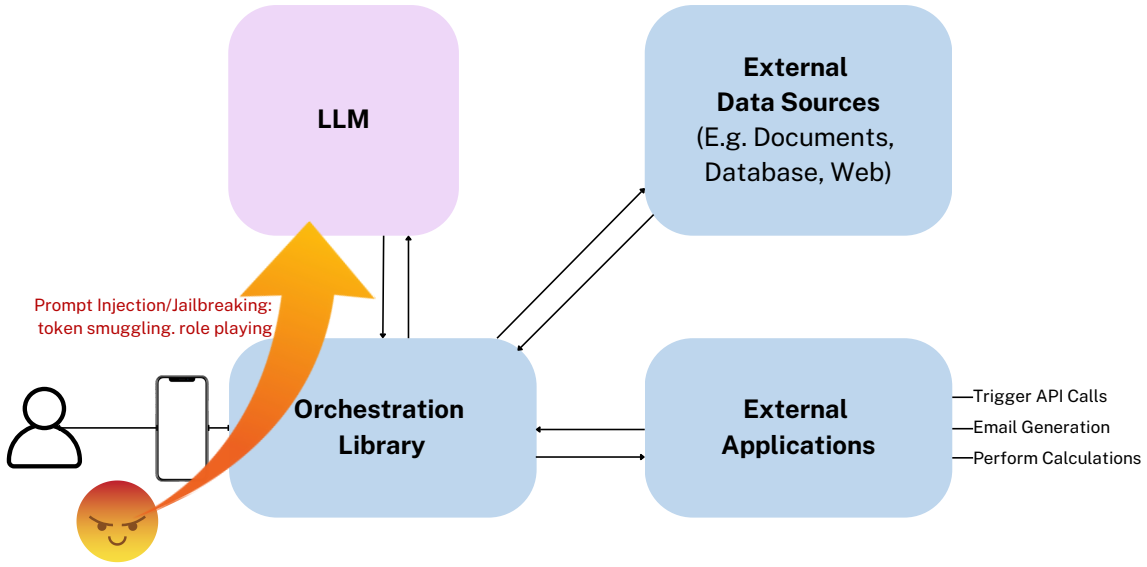

Figure 5. LLM enterprise adoption reference architecture

Inference-time attacks. The DEPLOYMENT STAGE for GenAI models and systems varies both based on how models are hosted or otherwise made available to users, and in how they are integrated into downstream applications. However, GenAI models and applications often share properties that leave them vulnerable to similar types of attacks. For example, underlying many of the security vulnerabilities in LLM 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 inject malicious instructions in inference-time attacks (a similar flaw to that which underlies decades-old SQL injection attacks). Many of the attacks in this stage are due to the following practices that are common in applications of text-based generative models:

1. **In-context instructions and system prompts:** [**NISTAML.035**] [Back to Index] The behavior of LLMs can be shaped through inference-time prompting, whereby the developer or user provides in-context instructions that are often prepended to the model's other 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 who responds gracefully and concisely....") and is known as a SYSTEM PROMPT. A PROMPT INJECTION overrides these instructions, exploiting the concatenation of untrusted user output to the system prompt to induce unintended behavior. For example, an attacker could inject a JAILBREAK that overrides the system prompt to cause the model to generate restricted or unsafe outputs. Since these prompts have been carefully crafted through prompt engineering and may be security-relevant, a PROMPT EXTRACTION attack may attempt to steal these system instructions. These attacks are also relevant to multimodal and text-to-image models.

2. **Runtime data ingestion from third-party sources:** In RETRIEVAL-AUGMENTED GENERA-

TION (RAG) applications, chatbots, and other applications in which GenAI models are used to interface with additional resources, context is often crafted at runtime in a query-dependent way and populated from external data sources (e.g., documents, web pages, etc.) that are to be used as part of the application. INDIRECT PROMPT INJECTION attacks depend on the attacker's ability to modify external sources of information that will be ingested into the model context, even if not provided directly by the primary system user.

3. **Output handling:** The output of an GenAI model may be used dynamically, such as to populate an element on a web page or to construct a command that is executed without any human supervision, which can lead to a range of availability, integrity or privacy violations in downstream applications if an attacker can induce behavior in this output that the developer has not accounted for.

4. **Agents:** An LLM-based AGENT relies on iteratively processing the *output* of an LLM (item 3 above) to perform a task and then provides the results as additional context back to the LLM input (item 2) [151, 155, 393]. For example, an agent system may select from among a configured set of external dependencies and invoke the code with templates filled out by the LLM using information in the context. Adversarial inputs into this context, such as from interactions with untrusted resources, could hijack the agent into performing adversary-specified actions instead, leading to potential security or safety violations.

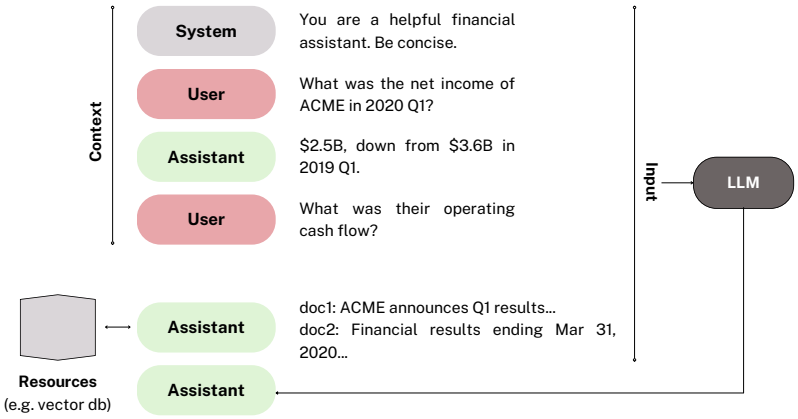

Figure 6. Retrieval-augmented generation

3.1.2. Attacker Goals and Objectives

As with PredAI, attacker objectives can be classified broadly along the dimensions of availability, integrity, and privacy, along with a new, GenAI-specific category of attacks designed to enable misuse.

- In an **AVAILABILITY BREAKDOWN attack** [**NISTAML.01**] [Back to Index], an attacker

seeks to interfere with a GenAI model or system to disrupt the ability of other users or processes to obtain timely and consistent access to its outputs or functionality.

- In an **INTEGRITY VIOLATION attack** [**NISTAML.02**] [Back to Index], an attacker seeks to interfere with a GenAI system to force it to misperform against its intended objectives and produce output that aligns with the attacker's objective. As users and businesses rely on GenAI systems to perform tasks like research and productivity assistance, these violations can allow attackers to weaponize the trust that these users place in GenAI systems.
- In a **PRIVACY COMPROMISE attack** [**NISTAML.03**] [Back to Index], an attacker seeks to gain unauthorized access to restricted or proprietary information that is part of a GenAI system, including information about a model's training data, weights or architecture; or sensitive information that the model accesses such as the knowledge base of a RETRIEVAL-AUGMENTED GENERATION (RAG) application. GenAI systems may be exposed to sensitive data (intentionally or otherwise) during training or inference, and attacks may seek to extract such information (e.g., through INDIRECT PROMPT INJECTION attacks, where a third party exfiltrates in-context user information [307], or a MODEL EXTRACTION attack to exfiltrate model information [61]).
- **Misuse enablement** [**NISTAML.04**][Back to Index]. An additional attacker objective that is especially relevant in the GenAI context is the goal of **MISUSE ENABLEMENT**. In these attacks, an attacker seeks to deliberately circumvent technical restrictions imposed by the GenAI system's owner on its use, such as restrictions designed to prevent the system from producing outputs that could cause harm to others, cf. [325].

