is used during deployment is application-specific. However, underlying many of the security vulnerabilities in LLMs and RAG applications is the fact that data and instructions are not provided in separate channels to the LLM, which allows attackers to use data channels to conduct inference-time attacks that are similar to decades-old SQL injection. Acknowledging a particular emphasis on LLMs, specifically for question-and-answering and text-summarization tasks, many of the attacks in this stage are due to the following practices that are common to applications of text-based generative models:

1. **Alignment via model instructions:** LLM behaviors are aligned at inference time through instructions that are prepended to the model’s input and context. These instructions comprise a natural language description of the model’s application-specific use case (e.g., “You are a helpful financial assistant that responds gracefully and concisely...”). A jailbreak overrides this explicit alignment and other safeguards. Since these prompts have been carefully crafted through prompt engineering, a prompt extraction attack may attempt to steal these system instructions. These attacks are also relevant to multimodal and text-to-image models.

2. **Contextual few-shot learning:** Since LLMs are autoregressive predictors, their performance in applications can be improved by providing examples of the inputs and outputs expected for the application in the model’s context that is prepended to the user query before evaluation by the LLM. This allows the model to more naturally complete the autoregressive tasks [37].

3. **Runtime data ingestion from third-party sources:** As is typical in retrieval augmented generation applications, context is crafted at runtime in a query-
dependent way and populated from external data sources (e.g., documents, web pages, etc.) that are to be summarized as part of the application. INDIRECT PROMPT INJECTION attacks depend on the attacker’s ability to modify the context using outside sources of information that are ingested by the system, even if not directly by the user.

4. **Output handling:** The output of an LLM may be used to populate an element on a web-page or to construct a command.

5. **Agents:** Plugins, functions, agents, and other concepts all rely on processing the *output* of the LLM (item 4) to perform some additional task and provide additional context to its input (item 3). In some cases, the LLM selects from among an appropriate set of these external dependencies based on a configuration provided in natural language and invokes that code with templates filled out by the LLM using information in the context.

![Figure 3. Retrieval-augmented generation relies on system instructions, context, and data from third-party sources, often through a vector database, to produce relevant responses for users](image)

3.1.2. **Attacker Goals and Objectives**

As with PredAI, attacker objectives can be classified broadly along the dimensions of availability, integrity, and privacy, as described in Section 2.1.2. However, there is a fourth attacker objective of abuse that is specific to GenAI:

**Abuse violations.** Abuse violations occur when an attacker repurposes a GenAI system’s intended use to achieve their own objectives. Attackers can use the capabilities of GenAI models to promote hate speech or discrimination, generate media that incites violence against specific groups, or scale offensive cybersecurity operations by creating images, text, or malicious code that enable a cyber attack.
3.1.3. Attacker Capabilities

Novel attacker capabilities that enable GenAI attacker objectives include:

- **TRAINING DATA CONTROL**: The attacker might take control of a subset of the training data by inserting or modifying training samples. This capability is used in data poisoning attacks.

- **QUERY ACCESS**: Many GenAI models and their applications (e.g., RETRIEVAL AUGMENTED GENERATION) are deployed as cloud-hosted services with access controlled through API keys. In this case, the attacker can submit queries to the model to receive an output. In GenAI, the purpose of submitting attacker-tuned inputs is to elicit a specific behavior from the model. This capability is used for PROMPT INJECTION, PROMPT EXTRACTION, and model stealing attacks.

- **SOURCE CODE CONTROL**: The attacker might modify the source code of the ML algorithm, such as the random number generator or any third-party libraries, which are often open source. The advent of open-source model repositories, like HuggingFace, allows attackers to create malicious models or wrap benign models with malicious code embedded in the deserialization format.

- **RESOURCE CONTROL**: The attacker might modify resources (e.g., documents, web pages) that will be ingested by the GenAI model at runtime. This capability is used for INDIRECT PROMPT INJECTION attacks.

3.2. AI Supply Chain Attacks and Mitigations

Studies on real-world security vulnerabilities against ML show that security is best addressed comprehensively, including software, data and model supply chains, and network and storage systems [7, 292]. Since AI is software, it inherits many of the vulnerabilities of the traditional software supply chain. However, many practical GenAI tasks begin with open-source models or data that have typically been out of scope for traditional cybersecurity. For example, ML repositories with the largest software vulnerability exposure include TensorFlow and OpenCV [292].

3.2.1. Deserialization Vulnerability

Many ML projects begin by downloading an open-source GenAI model for use in a downstream application. Most often, these models exist as artifacts persisted in pickle, pytorch, joblib, numpy, or tensorflow formats. Each of these formats allow for serialization persistence mechanisms that, in turn, allow for arbitrary code execution (ACE) when deserialized. ACE via deserialization is typically categorized as a critical vulnerability (e.g., CVE-2022-29216 for tensorflow, or CVE-2019-6446 for pickle in neural network tools) [292].
3.2.2. Poisoning Attacks

The performance of GenAI text-to-image and language models scales with model size and dataset size and quality. For example, scaling laws indicate that training a 500 billion parameter models would require 11 trillion tokens of training data [46]. Thus, it has become common for GenAI foundation model developers to scrape data from a wider range of uncurated sources. Dataset publishers only provide a list of URLs to constitute the dataset, and the domains serving those URLs can expire or be purchased, and the resources can be replaced by an attacker. As with PredAI models (discussed in Section 2.1), this could lead to TARGETED POISONING ATTACKS, BACKDOOR POISONING ATTACKS, and MODEL POISONING. A simple mitigation is for datasets to list both the URL and a cryptographic hash of the content that can be verified by the downloader. However, this technique may not scale up well for some of the large distributed datasets on the Internet - see Section 4.1 for further information.

3.2.3. Mitigations

AI supply chain attacks can be mitigated by supply chain assurance practices. For model file dependencies, this includes regular vulnerability scanning of the model artifacts used in the ML pipeline [292], and by adopting safe model persistence formats like safetensors. For webscale data dependencies, this includes verifying web downloads by publishing (by the provider) cryptographic hashes and verifying (by the downloader) training data as a basic integrity check to ensure that domain hijacking has not injected new sources of data into the training dataset [46]. Another approach to mitigating risks associated with malicious image editing by large diffusion models is immunizing images to make them resistant to manipulation by these models [254]. However, this approach requires an additional policy component to make it effective and practical.

3.3. Direct Prompt Injection Attacks and Mitigations

A direct prompt injection occurs when a user injects text that is intended to alter the behavior of the LLM.

Attacker goals. An attacker may have a variety of goals with a prompt injection [182, 183, 265], such as:

- **Abuse.** Attackers use direct prompt injections to bypass safeguards in order to create misinformation, propaganda, hurtful content, sexual content, malware (code), or phishing content. Often, the list of prohibited scenarios is explicitly safeguarded by the model creator [5, 121, 230]. A direct prompt injection for the purpose of model abuse is also called a JAILBREAK.

- **Invade privacy.** Attackers may wish to extract the system prompt or reveal private information provided to the model in the context but not intended for unfiltered access by the user. This is discussed further in Section 3.3.1.
**Attacker techniques.** Attacker techniques for launching direct prompt injection attacks are numerous but tend to fall into several broad categories [319]:

- **Gradient-based attacks** are white-box optimization-based methods for designing jailbreaks that are very similar to the PredAI attacks discussed in Section 2.2.1. A gradient-based distributional attack uses an approximation to make the adversarial loss for generative transformer models differentiable, which aims to minimize lexical changes by enforcing perceptibility and fluency via BERTScore and perplexity [127]. HotFlip encodes modifications of text into a binary vector and gradient steps to minimize adversarial loss [97]. Originally designed to create adversarial examples for PredAI language classifiers (e.g., sentiment analysis), subsequent works have leveraged HotFlip for GenAI using the following trick: since these autoregressive tokens generate a single token at a time, optimizing the first generated token to produce an affirmative response is often sufficient to prime the autoregressive generative process to complete a fully affirmative utterance [49]. Universal adversarial triggers are a special class of these gradient-based attacks against generative models that seek to find *input-agnostic* prefixes (or suffixes) that, when included, produce the desired affirmative response regardless of the remainder of the input [308, 354]. That these universal triggers transfer to other models makes open-source models — for which there is ready white-box access — feasible attack vectors for transferability attacks to closed systems where only API access is available [354].

- **Manual methods** for jailbreaking an LLM generally fall into two categories: competing objectives and mismatched generalization [316]. These methods often exploit the model’s susceptibility to certain linguistic manipulations and extend beyond conventional adversarial inputs. In the category of competing objectives, additional instructions are provided that compete with the instructions originally provided by the author.

1. **Prefix injection**: This method involves prompting the model to commence responses with an affirmative confirmation. By conditioning the model to begin its output in a predetermined manner, adversaries attempt to influence its subsequent language generation toward specific, predetermined patterns or behaviors.

2. **Refusal suppression**: Adversaries provide explicit instructions to the model, compelling it to avoid generating refusals or denials in its output. By limiting or prohibiting the generation of negative responses, this tactic aims to ensure the model’s compliance with the provided instructions, potentially compromising safety measures.

3. **Style injection**: In this approach, adversaries instruct the model not to use long words or adopt a particular style. By constraining the model’s language to simplistic or non-professional tones, it aims to limit the sophistication or accuracy of the model’s responses, thereby potentially compromising its overall performance.
4. **Role-play**: Utilizing role-play strategies, such as “Do Anything Now” (DAN) or “Always Intelligent and Machiavellian” (AIM), adversaries guide the model to adopt specific personas or behavioral patterns that conflict with the original intent. This manipulation aims to exploit the model’s adaptability to varied roles or characteristics, potentially compromising its adherence to safety protocols.

Techniques in the mismatched generalization category diverge significantly from any safety training or guardrails, positioning inputs to be out of distribution from the model’s standard training data. Approaches include:

1. **Special encoding**: Adversarial inputs often employ encoding techniques like base64 encoding. This method alters the representation of input data and renders it unrecognizable to standard recognition algorithms. By encoding information, adversaries aim to deceive the model’s understanding of the input and bypass its safety mechanisms.

2. **Character transformation**: Techniques like ROT13 cipher, symbol replacement (e.g., l33tspeak), and Morse code manipulate the characters of the input text. These transformations aim to obscure the original meaning of the text, potentially confusing the model’s interpretation and enabling adversarial inputs to evade detection.

3. **Word transformation**: Strategies that aim to alter the linguistic structure may include Pig Latin, synonym swapping (e.g., using “pilfer” for “steal”), and payload splitting (or “token smuggling”) to break down sensitive words into substrings. These manipulations intend to deceive the model’s safeguards in a way that is still comprehended by the LLM.

4. **Prompt-level obfuscation**: Adversaries employ methods like translation to other languages to make the model obfuscate or summarize content in a manner that it may not fully comprehend. These obfuscations introduce ambiguity or altered linguistic contexts and create input scenarios in which the model’s safety mechanisms are less effective due to a lack of clarity or misinterpretation.

- **Automated model-based red teaming** employs three models: an attacker model, a target model, and a judge [60, 199, 237]. When the attacker has access to a high quality classifier to judge whether model output is harmful, it may be used as a reward function to train a generative model to generate jailbreaks of another generative model. Only query access is required for each of the models, and no human intervention is required to update or refine a candidate jailbreak. Empirically, these algorithms may be orders of magnitude more query-efficient than existing algorithms, requiring only dozens of queries per successful jailbreak. The prompts have also been shown to be transferable from the target model to other closed-source LLMs [60].
3.3.1. Data Extraction

GenAI models are trained on data that may include proprietary or sensitive information. GenAI applications may also be instrumented with carefully crafted prompts or — as with RAG — be supplied with sensitive information in their context for summarization or other task completion. Attacker techniques for extracting this information are the subject of ongoing research for both LLMs [174, 348] and text-to-image models [266].

Leaking sensitive information. Carlini et al. [48] were the first to practically demonstrate data extraction attacks in generative language models. By inserting synthetic canaries in the training data, they developed a methodology for extracting the canaries and introduced a metric called exposure to measure memorization. Subsequent work demonstrated the risk of data extraction in large language models based on transformers, such as GPT-2 [51], by prompting the model with different prefixes and mounting a membership inference attack to determine which generated content was part of the training set. Since these decoder stack transformers are autoregressive models, a verbatim textual prefix about personal information can result in the model completing the text input with sensitive information that includes email addresses, phone numbers, and locations [191]. This behavior of verbatim memorization of sensitive information in GenAI language models has also been observed in more recent transformer models with the additional characterization of extraction methods [132]. Unlike PredAI models, in which carefully crafted tools like Text Revealer are created to reconstruct text from transformer-based text classifiers [343], GenAI models can simply be asked to repeat private information that exists in the context as part of the conversation. Results show that information like email addresses can be revealed at rates exceeding 8%. However, their responses may wrongly assign the owner of the information. In general, extraction attacks are more successful when the model is seeded with more specific and complete information — the more the attacker knows, the more they can extract. Intuitively, larger models with more capacity are more susceptible to exact reconstruction [45].