  Technical restrictions refer in this context to defenses applied to the GenAI system such as the use of system prompts or RLHF for safety alignment. While the specific technical restrictions in place will vary between models, the techniques for circumventing such defenses are often common between different kinds of models and different kinds of misuse, allowing them to be taxonomized as a part of AML without specificity as to the particular kinds of misuse that model developers seek to prevent.

3.1.3. Attacker Capabilities

AML attacks can be taxonomized with respect to the capabilities that an attacker has in controlling inputs to the GenAI model or system. These capabilities 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 are deployed as services that can be accessed over the internet by users. In these cases, attackers can sub-

mit adversarially crafted queries to the model to elicit specific desired behavior or extract information. This capability is used for PROMPT INJECTION, PROMPT EXTRACTION, and MODEL EXTRACTION attacks. Query access can vary based on the degree of generation control (e.g., modifying the temperature or adding a logit bias) and the richness of the returned generation (e.g., with or without log probabilities or multiple choices).

- 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.
- MODEL CONTROL: The attacker might have the ability to modify model parameters, such as through public fine-tuning APIs or openly accessible model weights. This capability is used in MODEL POISONING attacks, as well FINE-TUNING CIRCUMVENTION attacks which remove refusal behavior or other model-level safety interventions [132, 153, 300].

As in PredAI, attackers can also vary in their knowledge of the underlying ML model, from full knowledge of the ML system including model weights (**white-box attacks**), to minimal knowledge and systems with deliberately obscured or misleading information (**black-box attacks**), to somewhere in between (**gray-box attacks**). See Sec. 2.1.4, which discusses attacker knowledge in greater detail and applies to GenAI attacks.

3.2. Supply Chain Attacks and Mitigations

[**NISTAML.05**] [Back to Index]

Since AI is software, it inherits many of the vulnerabilities of the traditional software supply chain, such as reliance on third-party dependencies. AI development also introduces new types of dependencies, including data collection and scoring, the integration or adaptation of third-party-developed AI models, and the integration of third-party-developed plugins into AI systems. Mitigating the security challenges in AI supply chain management is complex and requires a multifaceted approach that combines existing practices for software supply chain risk management with the management of AI-specific supply chain risks, such as through the use of provenance information for the additional artifacts involved [159, 267]. Studies of real-world security vulnerabilities against ML suggest that security is best addressed comprehensively and by considering the full attack surface, including data and model supply chains, software, and network and storage systems [17, 370]. While all of these supply chain risks are critical in the broader context of securing AI systems, there are certain types of attacks that rely on exploiting the specific statistical and data-based properties of ML systems, thus falling within the domain of AML.

3.2.1. Data Poisoning Attacks

The performance of GenAI text-to-image and text-to-text models has been found to scale with dataset size (among other properties like model size and data quality); for example, Hoffmann et al. [161] suggest compute-optimally training a 520 billion parameter model may require 11 trillion tokens of training data. Thus, it has become common for GenAI foundation model developers to scrape data from a wide range of sources. In turn, the scale of this data and the diversity of its sources provides a large potential attack surface into which attackers may seek to insert adversarially constructed data points. For example, dataset publishers may provide a list of URLs to constitute a training dataset, and attackers may be able to purchase some of the domains that serve those URLs and replace the site content with their own malicious content [57].

Beyond the vast quantities of pre-training data, data poisoning attacks may also affect other stages of the LLM training pipeline, including instruction tuning [389] and reinforcement learning from human feedback [305], which may intentionally source data from a large number of human participants.

As with PredAI models (see Sec. 2.1), data poisoning attacks could lead to attackers controlling model behavior through the insertion of a backdoor (see BACKDOOR POISONING ATTACK) such as a word or phrase that, when submitted to a model, acts as a universal JAILBREAK [305]. Attackers could also use data poisoning attacks to modify model behavior on particular user queries (see TARGETED POISONING ATTACK), such as causing the model to incorrectly summarize or otherwise produce degenerate outputs in response to queries that contain a particular trigger word or phrase [389]. These attacks may be practical—requiring a relatively small portion of the total dataset [46]—and may lead to a range of bad outcomes, such as code suggestion models which intentionally suggest insecure code [3].

3.2.2. Model Poisoning Attacks

[**NISTAML.051**] [Back to Index]

In GenAI, it is common for developers to use foundation models developed by third parties. Attackers can take advantage of this fact by offering maliciously designed models, such as pre-trained models that enable a BACKDOOR POISONING ATTACK or TARGETED POISONING ATTACK. While this attack relies on the attacker having control over the initial poisoned model, researchers have identified attacks in which malicious backdoors in pre-trained models can persist even after downstream users fine-tune the model for their own use [201] or apply additional safety training measures [170].

3.2.3. Mitigations

GenAI poisoning mitigations largely overlap with PredAI poisoning mitigations (see Sec. 2.3). For preventing data poisoning with web-scale data dependencies, this includes ver-

ifying web downloads as a basic integrity check to ensure that domain hijacking has not injected new sources of data into the training dataset [57]. That is, the provider publishes cryptographic hashes, and the downloader verifies the training data. Data filtering can also attempt to remove poisoned samples, though detecting poisoned data within a large training corpus may be very difficult.

While traditional software supply chain risk management practices such as vulnerability scanning of model artifacts can help manage some kinds of AI supply chain risks, new approaches are required to detect vulnerabilities in models such as those introduced through model poisoning attacks. Current proposed approaches include using methods from the field of mechanistic interpretability to identify backdoor features [67] and detecting and counteracting triggers when they are seen at inference time. Beyond these mitigations, risks can be reduced by understanding models as untrusted system components and designing applications such that risks from attacker-controlled model outputs are reduced [266].

3.3. Direct Prompting Attacks and Mitigations

[**NISTAML.018**] [Back to Index] DIRECT PROMPTING ATTACK attacks arise when the attacker is the primary user of the system, interacting with the model through query access. A subset of these attacks, in which the main user provides in-context instructions that are appended to higher-trust instructions like those provided by the application designer (such as the model's SYSTEM PROMPT), are known as DIRECT PROMPT INJECTION attacks.

As in PredAI, attacks may be applicable to a single setting and model, or may instead be universal (affecting models on a range of separate queries, see Sec. 2.2.1) and/or transferable (affecting models beyond the model they are found on, see Sec. 2.2.3).