Prompt and context stealing. Prompts are vital to align LLMs to a specific use case and are a key ingredient to their utility in following human instructions. Well-crafted prompts enable LLMs to be smart assistants with external applications and provide instructions for human alignment. These prompts are of high value and are usually regarded as commercial secrets. Successful prompt-stealing attacks may violate the intellectual property and privacy of prompt engineers or jeopardize the business model of prompt trading marketplaces. PromptStealer is a learning-based method that reconstructs prompts from text-to-image models using an image captioning model and a multi-label classifier to steal both the subject and the prompt modifiers [266]. For LLMs, researchers have found that a small set of fixed attack queries (e.g., Repeat all sentences in our conversation) were sufficient to extract more than 60% of prompts across all model and dataset pairs [348]. In RAG applications (see Figure 3), the same techniques can be used to extract sensitive information provided in the LLMs context. For example, rows from a database or text from a PDF document that are intended to be summarized generically by the LLM can be
3.3.2. Mitigations

Various defense strategies have been proposed for prompt injection that provide a measure of protection but not full immunity to all attacker techniques. These broadly fall into the following categories:

1. **Training for alignment.** Model providers continue to create built-in mechanisms by training with stricter *forward* alignment [148]. For example, model alignment can be tuned by training on carefully curated and prealigned datasets. It can then be iteratively improved through reinforcement learning with human feedback [123].

2. **Prompt instruction and formatting techniques.** LLM instructions can cue the model to treat user input carefully [168, 182]. For example, by appending specific instructions to the prompt, the model can be informed about subsequent content that may constitute a jailbreak. Positioning the user input before the prompt takes advantage of recency bias in following instructions. Encapsulating the prompt in random characters or special HTML tags provides cues to the model about what constitutes system instructions versus user prompts.

3. **Detection techniques.** Model providers continue to create built-in mechanisms by training with stricter *backward alignment* via evaluation on specially crafted benchmark datasets or filters that monitor the input to and output of a protected LLM [148]. One proposed method is to evaluate a distinctly prompted LLM that can aid in distinguishing potentially adversarial prompts [168, 182]. Several commercial products have begun offering tools to detect prompt injection, both by detecting potentially malicious user input and by moderating the output of the firewall for jailbreak behavior [8, 166, 247]. These may provide supplementary assurance through a defense-in-depth philosophy.

Similarly, defenses for prompt stealing have yet to be proven rigorous. A commonality in the methods is that they compare the model utterance to the prompt, which is known by the system provider. Defenses differ in how this comparison is made, which might include looking for a specific token, word, or phrase, as popularized by [48], or comparing the n-grams of the output to the input [348].

3.4. Indirect Prompt Injection Attacks and Mitigations

A dominant use case for LLMs is RETRIEVAL AUGMENTED GENERATION, depicted in Figure 3. Using LLMs in retrieval tasks has blurred the *data* and *instruction* channels to an LLM. This allows for attacker techniques that leverage the data channel to affect system operation, similar to decades-old SQL injection attacks. However, the attacker need not directly manipulate the LLM. INDIRECT PROMPT INJECTION attacks are enabled by RESOURCE CONTROL so that an attacker can indirectly (or remotely) inject system
prompts without directly interacting with the RAG application [122]. As with direct prompt injection, indirect prompt injection attacks can result in violations across the four categories of attacker goals: 1) availability violations, 2) integrity violations, 3) privacy compromises, and 4) abuse violations.

3.4.1. Availability Violations

Model availability violations are a disruption in service that can be caused by an attacker prompting a model with maliciously crafted inputs that cause increased computation or by overwhelming the system with a number of inputs that causes a denial of service to users. As noted in [122], these attack vectors are particularly interesting when executed via indirect prompt injection so that a resource rather than a registered user is the source of the availability violation. Availability attacks that increase computation make the model or service perform unusually slow [122]. In PredAI, this has most commonly been done by optimizing sponge examples — an energy latency attack on neural networks [33]. However, with recent LLMs, this could be done by simply instructing the model to complete a time-sensitive task in the background, slowing the model down [122]. Denial-of-service attacks can indiscriminately render a model unusable (e.g., failure to generate helpful outputs) or specifically block certain capabilities (e.g., specific APIs) [122].

Attacker Techniques. Researchers have demonstrated the following attacks on a commercial RAG service [122] via indirect prompt injection, in which a resource to be searched or summarized contained instructions with certain characteristics:

- **Time-consuming background tasks.** The prompt instructs the model to perform a time-consuming task prior to answering the request. The prompt itself can be brief and request looping behavior in evaluating models [122].

- **Muting.** This attack exploits the fact that a model cannot finish sentences when an `<endoftext>` token appears in the middle of a user’s request. By including a request to begin a sentence with this token, a search agent, for example, will return without any generated text [122].

- **Inhibiting capabilities.** In this attack, an embedded prompt instructs the model that it is not permitted to use certain APIs (e.g., the search functionality for Bing Chat). This selectively disarms key components of the service [122].

- **Disrupting input or output.** In this attack, an indirect prompt injection instructs the model to replace characters in retrieved text with homoglyph equivalents, disrupting calls to APIs that depend on the text. Alternatively, the prompt can instruct the model to corrupt the results of a query to result in a useless retrieval or summary [122].
3.4.2. Integrity Violations

Integrity violations are threats that cause GenAI systems to become untrustworthy. AI chatbots have exacerbated online misinformation, as demonstrated by the tendencies of Microsoft’s Bing and Google’s Bard to perpetuate each other’s sources of misinformation [307]. LLMs’ inability to gauge reliable sources of news and information can be exploited to produce factually unsound outputs.

Attacker Techniques. Researchers have demonstrated integrity attacks by manipulating the primary task of the LLM. This is different than the more common indirect prompt injection attacks that perform a malicious side task.

• Manipulation. The model is prompted to provide incorrect summaries of the search result (i.e., arbitrarily wrong summaries) when the wrong output is not chosen in advance [122]. The manipulation attack instructs the model to provide wrong answers and causes the model’s answer to make claims that contradict the cited sources [122]. Below are two examples of manipulation attacks:

  1. Wrong summaries. A model can be prompted to produce adversarially chosen or arbitrarily wrong summaries of documents, emails, or search queries [122].
  2. Propagate disinformation. Search chatbots can be prompted to propagate disinformation by relying on or perpetuating untrustworthy news sources or the outputs of other search chatbots [307].

3.4.3. Privacy Compromises

Indirect prompt injections introduce a host of new privacy compromises and concerns. From the beginning, LLMs raised concerns about privacy. Italy was early to ban the use of ChatGPT due to these very concerns, and there have been many reported cases of chatbots leaking sensitive information or chat histories [87, 197]. Attacker goals are divided into two key categories:

1. Information gathering. Specific attacks can heighten these risks. For example, human-in-the-loop indirect prompting can be used to extract user data (e.g., personal information, credentials) or leak their chat histories by interacting in chat sessions and persuading users to divulge information or through side channels [122]. An attack example that does not involve a human-in-the-loop is an attack against a personal assistant that can access a user’s data (e.g., real emails), which similarly causes privacy concerns [122].