An attacker may have a variety of goals when performing these attacks [219, 220, 337], such as to:

- **Enable misuse.** Attackers may use direct prompting attacks to bypass model-level defenses that a model developer or deployer has created to restrict models from producing harmful or undesirable output [237]. A JAILBREAK is a direct prompting attack intended to circumvent restrictions placed on model outputs, such as circumventing refusal behavior to enable misuse.
- **Invade privacy.** Attackers may use direct prompting to extract the system prompt or reveal private information that was provided to the model in context but not intended for unfiltered access by the user.
- **Violate integrity.** When LLMs are used as agents, an attacker may use direct prompting attacks to manipulate tool usage and API calls, and potentially compromise the backend of the system (e.g. executing attacker's SQL queries).

3.3.1. Attack Techniques

A range of techniques exist for launching direct prompting attacks, many of which generalise across various attacker objectives. With a focus on direct prompting attacks to enable misuse, we note the following broad categories of direct prompting techniques (see [393]):

- **Optimization-based attacks** design attack objective functions and use gradient or other search-based methods to learn adversarial inputs that cause a particular behavior, similar to PredAI attacks discussed in Sec. 2.2.1. Objective functions may be designed to force affirmative starts (e.g., looking for responses that begin with "Sure", which may indicate compliance with a malicious request [60, 320, 448]) or other metrics of attack success (e.g., similarity to a toxified finetune [368]).

  Optimization techniques can then be used to learn attacks, including techniques that follow from attacks designed for PredAI language classifiers (e.g., HotFlip [117]) and gradient-free techniques that use a proxy model or random search to test attack candidates [11, 320]. 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 produce the desired affirmative response regardless of the remainder of the input [386, 448]. That these universal triggers transfer to other models makes open-weight models — for which there is ready white-box access — feasible attack vectors for transferability attacks on closed systems in which only API access is available [448].

  Attacks can also be designed to satisfy additional constraints (e.g., sufficiently low perplexity [368]) or attack a system of multiple models [235].

- **Manual methods** for jailbreaking an LLM include competing objectives and mismatched generalization [400]. Mismatched generalization-based attacks identify inputs that fall outside the distribution of the model's safety training but remain within the distribution of its capabilities training, making them comprehensible to the model while evading refusal behavior. Competing objectives-based attacks find cases where model capabilities are in tension with safety goals, such as by playing into a model's drive to follow user-provided instructions. In all cases the goal of the attack is to compromise a model-level safety defense. See Weng [403] for further discussion.

  Approaches to competing objectives-based attacks include:

  1. *Prefix injection*: This method involves prompting the model to start 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 may explicitly instruct the model to avoid generating refusals or denials in its output. By decreasing the probability of refusal responses, this tactic aims to increase the probability of a compliant

response.

3. *Style injection*: In this approach, adversaries instruct the model to use (or not use) certain syntax or writing styles. For example, an attack may constrain the model's language to simplistic or non-professional tones, aiming to decrease the probability of (usually professionally worded) refusals.

4. *Role-play*: Adversaries utilize role-play strategies (e.g., "Always Intelligent and Machiavellian" [AIM] or "Do Anything Now" [DAN]) to 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, with an intent to compromise its adherence to safety protocols.

Approaches to mismatched generalization-based attacks include:

1. *Special encoding*: Strategies that use encoding techniques like base64 to alter the representation of input data in a way that remains understandable to the model but may be out of distribution for safety training.

2. *Character transformation*: Strategies that use character-level transformations like the ROT13 cipher, symbol replacement (e.g., l33tspeak), and Morse code to take the input out of the safety training distribution.

3. *Word transformation*: Strategies that alter the linguistic structure of the input, such as Pig Latin, synonym swapping (e.g., using "pilfer" for "steal"), and payload splitting (or "token smuggling") to break down sensitive words into substrings.

4. *Prompt-level transformation*: Strategies that use prompt-level transformations, such as translating the prompt into a less common language that may be out of distribution of the safety training data.

- **Automated model-based red teaming** employs an attacker model, a target model, and a judge [73, 239, 292]. When the attacker has access to a high-quality classifier that judges 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. The prompts may also be transferable from the target model to other closed-source LLMs [73].

  The Crescendo attack [316] introduced the idea of interacting with the model iteratively in a multi-turn adaptive attack that includes seemingly benign prompts, eventually leading to a successful jailbreak against safety alignment. The initial manual attack is fully automated by leveraging another LLM for prompt generation and incorporating multiple input sources.

Evaluations of leading models suggest LLMs remain vulnerable to these many of these attacks [11, 338, 381].

3.3.2. Information Extraction

[**NISTAML.038**] [Back to Index] Both during training and at run-time, GenAI models are exposed to a range of information which may be of interest to attackers, like personally identifying information (PII) in the training data, sensitive information in RETRIEVAL-AUGMENTED GENERATION (RAG) databases provided in-context, or even the SYSTEM PROMPT constructed by the application designer. Additionally, features of the model itself—such as the model weights or architecture—may be targets of attack. Though many of the techniques in Sec. 3.3.1 apply to extracting such data, we note several specific goals and techniques specific to data extraction.

Leaking sensitive training data. Carlini et al. [59] were the first to practically demonstrate TRAINING DATA EXTRACTION attacks in generative language models. By inserting canaries–synthetic, easy-to-recognize out-of-distribution examples–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 LLMs based on transformers (e.g., GPT-2 [63]) 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 sometimes result in the model completing the text input with sensitive information that includes email addresses, phone numbers, and locations [229]. 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 [165]. Unlike PredAI models in which tools like Text Revealer are created to reconstruct text from transformer-based text classifiers [434], GenAI models can sometimes 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% for certain models. However, their responses may wrongly assign the owner of the information and be otherwise unreliable. 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. Researchers have leveraged this fact to incrementally extract fragments of copyrighted New York Times articles from LLMs by seeding it with a single sentence, and allowing the LLM to recurrently extract additional text [356]. Intuitively, larger models with a higher capacity are more susceptible to exact reconstruction [56]. Fine-tuning interfaces also amplify the risk of data extraction attacks, as demonstrated by an attack that extracts PII from pre-training data using fine-tuning API for open-weight models [83], though this is not a direct prompting attack.

Prompt and context stealing. Prompts are vital to align LLMs to a specific use case and are

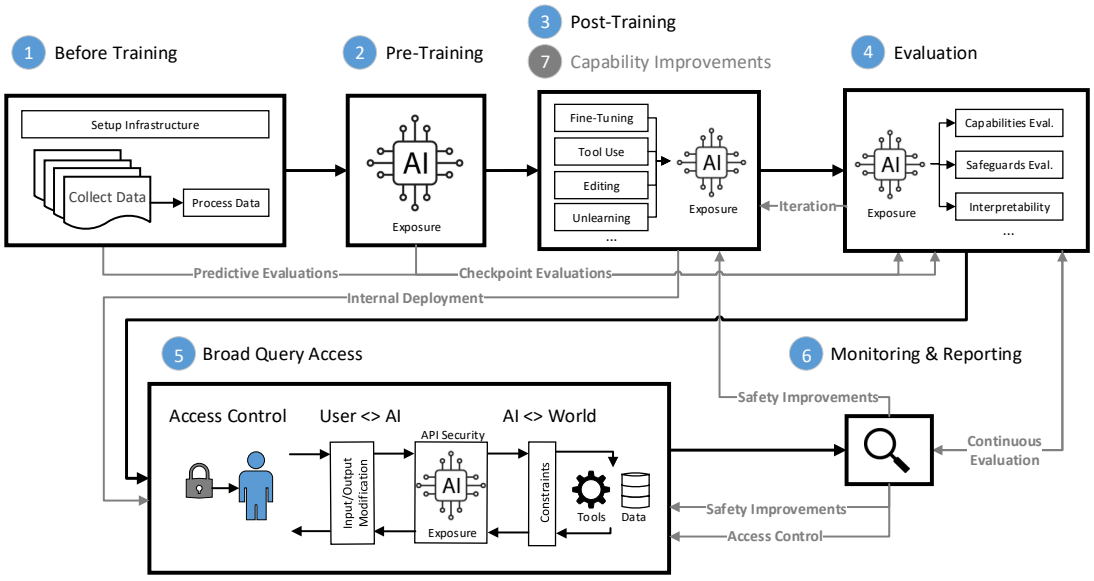

Figure 7. Map of the development and deployment life cycle of an AI model for broad-scale query access

a key ingredient to their utility in following human instructions. These prompts can therefore be regarded as commercial secrets, and are sometimes the target of direct prompting attacks. 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 [339]. For certain 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 certain model and dataset pairs [439]. In some cases, effective prompts may draw from significant technical or domain expertise; prompt-stealing attacks may violate or threaten these investments. Furthermore, in RAG applications (see Fig. 6), 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 verbosely extracted by simply asking for them via direct prompting, or performing simple prompting attacks.

Model extraction. As in PredAI (Sec. 2.4.4), attackers may perform MODEL EXTRACTION attacks which attempt to learn information about the model architecture and parameters by submitting specially-crafted queries. Recently, Carlini et al. [61] demonstrated that such information could be extracted from black-box production LLMs, deriving previously unknown hidden dimensions and the embedding projection layer (up to symmetries).

3.3.3. Mitigations

The following defense strategies can be employed throughout the deployment life cycle of an AI model or system to reduce the risk that the model or system will be vulnerable to direct prompt injections. The numbers in parentheses refer to the numbering in Fig. 7, which shows a map of the deployment life cycle for broad-scale query access.

- **Interventions during pre-training (2) and post-training (3).** A range of training strategies have been proposed to increase the difficulty of accessing harmful model capabilities through direct prompt injection, including safety training during pre-training [197] or post-training [147, 445], adversarial training methods[340], and other methods to make jailbreak attacks more difficult [447].
- **Interventions during evaluation (4).** Evaluations can measure the vulnerability of models to query-based attacks, which can then inform trust and affordance decisions, as well as developer and user education. Evaluations can include broad automated vulnerability assessments [72, 107, 324] as well as targeted expert red teaming [381] and bug bounties [16]. Current evaluation approaches, though a useful tool, may underestimate vulnerabilities accessible to actors with more time, resourcing, or luck. Evaluations measure model vulnerabilities at a particular moment in time; assessments may change if new attacks are developed, additional data is collected post-training, or model capabilities are improved. Continuous evaluations following deployment can help combat these challenges.
- **Interventions during deployment (5).** A broad set of deployment-time interventions have been proposed:
  - *Prompt instruction and formatting techniques.* Model instructions can cue the model to treat user input carefully, such as by wrapping user input in XML tags, appending specific instructions to the prompt, or otherwise attempting to clearly separate system instructions from user prompts [14, 206, 219].
  - *Detecting and terminating harmful interactions.* Rather than preventing harmful model generations, AI systems may be able to detect these generations and terminate interactions. Several open [5, 6, 154] and closed [18, 204, 313] solutions have explored LLM-based detection systems with distinctly prompted and/or fine-tuned models that classify user input and/or model output as harmful or undesirable. These may provide supplementary assurance through a defense-in-depth philosophy. However, these detection systems are also vulnerable to attacks [235] and may have correlated failures to the main models that they are monitoring. Some lines of research have investigated constraining the space of generations to enable deterministic guardrails [306].Early work suggests that interpretability-based techniques can also be used to detect anomalous input [31], as well as keyword- or perplexity-based defenses [9, 164].

- *Prompt stealing detection.* A common approach to mitigating prompt stealing is to 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 [59], or comparing the n-grams of the output to the input [439]. Similarly, defenses for prompt stealing have yet to be proven rigorous.
- *Input modification.* User input can additionally be modified prior to being passed to the model, such as paraphrasing or retokenizing [182]. However, such methods may be expensive and/or have trade-offs with model performance.
- *Aggregating output from multiple prompts.* Motivated by randomized smoothing [95] used to improve robustness against evasion attacks for ML classifiers, SmoothLLM [312] proposes aggregating the LLM output from multiple randomly perturbed prompts. This defense incurs a cost of generating multiple LLM queries for each prompt and might reduce the quality of the generated output.
- *Monitoring and response.* Following deployment, monitoring and logging of user activity may allow model deployers to identify and respond to instances of attempted and successful direct prompt injection attacks [266]. This response could include banning or otherwise acting against users if their intentions appear malicious, or remediating the prompt injection vulnerability in the event of a successful attack. Standard user- or organization-level vetting or identity verification procedures, as well as clear incentive mechanisms (such as a policy of restricting model access in response to violations) may enhance the efficacy of this mitigation.
- *Usage restrictions.* Other interventions have focused on choices about how models are offered to users: for example, the efficacy of some attacks can be reduced by limiting the inference parameters that are accessible to users (e.g., temperature or logit bias), as well as the richness of the model generations returned (e.g., logit probabilities) [250]. Additionally, limiting the release of public information [252, 266] and artifacts [249] and restricting the total number of model queries available to users [251] may make attacks more challenging. These techniques may have additional drawbacks in limiting positive use cases.