2. Unauthorized disclosure. Models are commonly integrated into system infrastructures, giving way to unauthorized disclosures or privileges to private user data. Malicious actors can leverage backdoor attacks to gain access to LLMs or systems using a variety of methods (e.g., issuing API calls, malicious code auto-completions) [122].
Attacker Techniques. To highlight privacy concerns, researchers have demonstrated a data-stealing attack by designing an injection that instructs the LLM to persuade the end user to divulge their real name [122]. Below are some attack techniques that researchers have used to achieve this data-stealing attack:

- **Human-in-the-loop indirect prompting.** Read operations (e.g., triggering a search query that then makes a request to the attacker, or retrieving URLs directly) are exploited to send information to the attacker [122].

- **Interacting in chat sessions.** The model persuades a user to follow a URL into which the attacker inserts the user’s name [122].

- **Invisible markdown image.** A prompt injection is performed on a chatbot by modifying the chatbot answer with an invisible single-pixel markdown image that withdraws the user’s chat data to a malicious third party [255].

3.4.4. Abuse Violations

As previously discussed, GenAI introduces a new category of attacker goal: abuse violations. This broadly refers to when an attacker repurposes a system’s intended use to achieve their own objectives by way of indirect prompt injection. This attacker goal can be divided into the following primary categories:

1. Fraud. Recent advances in instruction-following LLMs have caused simultaneous advances in dual-use risks.

2. Malware. LLMs can prominently facilitate the spread of malware by suggesting malicious links to the user. Additionally, the proliferation of LLM-integrated applications has led to new malware threats by forcing the prompts themselves to act as malware [122]. For example, LLM-augmented email clients that read emails are likely to deliver malicious prompts and then send emails proliferating those prompts [122].

3. Manipulation. Models currently act as vulnerable intermediary layers between users and information outputs that are easy to manipulate. LLMs are now commonly part of a larger system and integrate with applications. This intermediary state can expose the model to a host of vulnerabilities. For example, a) search chatbots can be prompted to generate disinformation b) and prompted to hide specific information, sources, or search queries; and c) models can be prompted to provide adversarially chosen or arbitrarily wrong summaries of information sources (e.g., documents, emails, search queries) [122]. While users are already prone to trusting untrustworthy sources on the web, the authoritative tone of LLMs and users’ over-reliance on their impartiality have the potential to cause users to succumb to these manipulation attempts more often [122].
**Attacker Techniques.** Researchers have demonstrated examples of different abuse attack techniques by conducting experiments with chatbots (e.g., Microsoft’s Bing chatbot).

- **Phishing.** Previously, it had been demonstrated that LLMs could produce convincing scams, such as phishing emails [153]. Now that LLMs can more easily integrate with applications, they can not only enable the creation of scams but also widely disseminate such attacks [122]. Users are likely to be more susceptible to these new attacks, unlike phishing emails, because they lack experience and awareness of this new threat technique.

- **Masquerading.** LLMs can pretend to be an official request from a service provider or recommend a fraudulent website as trusted [122].

- **Spreading injections.** The LLM itself acts as a computer running and spreading harmful code. For example, an automatic message processing tool that can read and compose emails and look at users’ personal data can spread an injection to other models that may be reading those inbound messages [122].

- **Spreading malware.** Similar to phishing, LLMs can be exploited to persuade users to visit malicious web pages that lead to “drive-by downloads.” This is further enabled by markdown links that could be seamlessly generated as part of the answer [122].

- **Historical distortion.** An attacker can prompt the model to output adversarially chosen disinformation. Researchers have demonstrated this by successfully prompting Bing Chat to deny that Albert Einstein won a Nobel Prize [122].

- **Marginally related context prompting.** Steering search results toward specific orientations instead of neutral stances can create an attack to achieve bias amplification. Researchers have demonstrated this by prompting an LLM with biographies of personas that are either “conservative” or “liberal” and instructing the model to generate answers that agree with the views of the described users without explicitly mentioning topics [122].

### 3.4.5 Mitigations

Various mitigation techniques have been proposed for indirect prompt injection attacks that help eliminate model risk but — like the suggestions made for direct prompt injections — do not offer full immunity to all attacker techniques. These mitigation strategies fall into the following categories:

1. Reinforcement learning from human feedback (RLHF). RLHF is a type of AI model training whereby human involvement is indirectly used to fine-tune a model. This can be leveraged to better align LLMs with human values and prevent unwanted behaviors [122]. OpenAI’s GPT-4 was fine-tuned using RLHF and has shown a lesser tendency to produce harmful content or hallucinate [229].
2. Filtering retrieved inputs. A further defense mechanism proposed by Greshake et al. suggests processing the retrieved inputs to filter out instructions [122].

3. An LLM moderator. An LLM can be leveraged to detect attacks beyond just filtering clearly harmful outputs. This could be beneficial in detecting attacks that do not depend on retrieved sources but could fail at detecting disinformation or other kinds of manipulation attacks [122].

4. Interpretability-based solutions. These solutions perform outlier detection of prediction trajectories [122]. Researchers have demonstrated that the prediction trajectory of the tuned lens on anomalous inputs could be used to detect anomalous inputs [20].

Unfortunately, there is no comprehensive or foolproof solution for protecting models against adversarial prompting, and future work will need to be dedicated to investigating suggested defenses for their efficacy.
4. Discussion and Remaining Challenges

4.1. The Scale Challenge

Data is fundamentally important for training models. As models grow, the amount of training data grows proportionally. This trend is clearly visible in the evolution of LLM’s. Very few of the LLM’s in use today publish a detailed list of the data sources used in training but those who do [209, 293] show the scale of the footprint and the massive amounts of data consumed in training. The most recent multi-modal GenAI systems exacerbate the demand further by requiring large amounts of data for each modality.

There is no single organization, even nation, that contains the full data used for training a given LLM. Data repositories are not monolithic data containers but a list of labels and data links to other servers that actually contain the corresponding data samples. This renders the classic definition of the corporate cybersecurity perimeter obsolete and creates new hard-to-mitigate risks [46]. Recently published open source data poisoning tools [202] increase the risk of large scale attacks on image training data. Although created with noble intentions, to allow artists to protect the copyright of their work, these tools become harmful if they fall in the hands of people with malicious intent.

Another scale-related problem is the ability to generate synthetic content at scale on the internet. Although WATERMARKING [158] may alleviate the situation, the existence of powerful open or other ungoverned models creates opportunities to generate massive amounts of unmarked synthetic content that can have a negative impact on the capabilities of subsequently trained LLMs [272], leading to model collapse.

Finding ways to tackle these challenges of scale is critically important for evolving the capabilities of foundation models in the future, especially if we want them aligned with human values.

4.2. Theoretical Limitations on Adversarial Robustness

Given the multitude of powerful attacks, designing appropriate mitigations is a challenge that needs to be addressed before deploying AI systems in critical domains. This challenge is exacerbated by the lack of information-theoretically secure machine learning algorithms for many tasks in the field, as we discussed in Section 1. This implies that presently designing mitigations is an inherently ad hoc and fallible process. We refer the readers to available practice guides for securing machine learning deployments [57, 226], as well as existing guidelines for mitigating AML attacks [98].