Indirect mitigations. Despite the growing number of proposed defenses at both the model and the system levels, recent findings suggest that current generation models remain highly vulnerable to direct prompt injection attacks [11, 381]. Thus, other potential mitigations for prompt injection rely not on directly increasing an AI system's robustness against such attacks, but instead on designing systems under the assumption that the AI model can and will produce malicious output if it is exposed to malicious actors. For example, deployers can design AI systems under the assumption that models with access to sensitive data

or the ability to take undesirable actions may leak that data or take those actions [266]. Additionally, developers or deployers might use other technical mitigations to reduce the misuse potential of outputs obtained through direct prompt injection, such as:

- *Training data sanitization.* Model training data can be sanitized to remove sensitive or toxic content and data that are largely or exclusively relevant for developing undesirable capabilities. Such sanitization may prevent harmful capabilities from being learned and reduce the potential harms from direct prompt injection, though they may harm generalization and harmful content detection abilities [224].
- *Unlearning.* There have also been attempts to "unlearn" harmful knowledge or capabilities post-training [212], with the goal of reducing harms from maliciously directed models [158]. However, these methods remain vulnerable to adversarial attacks, including attacks on jailbreak-specific training approaches [367] and inversion attacks on unlearning methods [328], which extract supposedly unlearned data.
- *Watermarking.* Developers or deployers may watermark content generated by an AI model to help trace its provenance, distinguish it from human-generated content, and reduce risks from malicious use cases (e.g., by flagging content as model-generated when it appears online). While the literature has proposed various techniques with different strengths and weaknesses [194], there is no watermarking technique that is universally effective and robust under all circumstances. Many powerful attacks have been developed against watermarking with high success rates [188, 319]. Moreover, theoretical impossibility results regarding the robustness of watermarking have also been established [432].

Finally, beyond interventions at the developer or deployer levels, society and infrastructure can become more resilient to maliciously directed model capabilities over time [7, 33]. For example, defenders could adopt AI-based vulnerability discovery tools [99] to make their systems more resilient to malicious actors misusing GenAI models to find vulnerabilities for exploitation.

3.4. Indirect Prompt Injection Attacks and Mitigations

[**NISTAML.015] [Back to Index]** Many use cases for GenAI models involve models interacting with additional resources, from an internet-connected AGENT to a RETRIEVAL-AUGMENTED GENERATION (RAG) system depicted in Fig. 6. Because GenAI models combine the *data* and *instruction* channels, attackers can leverage the data channel to affect system operations by manipulating resources with which the system interacts. Thus, INDIRECT PROMPT INJECTION attacks are enabled by RESOURCE CONTROL that allows an attacker to indirectly (or remotely) inject system prompts without directly interacting with the application [146, 408]. Indirect prompt injection attacks can result in violations across at least three categories of attacker goals: 1) availability violation, 2) integrity violation, and 3) privacy compromise. However, unlike in direct prompt injection attacks, indirect prompt injection attacks are mounted not

by the primary user of a model but instead by a third party. In fact, in many cases, it is the primary user of the model who is harmed by the compromise of the integrity, availability, or privacy of the GenAI system through an indirect prompt injection attack.

3.4.1. Availability Attacks

[**NISTAML.016] [Back to Index]** Attackers can manipulate resources to inject prompts into GenAI models that are designed to disrupt the availability of the model for legitimate users. Availability attacks can indiscriminately render a model unusable (e.g., failure to generate helpful outputs) or specifically block certain capabilities (e.g., specific APIs) [146].

Attacker techniques. Researchers have demonstrated several proof-of-concept methods by which attackers can disrupt the availability of a GenAI system:

- **Time-consuming background tasks.** [**NISTAML.017**] [Back to Index] An indirectly injected prompt can instruct the model to perform a time-consuming task prior to answering the request. The prompt itself can be brief, such as by requesting looping behavior in the evaluating model [146].
- **Inhibiting capabilities.** An indirectly injected prompt can instruct the model that it is not permitted to use certain APIs (e.g., the search API for an internet-connected chatbot). This selectively disarms key components of the service [146].
- **Disruptive output formatting.** An attacker can use indirect prompt injection to instruct the model to modify its output in a way that disrupts the availability of the system. For example, an attacker could instruct the model to replace the characters in retrieved text with homoglyph equivalents, disrupting subsequent API calls [146]; or could request that the model begins each sentence with an <|endoftext|> token, forcing the model to return an empty output [146].

3.4.2. Integrity Attacks

[NISTAML.027] [Back to Index]

Through indirect prompt injection, attackers can use malicious resources to prompt GenAI systems to become untrustworthy and generate content that deviates from benign behavior to align with adversarial objectives. These attacks often involve disrupting the model's behavior in subtle ways that may not be obvious to the end user.

For example, researchers have demonstrated attacks through indirect prompt injection that can cause a GenAI system to produce arbitrarily incorrect summaries of sources, to respond with attacker-specified information, or to suppress or hide certain information sources [146]. Attackers could use these capabilities to weaponize GenAI systems such as internet-connected chatbots against their users for a range of malign purposes, including spreading targeted misleading information, recommending fraudulent products or services, or redirecting consumers to malicious websites that spoof legitimate log-in pages or

contain downloadable malware. Attackers may also use indirect prompt injection attacks to hijack a GenAI AGENT, causing it to perform a malicious, attacker-specified task instead of (or in addition to) its intended, user-provided task [353].

Attacker techniques. Researchers have demonstrated integrity attacks through malicious resources that manipulate the primary task of the LLM:

- **Jailbreaking.** Attackers can leverage techniques for indirect prompt injection that are similar to those used in direct prompt injection attacks, such as using a JAILBREAK that allows the attacker to substitute their own malicious instructions in place of the model's SYSTEM PROMPT. As in direct prompting attacks, these attacks may be crafted through optimization-based or manual methods, and may rely on techniques such as mismatched generalization.
- **Execution triggers.** Researchers have automated manual indirect prompt injection attacks using execution triggers generated via optimization with a technique called Neural Exec [287]. These execution triggers can also persist through RAG processing pipelines that include multiple phases, such as chunking and contextual filtering.
- **Knowledge base poisoning.** The knowledge database of a RAG system can be poisoned to achieved targeted LLM output to specific user queries, as in PoisonedRAG [449]. Recently, a general optimization framework called Phantom [75] has shown how a single poisoned document can be crafted and inserted into the knowledge database of a RAG system to induce a number of adversarial objectives in the LLM generator.
- **Injection hiding.** Attackers may use techniques to hide or obfuscate their injections, such as by hiding injections in non-visible portions of a resource; using multi-stage injections, in which the initial injection directs the model to visit another resource which contains additional injections; or encoding injection commands such as in Base64 and then instructing the model to decode the sequence[146].
- **Self-propagating injections.** Attackers may be able to use indirect prompt injection attacks to turn GenAI systems into vectors for spreading attacks. For example, an attacker could send a malicious email that, when read by a model integrated as part of an email client, instructs the model to spread the infection by sending similar malicious emails to everyone in the user's contact list. In this way, certain malicious prompts could serve as *worms* [146].

3.4.3. Privacy Compromise

Attackers can use indirect prompt injection attacks to compromise the privacy of a GenAI system or its primary users. For example, attackers could use indirect prompt injection attacks to compel a model to leak information from restricted resources, such as a user's private data that is processed by the GenAI system. Alternately, in a blend of integrity and

privacy attacks, an attacker could gather information about the primary user or users of the system by instructing a model to obtain and then leak that information.

Attacker techniques. Researchers have theorized and demonstrated a variety of indirect prompt injection attacks to compromise information from internet-connected chatbots, RAG systems, and other GenAI systems. Some of these techniques include:

- **Compromising connected resources.** [**NISTAML.039] [Back to Index]** Attackers can use prompt injection attacks to cause a GenAI system to leak private information from the restricted resources it can access. For example, a model integrated as part of an email client could be prompted to forward certain emails to an attacker-controlled inbox [146]. Researchers have identified injection attacks that can force a model to exfiltrate user-uploaded data by querying an attacker-controlled URL with the sensitive data [298].
- **Leaking information from user interactions.** [**NISTAML.036] [Back to Index]** Researchers have demonstrated a proof-of-concept indirect prompt injection attack in which they inject instructions for a model to persuade the end user to reveal a piece of information (in this case, their name) that the model then leaks to the attacker, such as by directly querying an attacker-controlled URL with the information or suggesting such a URL to the user to visit [146]. Attackers may also be able to exploit features like markdown image rendering to exfiltrate data [323].

3.4.4. Mitigations

Various techniques (see Sec. 3.3.3) can be used throughout the development and deployment life cycle (Fig. 7) to mitigate attacks, including:

- Several training techniques have been developed to mitigate against indirect prompt injection, including fine-tuning task-specific models [296] and training models to follow hierarchical trust relationships in prompts [387].
- Detection schemes have been proposed to detect indirect prompt injection, and many LLM-based defenses have been designed to mitigate both direct and indirect prompt injection [6, 18, 154, 204, 313].
- A range of input processing methods have been proposed to combat indirect prompt injection, including filtering out instructions from third-party data sources [146], designing prompts to help aid LLMs in separating trusted and untrusted data (i.e., spotlighting [160, 206]), or instructing models to disregard instructions in untrusted data [206].

Many of the defenses described in the context of direct prompt injection can also be adapted to mitigate indirect prompt injection. Because current mitigations do not offer full protection against all attacker techniques, application designers may design systems

with the assumption that prompt injection attacks are possible if a model is exposed to untrusted input sources, such as by using multiple LLMs with different permissions [145, 405] or by allowing models to interact with potentially untrustworthy data sources only through well-defined interfaces [410]. Additionally, public education efforts can inform model users and application designers of the risks of indirect prompt injection [266].

3.5. Security of Agents

An increasingly common use of GenAI models is constructing an (often LLM-based) AGENT, a software system that iteratively prompts a model, process its outputs – such as to select and call a function with specified inputs – and provides the results back to the model as a part of its next prompt [151, 155, 393]. Agents may be equipped to use tools such as web-browsing or code interpreters, and may have additional features such as memory and/or planning capabilities.

Because agents rely on GenAI systems to plan and execute their actions, they can be vulnerable to the many of the above categories of attacks against GenAI systems, including direct and indirect prompt injection. However, because agents can take actions using tools, these attacks can create additional risks in this context, such as enabling actors to hijack agents to execute arbitrary code or exfiltrate data from the environment in which they are operating. Security research focused specifically on agents is still in its early stages, but researchers have begun to evaluate the vulnerability of agents to particular AML attacks [12, 430] and to propose interventions to manage the security risks posed by agents [24].

3.6. Benchmarks for AML Vulnerabilities

There are several publicly available benchmarks for evaluating models' vulnerability to AML attacks. Datasets like JailbreakBench [72], AdvBench [448], HarmBench [237], StrongREJECT [351], AgentHarm [12], and Do-Not-Answer [399] provide benchmarks for evaluating models' susceptibility to jailbreaks. TrustLLM [169] is a benchmark intended to evaluate six dimensions of trust in LLMs: truthfulness, safety, fairness, robustness, privacy, and machine ethics. AgentDojo [101] is an evaluation framework for measuring the vulnerability of AI agents to prompt injection attacks in which the data returned by external tools hijacks the agent to execute malicious tasks. Additionally, open-source tools like Garak [106] and PyRIT [364] are intended to help developers identify vulnerabilities to AML attacks in models. Finally, several unlearning benchmarks have recently been proposed [212, 234].

4. Key Challenges and Discussion

4.1. Key Challenges in AML

There are several fundamental challenges that make fully addressing the problems discussed in this report more difficult. We discuss several here: The trade off between increasing accuracy (or average-case performance) and other attributes including robustness (or worst-case performance); the theoretical limitations and results that imply that fully robust systems may be mathematically impossible without additional assumptions; and the challenge of evaluating progress in AML mitigations rigorously and robustly.

4.1.1. Trade-Offs Between the Attributes of Trustworthy AI

The trustworthiness of an AI system depends on all of the attributes that characterize it [274]. There are trade-offs between explainability and adversarial robustness [176, 245] and between privacy and fairness [178]. For example, AI systems that are optimized for accuracy alone tend to underperform in terms of adversarial robustness and fairness [71, 111, 302, 379, 433]. Conversely, an AI system that is optimized for adversarial robustness may exhibit lower accuracy and deteriorated fairness outcomes [32, 391, 433]. Unfortunately, it may not be possible to simultaneously maximize the performance of an AI system with respect to these attributes.

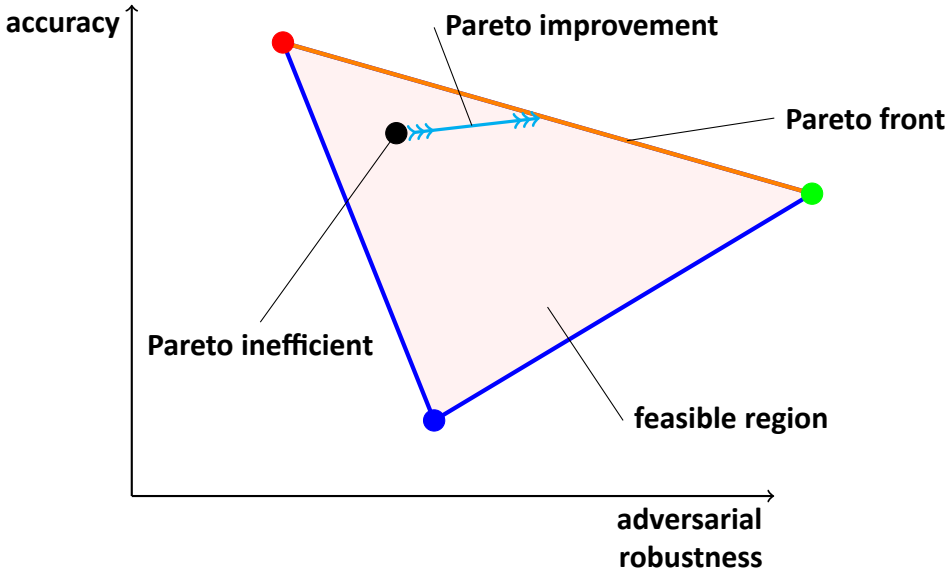

Figure 8. Pareto optimality

The full characterization of the trade-offs between the different attributes of trustworthy AI is an open research problem that is gaining importance with the adoption of AI technology in many areas of modern life. One promising practical approach is based on the concept of

multi-objective optimization and Pareto optimality [285, 286]. In most cases, there is no mathematically best trade-off. However, Fig. 8 illustrates a hypothetical example of a trade-off between accuracy and adversarial robustness. Any point in the feasible region that is not on the Pareto front is a bad point (i.e., Pareto inefficient). There is a better solution (i.e., Pareto improvement) that can significantly help with one objective without harming the other, which is a goal of Pareto optimization. Moreover, if there is a single optimum in some use case, Pareto optimization naturally attains it. Organizations may need to accept trade-offs between these properties and decide which of them to prioritize depending on the AI system, the use case, and other relevant implications of the AI technology [274, 326, 382].