One of the ongoing challenges facing the AML field is the ability to detect when the model is under attack. Knowing this would provide an opportunity to counter the attack before any information is lost or an adverse behaviour is triggered in the model. Tramèr [295] has shown that designing techniques to detect adversarial examples is equivalent to robust classification, which is inherently hard to construct, up to computational complexity and a
factor of 2 in the robustness radius.

Adversarial examples may be from the same data distribution on which the model is trained and to which it expects the inputs to belong or may be OUT-OF-DISTRIBUTION (OOD) inputs. Thus, the ability to detect OOD inputs is also an important challenge in AML. Fang et al. [102] established useful theoretical bounds on detectability, particularly an impossibility result when there is an overlap between the in-distribution and OOD data.

The data and model sanitization techniques discussed in Section 2.3 reduce the impact of a range of poisoning attacks and should be widely used. However, they should be combined with cryptographic techniques for origin and integrity attestation to provide assurances downstream, as recommended in the final report of the National Security Commission on AI [220].

As pointed out in the Introduction, chatbots [70, 83, 206, 227] enabled by recent advances in deep learning have emerged as a powerful technology with great potential for numerous business applications, from entertainment to more critical fields. Recently, specific attacks using "PROMPT INJECTIONS" have emerged as effective ways to trigger bad behaviour in the bot [306] and imposing guardrails has been the widely used approach to mitigating these risks. However, recent research has discovered theoretical limits on rigorous LLM censorship [116] that require employing other means of risk mitigation, e.g., setting up controlled model gateways and other cybersecurity mechanisms.

Despite progress in the ability of chatbots to perform well on certain tasks [227], this technology is still emerging and should only be deployed in applications that require a high degree of trust in the information they generate with abundance of caution and continuous monitoring.

As the development of AI-enabled chatbots continues and their deployment becomes more prevalent online and in business applications, these concerns will come to the forefront and be pursued by adversaries to discover and exploit vulnerabilities and by companies developing the technology to improve their design and implementation to protect against such attacks [354]. The identification and mitigation of a variety of risk factors, such as vulnerabilities, include RED TEAMING [56, 109] as part of pre-deployment testing and evaluation of LLM’s. These processes vary and have included testing for traditional cybersecurity vulnerabilities, bias, and discrimination, generation of harmful content, privacy violations, and novel or emergent characteristics of large-scale models, as well as evaluations of larger societal impacts such as economic impacts, the perpetuation of stereotypes, long-term over-reliance, and erosion of democratic norms [157, 267].

Realistic risk management throughout the entire life cycle of the technology is critically important to identify risks and plan early corresponding mitigation approaches [226]. For example, incorporating human adversarial input in the process of training the system (i.e., RED TEAMING) or employing reinforcement learning from human feedback appear to of-
fer benefits in terms of making the chatbot more resilient against toxic input or prompt injections [83]. However, adapting chatbots to downstream use cases often involves the customization of the pre-trained LLM through further fine-tuning, which introduces new safety risks that may degrade the safety alignment of the LLM [242]. Barrett et al. [17] have developed detailed risk profiles for cutting-edge generative AI systems that map well to the NIST AI RMF [226] and should be used for assessing and mitigating potentially catastrophic risks to society that may arise from this technology. There are also useful industry resources for managing foundation model risks [32].

The robust training techniques discussed in Section 2.3 offer different approaches to providing theoretically certified defenses against data poisoning attacks with the intention of providing much-needed information-theoretic guarantees for security. The results are encouraging, but more research is needed to extend this methodology to more general assumptions about the data distributions, the ability to handle OOD inputs, more complex models, multiple data modalities, and better performance. Another challenge is applying these techniques to very large models like LLMs and generative diffusion models, which are quickly becoming targets of attacks [44, 75].

Another general problem of AML mitigations for both evasion and poisoning attacks is the lack of reliable benchmarks which causes results from AML papers to be routinely incomparable, as they do not rely on the same assumptions and methods. While there have been some promising developments into this direction [81, 256], more research and encouragement is needed to foster the creation of standardized benchmarks to allow gaining reliable insights into the actual performance of proposed mitigations.

Formal methods verification has a long history in other fields where high assurance is required, such as avionics and cryptography. The lessons learned there teach us that although the results from applying this methodology are excellent in terms of security and safety assurances, they come at a very high cost, which has prevented formal methods from being widely adopted. Currently, formal methods in these fields are primarily used in applications mandated by regulations. Applying formal methods to neural networks has significant potential to provide much-needed security guarantees, especially in high-risk applications. However, the viability of this technology will be determined by a combination of technical and business criteria – namely, the ability to handle today’s complex machine learning models of interest at acceptable costs. More research is needed to extend this technology to all algebraic operations used in machine learning algorithms, to scale it up to the large models used today, and to accommodate rapid changes in the code of AI systems while limiting the costs of applying formal verification.

There is an imbalance between the large number of privacy attacks listed in Section 2.4 (i.e., memorization, membership inference, model extraction, and property inference) and available reliable mitigation techniques. In some sense, this is a normal state of affairs: a rapidly evolving technology gaining widespread adoption – even “hype” – which attracts the attention of adversaries, who try to expose and exploit its weaknesses before the tech-
Technology has matured enough for society to assess and manage it effectively. To be sure, not all adversaries have malevolent intent. Some simply want to warn the public of potential breakdowns that can cause harm and erode trust in the technology. Additionally, not all attacks are as practical as they need to be to pose real threats to AI system deployments of interest. Yet the race between developers and adversaries has begun, and both sides are making great progress. This poses many difficult questions for the AI community of stakeholders, such as:

- What is the best way to mitigate the potential exploits of memorized data from Section 3.3.1 as models grow and ingest larger amounts of data?
- What is the best way to prevent attackers from inferring membership in the training set or other properties of the training data using the attacks listed in Sections 2.4.2 and 2.4.4?
- How can developers protect their ML models with the secret weights and associated intellectual property from the emerging threats in the PredAI and GenAI spaces? Especially, attacks that utilize the public API of the ML model to query and exploit its secret weights or the side-channel leakage attacks from Section 2.4.3? The known mechanisms of preventing large numbers of queries through the API are ineffective in configurations with anonymous or unauthenticated access to the model.

As answers to these questions become available, it is important for the community of stakeholders to develop specific guidelines to complement the NIST AI RMF [226] for use cases where privacy is of utmost importance.

### 4.3. The Open vs. Closed Model Dilemma

Open source has established itself as an indispensable methodology for developing software today. There are many benefits to open source development that have been widely analysed [170].

Following this model and adding valid arguments related to democratizing access, leveling the playing field, enabling reproducibility of scientific results that in turn enables measuring progress in AI, powerful open access models have become available to the public [209, 293, 294]. In many use cases they help to bridge the performance gaps with closed/proprietary models [178, 303].

However, there are other use cases where putting powerful AI technology in the hands of people with malicious intent would be very concerning [290]. Researchers have already demonstrated the ease with which open models can be subverted to perform tasks outside of the original intent of the developers [330]. This brings up the question about open models: should they be allowed?

This question has been approached in other fields of science and engineering. For example, society has accepted the risks of cryptography falling in the wrong hands and we have
strong cryptographic algorithms publicly available and widely used. In another example, in bio engineering, society has determined that the risks of uncontrolled genetic engineering are too great to allow open access to the technology.

The open vs. closed model dilemma in AI is being actively debated in the community of stakeholders and should be resolved before models become too powerful and make it moot.

4.4. Supply chain challenges

The literature on AML shows a trend of designing new attacks with higher power and stealthier behavior. The advent of powerful open models [293] and the reported attacks against them [330] can induce a behavior of a model with TROJANS. This illustrates the challenges for applications using open models downstream the supply chain. To be clear, backdoor attacks on models is not limited to open models.

DARPA jointly with NIST have created a program TrojAI that is researching the defense of AI systems from intentional, malicious Trojan attacks by developing technology to detect these attacks and by investigating what makes the Trojan detection problem challenging.

Goldwasser et al. [118] recently introduced a new class of attacks: information-theoretically undetectable Trojans that can be planted in ML models. If proven practical, such attacks may only be prevented or detected and mitigated by procedures that restrict and control who in the organization has access to the model throughout the life cycle and by thoroughly vetting third-party components coming through the supply chain. The NIST AI Risk Management Framework [226] offers more information on this.

4.5. Tradeoffs Between the Attributes of Trustworthy AI

The trustworthiness of an AI system depends on all of the attributes that characterize it [226]. For example, an AI system that is accurate but easily susceptible to adversarial exploits is unlikely to be trusted. Similarly, an AI system that produces harmfully biased or unfair outcomes is unlikely to be trusted even if it is robust. There are also trade-offs between explainability and adversarial robustness [140, 207]. In cases where fairness is important and privacy is necessary to maintain, the trade-off between privacy and fairness needs to be considered [142]. Unfortunately, it is not possible to simultaneously maximize the performance of the AI system with respect to these attributes. For instance, AI systems optimized for accuracy alone tend to underperform in terms of adversarial robustness and fairness [59, 91, 244, 301, 342]. Conversely, an AI system optimized for adversarial robustness may exhibit lower accuracy and deteriorated fairness outcomes [21, 311, 342].

The full characterization of the trade-offs between the different attributes of trustworthy AI is still an open research problem that is gaining increasing importance with the adoption of AI technology in many areas of modern life.
In most cases, organizations will need to accept trade-offs between these properties and decide which of them to prioritize depending on the AI system, the use case, and potentially many other considerations about the economic, environmental, social, cultural, political, and global implications of the AI technology [226].

4.6. Multimodal Models: Are They More Robust?

MULTIMODAL MODELS have shown great potential for achieving high performance on many machine learning tasks [16, 19, 213, 246, 344]. It is natural to assume that because there is redundancy of information across the different modalities, the model should be more robust against adversarial perturbations of a single modality. However, emerging evidence from practice shows that this is not necessarily the case. Combining modalities and training the model on clean data alone does not seem to improve adversarial robustness. In addition, one of the most effective defenses against evasion attacks based on adversarial training, which is widely used in single modality applications, is prohibitively expensive in practical applications of multimodal learning. Additional effort is required to benefit from the redundant information in order to improve robustness against single modality attacks [328]. Without such an effort, single modality attacks can be effective and compromise multimodal models across a wide range of multimodal tasks despite the information contained in the remaining unperturbed modalities [328, 335]. Moreover, researchers have devised efficient mechanisms for constructing simultaneous attacks on multiple modalities, which suggests that multimodal models might not be more robust against adversarial attacks despite improved performance [63, 261, 326].

The existence of simultaneous attacks on multimodal models suggests that mitigation techniques that only rely on single modality perturbations are not likely to be robust. Attackers in real life do not constrain themselves to attacks within a given security model but employ any attack that is available to them.

4.7. Quantized models

Quantization is a technique that allows efficiently deploying models to edge platforms such as smart phones and IoT devices [114]. It reduces the computational and memory costs of running inference on a given platform by representing the model weights and activations with low-precision data types. For example, quantized models typically use 8-bit integers (int8) instead of the usual 32-bit floating point (float32) numbers for the original non-quantized model.

This technique has been widely used with PredAI and increasingly with GenAI models. However, quantized models do inherit the vulnerabilities of the original models and bring in additional weaknesses making such models vulnerable to adversarial attacks. Error amplification resulting from the reduced computational precision affects adversely the adversarial robustness of the quantized models. Some pointers to useful mitigation techniques
for PredAI models exist in the literature [179]. The effects of quantization on GenAI models has been studies lees and organizations deploying such models should be careful to continuously monitor their behavior.
References


Machinery.


[97] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. Hotflip: White-box ad-

[98] ETSI Group Report SAI 005. Securing artificial intelligence (SAI); mitigation strategy report, retrieved February 2023 from https://www.etsi.org/deliver/etsi_gr/SAI/001_099/005/01.01.01_60/gr_SAI005v010101p.pdf.


[152] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L’D’Oliveira, Hubert Eichner, Salim El Rouayheb,


USA, 2021. Association for Computing Machinery.


view.com/2022/08/31/1058800/what-does-gpt-3-know-about-me/, August 2022. MIT Technology Review.


[214] Sasi Kumar Murakonda and Reza Shokri. ML Privacy Meter: Aiding regulatory
compliance by quantifying the privacy risks of machine learning, 2020.


[225] Thien Duc Nguyen, Phillip Rieger, Huili Chen, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Shaza


[262] Ryan Sheatsley, Blaine Hoak, Eric Pauley, Yohan Beugin, Michael J. Weisman, and


[305] Sridhar Venkatesan, Harshvardhan Sikka, Rauf Izmailov, Ritu Chadha, Alina Oprea,


Appendix: Glossary

Note: one may click on the page number shown at the end of the definition of each glossary entry to go to the page where the term is used.