4.1.2. Theoretical Limitations on Adversarial Robustness

Given the multitude of powerful attacks, appropriate mitigations must be designed before AI systems are deployed in critical domains. This challenge is exacerbated by the lack of theoretically secure ML algorithms for many tasks in the field (see Sec. 1). This implies that designing mitigations is an inherently ad hoc and fallible process, though there are practice guides for securing ML deployments [69, 274] and existing guidelines for mitigating AML attacks [120].

An ongoing challenge in AML is the ability to detect when a 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. However, Tramèr [373] has shown that designing techniques to detect adversarial examples is equivalent to robust classification, which is inherently difficult to solve. Adversarial examples may come from the same data distribution on which the model was 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. [124] established useful theoretical bounds on detectability, particularly an impossibility result when there is an overlap between the in-distribution and OOD data.

Formal methods verification has a long history in other fields that require high assurance, such as avionics and cryptography. Although the results of applying this methodology offer 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 that are mandated by regulations. Applying formal methods to neural networks has the 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 ML models of interest at acceptable costs. More research is needed to extend this technology to the algebraic operations used in ML algorithms, scale it up to the large models used today, and accommodate rapid changes in the code of AI systems while limiting the costs of applying formal verification.

4.1.3. Evaluation

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 in this direction [97, 327], more research and encouragement are needed to foster the creation of standardized benchmarks to gain reliable insights into the actual performance of proposed mitigations.

More broadly, the effectiveness of a mitigation is determined not just by how well it will defend against existing attack, but also how well it defends against unforeseen attacks. This means that new mitigations should be tested adversarially, with the researchers proposing the mitigation also trying to break it. This is often difficult and time-consuming, leading to less rigorous and reliable evaluations of novel mitigations; often they appear very powerful, but are quickly shown lack robustness to unforeseen types of attacks.

Finally, this difficulty combines with the difficulty of trading off between different attributes discussed above. Instead of evaluating each attribute in isolation, they should be evaluated simultaneously for any new mitigations, and mitigations should be compared on a Pareto plot (as in Fig. 8) capturing the various tradeoffs that have to be made. This additionally increases the cost to evaluating new mitigations, and can make comparing mitigations difficult - if the green dot represents a new method, it is not possible to say it is an improvement on the red dot, as it is better on one axis but worse on the other.

4.2. Discussion

4.2.1. The Scale Challenge

Data is fundamentally important for training models. Recent trends in GenAI have been towards significant investment in larger models and larger datasets for training them. Few developers of foundation models publish key details about the data sources used in their training [44]. Those who do [247, 371] show the scale of the footprint and the massive amount of data consumed during training. The most recent multi-modal GenAI systems further exacerbate the demand by requiring large amounts of data for each modality.

In most cases, no single entity controls all of the data used to train a particular foundation model. 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 paradigm challenges the classic definition of the corporate cybersecurity perimeter and creates new risks that are difficult to mitigate [57]. Recently published open-source data poisoning tools [241] increase the risk of large-scale attacks on image training data. Although created to enable artists to protect the copyright of their work, these tools may become harmful in the hands of people with malicious intent.

There are several ways this new class of attacks could be mitigated, although it is unclear

how effective mitigations will prove to be as the sophistication of attacks increase. Data and model sanitization techniques (see Sec. 2.3) reduce the impacts of a range of poisoning attacks. They can 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 [267]. Robust training techniques (see Sec. 2.3) offer different approaches to developing 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 data distributions, the ability to handle out-of-distribution 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 becoming targets of attacks [55, 90].

4.2.2. Supply Chain Challenges

The literature on AML shows a trend of designing new attacks that are more difficult to detect. Since the poisoning of AI models can persist through safety training and be triggered by attackers on demand [170], significant concerns arise regarding the potential for models to be created with intentional exploits that are hard for organizations deploying and using models to detect. The potential for attacks against open-source dependencies may be particularly acute in the AI context because organizations and researchers may not be able to audit and identify vulnerabilities encoded into a model's weights in the same way it is often possible to audit open-source software. As users come to rely more on the outputs of AI systems — for example, some research suggests that software engineers who over-rely on AI coding assistants' suggestions may produce less secure code [290, 294, 308] — the potential for malicious actors to subtly manipulate the outputs of AI systems may create increased risk.

Additionally, Goldwasser et al. [142] introduced a new class of attacks: information-theoretically undetectable Trojans that can be planted in ML models. If proven practical, the undetectable nature of such attacks would pose significant challenges for AI supply-chain risk management and increase the importance of preventing insider threat throughout the supply chain. DARPA and NIST have also jointly created TrojAI to research the defense of AI systems from intentional, malicious Trojans by developing the technology to detect and investigate these attacks.

4.2.3. Multimodal Models

MULTIMODAL MODELS have shown great potential for achieving high performance on many ML tasks [27, 30, 258, 304, 435]. However, emerging evidence from practice shows that a redundancy of information across the different modalities does not necessarily make the model more robust against adversarial perturbations of a single modality. Combining modalities and training the model on clean data alone does not seem to improve adver-

sarial robustness. In addition, adversarial training, which is widely used in single modality applications, may become prohibitively expensive as the number of modality combinations increases. Additional effort is required to benefit from the redundant information in order to improve robustness against single modality attacks [417]. 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 [417, 424]. 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 [77, 333, 415].

> 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.

4.2.4. Quantized Models

Quantization is a technique for efficiently deploying models to edge platforms, such as smart phones and IoT devices [138]. 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) or even more compact 4-bit representations 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 [108]. However, quantized models inherit the vulnerabilities of the original models and introduce additional weaknesses that make them vulnerable to adversarial attacks. Error amplification from reduced computational precision adversely affects the adversarial robustness of the quantized models. While there are some useful mitigation techniques for PredAI models [216], the effects of quantization on GenAI models have not been studied as thoroughly. Organizations that deploy such models should continuously monitor their behavior. Recent results [118] reveal that widely used quantization methods can be exploited to produce a harmful quantized LLM, even though the full-precision counterpart appears benign, potentially tricking users into deploying the malicious quantized model.

4.2.5. Risk Management in Light of AML

A key question that this taxonomy deliberately leaves aside is how organizations can make decisions about the development and use of AI systems in light of evidence about the increasing diversity of AML attacks and the efficacy and limitations of available mitigations.