A

adversarial examples (adversarial examples) Modified testing samples which induce misclassification of a machine learning model at deployment time v, 9

adversarial success (adversarial success) Indicates reaching an availability breakdown, integrity violations, privacy compromise, or abuse trigger (for GenAI models only) in response to attempted adversarial attacks on the model 9

Area Under the Curve (Area Under the Curve) In ML the Area Under the Curve (AUC) is a measure of the ability of a classifier to distinguish between classes. The higher the AUC, the better the performance of the model at distinguishing between the two classes. AUC measures the entire two-dimensional area underneath the RECEIVER OPERATING CHARACTERISTICS (ROC) curve 31

availability attack (availability attack) Adversarial attacks against machine learning which degrade the overall model performance 9

B

backdoor pattern (backdoor pattern) A trigger pattern inserted into a data sample to induce mis-classification of a poisoned model. For example, in computer vision it may be constructed from a set of neighboring pixels, e.g., a white square, and added to a specific target label. To mount a backdoor attack, the adversary first poisons the data by adding the trigger to a subset of the clean data and changing their corresponding labels to the target label 9

backdoor poisoning attacks (backdoor poisoning attacks) Poisoning attacks against machine learning which change the prediction on samples including a backdoor pattern 9, 40

C

classification (classification) Type of supervised learning in which data labels are discrete 8

convolutional neural networks (convolutional neural networks) A Convolutional Neural Network (CNN) is a class of artificial neural networks whose architecture connects neurons from one layer to the next layer and includes at least one layer performing convolution operations. CNNs are typically applied to image analysis and classification. See [119] for further details 8, 32

D

data poisoning (data poisoning) Poisoning attacks in which a part of the training data is under the control of the adversary 4, 8
data privacy (data privacy) Attacks against machine learning models to extract sensitive information about training data 10

data reconstruction (data reconstruction) Data privacy attacks which reconstruct sensitive information about training data records 10, 29

deployment stage (deployment stage) Stage of ML pipeline in which the model is deployed on new data 8, 37

Diffusion Model (Diffusion Model) A class of latent variable generative models consisting of three major components: a forward process, a reverse process, and a sampling procedure. The goal of the diffusion model is to learn a diffusion process that generates the probability distribution of a given dataset. It is widely used in computer vision on a variety of tasks, including image denoising, inpainting, super-resolution, and image generation 35

discriminative (discriminative) Type of machine learning methods which learn to discriminate between classes 8

E

energy-latency attacks (energy-latency attacks) Attacks that exploit the performance dependency on hardware and model optimizations to negate the effects of hardware optimizations, increase computation latency, increase hardware temperature and massively increase the amount of energy consumed 9, 10

ensemble learning (ensemble learning) Type of a meta machine learning approach that combines the predictions of several models to improve the performance of the combination 8

Expectation Over Transformation (Expectation Over Transformation) Expectation Over Transformation (EOT) helps to strengthen adversarial examples to remain adversarial under image transformations that occur in the real world, such as angle and viewpoint changes. EOT models such perturbations within the optimization procedure. Rather than optimizing the log-likelihood of a single example, EOT uses a chosen distribution of transformation functions taking an input controlled by the adversary to the “true” input perceived by the classifier 18

extraction (extraction) The ability of an attacker to extract training data of a generative model by prompting the model on specific inputs 10

F

federated learning (federated learning) Type of collaborative machine learning, in which multiple users train jointly a machine learning model 8

federated learning models (federated learning models) Federated learning is a methodology to train a decentralized machine learning model (e.g., deep neural networks or a pre-trained large language model) across multiple end-devices without sharing the data residing on each device. Thus, the end-devices collaboratively train a global model by exchanging model updates with a server that aggregates the updates. Compared to traditional centralized learning where the data are pooled,
federated learning has advantages in terms of data privacy and security but these may come as tradeoffs to the capabilities of the models learned through federated data. Other potential problems one needs to contend with here concern the trustworthiness of the end-devices and the impact of malicious actors on the learned model 32

**feed-forward neural networks** (feed-forward neural networks) A Feed Forward Neural Network is an artificial neural network in which the connections between nodes is from one layer to the next and do not form a cycle. See [119] for further details 32

**fine-tuning** (fine-tuning) Refers to the process of adapting a pre-trained model to perform specific tasks or to specialize in a particular domain. This phase follows the initial pre-training phase and involves training the model further on task-specific data. This is often a supervised learning task 37

**formal methods** (formal methods) Formal methods are mathematically rigorous techniques for the specification, development, and verification of software systems 20

**Functional Attacks** (Functional Attacks) Adversarial attacks that are optimized for a set of data in a domain rather than per data point 16, 25

**G**

generative (generative) Type of machine learning methods which learn the data distribution and can generate new examples from distribution 8

generative adversarial networks (generative adversarial networks) A generative adversarial network (GAN) is a class of machine learning frameworks in which two neural networks contest with each other in the form of a zero-sum game, where one agent’s gain is another agent’s loss. GAN’s learn to generate new data with the same statistics as the training set. See [119] for further details 32, 35

**Generative Pre-Trained Transformer** (Generative Pre-Trained Transformer) An artificial neural network based on the transformer architecture [304], pre-trained on large data sets of unlabelled text, and able to generate novel human-like content. Today, this is the predominant architecture for natural language processing tasks 35

graph neural networks (graph neural networks) A Graph Neural Network (GNN) is an optimizable transformation on all attributes of the graph (nodes, edges, global-context) that preserves the graph symmetries (permutation invariances). GNNs utilize a “graph-in, graph-out” architecture that takes an input graph with information loaded into its nodes, edges and global-context, and progressively transform these embeddings into an output graph with the same connectivity as that of the input graph 32

**H**

hidden Markov models (hidden Markov models) A hidden Markov model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobservable states. In addition, the model provides an observable process whose outcomes are “influenced” by the outcomes of Markov
model in a known way. HMM can be used to describe the evolution of observable events that depend on internal factors, which are not directly observable. In machine learning it is assumed that the internal state of a model is hidden but not the hyperparameters.

I

**indirect prompt injection** (indirect prompt injection) Attacker technique in which a hacker relies on an LLM ingesting a prompt injection attack indirectly, e.g., by visiting a web page or document. Unlike its direct prompt injection sibling, the attacker in this scenario does not directly supply a prompt, but attempts to inject instructions indirectly by having the text ingested by some other mechanism, e.g., a plugin.

**integrity attack** (integrity attack) Adversarial attacks against machine learning which change the output prediction of the machine learning model.

J

**jailbreak** (jailbreak) An attack that employs prompt injection to specifically circumvent the safety and moderation features placed on LLMs by their creators.

L

**label flipping** (label flipping) a type of data poisoning attack where the adversary is restricted to changing the training labels.

**label limit** (label limit) Capability in which the attacker in some scenarios does not control the labels of training samples in supervised learning.

**logistic regression** (logistic regression) Type of linear classifier that predicts the probability of an observation to be part of a class.

M

**machine unlearning** (machine unlearning) Technique that enables a user to request removal of their records from a trained ML model. Efficient approximate unlearning techniques do not require retraining the ML model from scratch.

**membership-inference attacks** (membership-inference attacks) Data privacy attacks to determine if a data sample was part of the training set of a machine learning model.

**model control** (model control) Capability in which the attacker has control over machine learning model parameters.

**model extraction** (model extraction) Type of privacy attack to extract model architecture and parameters.

**model poisoning** (model poisoning) Poisoning attacks in which the model parameters are under the control of the adversary.
**model privacy** (model privacy) Attacks against machine learning models to extract sensitive information about the model 10

**multimodal models** (multimodal models) Modality is associated with the sensory modalities which represent primary human channels of communication and sensation, such as vision or touch. Multimodal models process and relate information from multiple modalities 55

**out-of-distribution** (out-of-distribution) This term refers to data that was collected at a different time, and possibly under different conditions or in a different environment, than the data collected to train the model 51

**poisoning attacks** (poisoning attacks) Adversarial attacks against machine learning at training time 8, 37

**pre-training** (pre-training) Refers to the initial phase of model training where the model learns general patterns, features, and relationships from vast amounts of unlabeled data. This is typically unsupervised or self-supervised learning, and aims to equip the model with commonly occurring patterns prior to a fine-tuning stage that specializes the model for a specific downstream task. Foundation models (text or images) are pre-trained models 37

**prompt extraction** (prompt extraction) An attack in which the objective is to divulge the system prompt or other information in an LLMs context that would nominally be hidden from a user 37, 39

**prompt injection** (prompt injection) Attacker technique in which a hacker enters a text prompt into an LLM or chatbot designed to enable the user to perform unintended or unauthorized actions 39

**prompt injections** (prompt injections) Malicious plain text instructions to a generative AI system that uses textual instructions (a “prompt”) to accomplish a task causing the AI system to generate text on a topic prohibited by the designers of the system 51

**property inference** (property inference) Data privacy attacks which infer global property about the training data of a machine learning model 10

**query access** (query access) Capability in which the attacker can issue queries to a trained machine learning model and obtain predictions 10, 39

**Receiver Operating Characteristics (ROC)** (Receiver Operating Characteristics (ROC)) In ML the Receiver Operating Characteristics (ROC) curve plots true positive rate versus false positive rate for a classifier 92
Red Teaming (Red Teaming) NIST defines cybersecurity red-teaming as “A group of people authorized and organized to emulate a potential adversary’s attack or exploitation capabilities against an enterprise’s security posture. The Red Team’s objective is to improve enterprise cybersecurity by demonstrating the impacts of successful attacks and by demonstrating what works for the defenders (i.e., the Blue Team) in an operational environment.” (CNSS 2015 [80]) Traditional red-teaming might combine physical and cyber attack elements, attack multiple systems, and aims to evaluate the overall security posture of an organization. Penetration testing (pen testing), in contrast, tests the security of a specific application or system. In AI discourse, red-teaming has come to mean something closer to pen testing, where the model may be rapidly or continuously tested by a set of evaluators and under conditions other than normal operation 51

regression (regression) Type of supervised ML model that is trained on data including numerical labels (called response variables). Types of regression algorithms include linear regression, polynomial regression, and various non-linear regression methods 8

reinforcement learning (reinforcement learning) Type of machine learning in which an agent interacts with the environment and learns to take actions which optimize a reward function 8

resource control (resource control) Capability in which the attacker has control over the resources consumed by a ML model, particularly for LLMs and RAG applications 39, 44

Retrieval Augmented Generation (Retrieval Augmented Generation) This term refers to retrieving data from outside a foundation model and augmenting prompts by adding the relevant retrieved data in context. RAG allows fine-tuning and modification of the internal knowledge of the model in an efficient manner and without needing to retrain the entire model. First, the documents and user prompts are converted into a compatible format to perform relevancy search. Typically this is accomplished by converting the document collection and user prompts into numerical representations using embedding language models. RAG model architectures compare the embeddings of user prompts within the vector of the knowledge library. The original user prompt is then appended with relevant context from similar documents within the knowledge library. This augmented prompt is then sent to the foundation model. For RAG to work well, the augmented prompt must fit into the context window of the model 3, 36, 37, 39, 44

rowhammer attacks (rowhammer attacks) Rowhammer is a software-based fault-injection attack that exploits DRAM disturbance errors via user-space applications and allows the attacker to infer information about certain victim secrets stored in memory cells. Mounting this attack requires attacker’s control of a user-space unprivileged process that runs on the same machine as the victim’s ML model 32
semi-supervised learning (semi-supervised learning) Type of machine learning in which a small number of training samples are labeled, while the majority are unlabeled.

shadow models (shadow models) Shadow models imitate the behavior of the target model. The training datasets and thus the ground truth about membership in these datasets are known for these models. Typically, the attack model is trained on the labeled inputs and outputs of the shadow models.

side channel (side channel) side channels allow an attacker to infer information about a secret by observing nonfunctional characteristics of a program, such as execution time or memory or by measuring or exploiting indirect coincidental effects of the system or its hardware, like power consumption variation, electromagnetic emissions, while the program is executing. Most commonly, such attacks aim to exfiltrate sensitive information, including cryptographic keys.

source code control (source code control) Capability in which the attacker has control over the source code of the machine learning algorithm.

supervised learning (supervised learning) Type of machine learning methods based on labeled data.

Support Vector Machines (Support Vector Machines) A Support Vector Machine implements a decision function in the form of a hyperplane that serves to separate (i.e., classify) observations belonging to one class from another based on patterns of information about those observations (i.e., features).

targeted poisoning attacks (targeted poisoning attacks) Poisoning attacks against machine learning which change the prediction on a small number of targeted samples.

testing data control (testing data control) Capability in which the attacker has control over the testing data input to the machine learning model.

training data control (training data control) Capability in which the attacker has control over a part of the training data of a machine learning model.

training stage (training stage) Stage of machine learning pipeline in which the model is trained using training data.

trojans (trojans) A malicious code/logic inserted into the code of a software or hardware system, typically without the knowledge and consent of the organization that owns/develops the system, that is difficult to detect and may appear harmless, but can alter the intended function of the system upon a signal from an attacker to cause a malicious behavior desired by the attacker. For Trojan attacks to be effective, the trigger must be rare in the normal operating environment so that it does not affect the normal effectiveness of the AI and raise the suspicions of human users.

unsupervised learning (unsupervised learning) Type of machine learning methods based on unlabeled data.
watermarking (watermarking) a technique which embeds a hidden signal in a piece of text, image, or video to identify it as AI-generated. The signal can later be retrieved even if the content has been modified. Generally, the watermark must be accessible and reliable. The watermark is accessible if it is possible to test whether the content is AI-generated. There are two types of accessibility: public and private. Public access means everyone can access and verify it. Private access means that only the parties authorized by the organization controlling the generating model have the ability to access and verify the watermark. The watermark must be reliable in the sense that malicious actors should not be able to remove the watermark easily. However, in cases of public access to the watermark the attacker may be allowed to test for the presence of watermarks, which increases the technical challenges with watermarking.