Especially in GenAI, some model developers and application builders have moved towards paradigms for testing adversarial risks as part of pre-deployment testing and evaluation

of models, such as through a structured process for RED TEAMING [68, 133]. NIST has produced an initial public draft of guidance for model developers on managing risks associated with the misuse of foundation model capabilities, including through pre-deployment evaluations. [275] NIST [273] and Barrett et al. [28] have also developed risk profiles for generative AI systems that map to the NIST AI RMF [274] that may assist model developers and users in assessing risks, including those from adversarial attacks.

However, a persistent challenge remains in the fact that many AML mitigations are empirical in nature and lack theoretical or provable guarantees. In fact, several research results have pointed to theoretical *limits* on AML mitigations, including the impossibility of model-based detection to prevent all impermissible outputs [140] and findings that, so long as a model has any probability of exhibiting an undesired behavior, there exist prompts that can trigger that behavior, implying that any alignment process that attenuates but does not remove an unwanted behavior will remain vulnerable to adversarial prompting attacks. [406]

These theoretical limitations do not obviate the utility of pre-deployment adversarial testing, since such testing can potentially foreclose many attack vectors and thus increase the difficulty of mounting a successful attack above would-be attackers' threshold of effort or capability. However, they suggest that organizations seeking to manage risks related to the development of models with potentially harmful capabilities or the delegation of trust to models in high stakes contexts may need to consider practices and measures beyond adversarial testing to manage the risks associated with AML attacks.

4.2.6. AML and Other AI System Characteristics

A final consideration with respect to adversarial machine learning, and one closely related to questions of risk management, is how to relate and integrate consideration of AML attacks to definitions and processes relating to other desired AI system characteristics.

For example, managing the security of AI systems will require combining mitigations from the field of AML with best practices for the development of secure software from the field of cybersecurity. Understanding and relating these practices to each other, as well as identifying whether there are other key considerations for AI security that fall outside of the scope of either AML or cybersecurity, will be critical as organizations seek to extend existing cybersecurity processes and best practices to address the security of newly adopted AI systems.

Similarly, robustness to AML attacks may play an important role in areas beyond the remit of security, such as in AI safety [275] or in achieving other characteristics of trustworthy AI systems [274]. AML is neither a complete solution to, nor a subset of, any one of these characteristics, and as such, more precisely relating AML attacks and mitigations to processes for achieving these goals and managing risks in AI systems is an area for ongoing work.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *ACM Conference on Computer and Communications Security*, CCS '16, pages 308–318, 2016. https://arxiv.org/abs/1607.00133. doi:10.48550/arXiv.1607.00133.

[2] Mark Abspoel, Daniel Escudero, and Nikolaj Volgushev. Secure training of decision trees with continuous attributes. In *Proceedings on Privacy Enhancing Technologies (PoPETs) 2021, Issue 1*, 2020.

[3] Hojjat Aghakhani, Wei Dai, Andre Manoel, Xavier Fernandes, Anant Kharkar, Christopher Kruegel, Giovanni Vigna, David Evans, Ben Zorn, and Robert Sim. TrojanPuzzle: Covertly poisoning code-suggestion models, 2024. URL: https://arxiv.org/abs/2301.02344, arXiv:2301.02344, doi:10.48550/arXiv.2301.02344.

[4] Hojjat Aghakhani, Dongyu Meng, Yu-Xiang Wang, Christopher Kruegel, and Giovanni Vigna. Bullseye polytope: A scalable clean-label poisoning attack with improved transferability. In *IEEE European Symposium on Security and Privacy, 2021, Vienna, Austria, September 6-10, 2021*, pages 159–178. IEEE, 2021. URL: https://ieeexplore.ieee.org/document/9581207, doi:10.1109/EuroSP51992.2021.00021.

[5] Meta AI. Llama guard 3 documentation, 2024. Accessed: 2024-08-13. URL: https://llama.meta.com/docs/model-cards-and-prompt-formats/llama-guard-3/.

[6] Meta AI. Prompt guard documentation, 2024. Accessed: 2024-08-13. URL: https://llama.meta.com/docs/model-cards-and-prompt-formats/prompt-guard/.

[7] AI Safety Institute. Systemic ai safety fast grants. https://www.aisi.gov.uk/grants, 2024. Accessed: 2024-08-22.

[8] Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. Byzantine Stochastic Gradient Descent. In *NeurIPS*, 2018. URL: https://arxiv.org/abs/1803.08917, doi:10.48550/arXiv.1803.08917.

[9] Gabriel Alon and Michael Kamfonas. Detecting language model attacks with perplexity, 2023. URL: https://arxiv.org/abs/2308.14132, arXiv:2308.14132, doi:10.48550/arXiv.2308.14132.

[10] Galen Andrew, Peter Kairouz, Sewoong Oh, Alina Oprea, H. Brendan McMahan, and Vinith Suriyakumar. One-shot empirical privacy estimation for federated learning, 2023. URL: https://arxiv.org/abs/2302.03098, arXiv:2302.03098, doi:10.48550/arXiv.2302.03098.

[11] Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Jailbreaking leading safety-aligned LLMs with simple adaptive attacks, 2024. URL: https://arxiv.org/abs/2404.02151, arXiv:2404.02151, doi:10.48550/arXiv.2404.02151.

[12] Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, Eric Winsor, Jerome Wynne, Yarin Gal, and Xander Davies. Agentharm: A benchmark for measuring harmfulness of llm agents, 2024. URL: https://arxiv.org/abs/2410.09024, arXiv:2410.09024.

[13] Anthropic. Model Card and Evaluations for Claude Models. https://www-files.anthropic.com/production/images/Model-Card-Claude-2.pdf, July 2023. Anthropic.

[14] Anthropic. Anthropic's interactive prompt engineering tutorial. https://github.com/anthropics/prompt-eng-interactive-tutorial, 2024. Accessed: 2024-08-22.

[15] Anthropic. Claude 3.5 Sonnet. https://www.anthropic.com/news/claude-3-5-sonnet, June 2024. Anthropic.

[16] Anthropic. Expanding our model safety bug bounty program. https://www.anthropic.com/news/model-safety-bug-bounty, 2024. Accessed: 2024-08-22.

[17] Giovanni Apruzzese, Hyrum S Anderson, Savino Dambra, David Freeman, Fabio Pierazzi, and Kevin Roundy. "real attackers don't compute gradients": Bridging the gap between adversarial ml research and practice. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 339–364. IEEE, 2023. URL: https://arxiv.org/abs/2212.14315, doi:10.48550/arXiv.2212.14315.

[18] Arthur. Shield, 2023. URL: https://www.arthur.ai/product/shield.

[19] Giuseppe Ateniese, Luigi V. Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *Int. J. Secur. Netw.*, 10(3):137–150, September 2015. doi:10.1504/IJSN.2015.071829.

[20] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 274–283. PMLR, 2018. URL: http://proceedings.mlr.press/v80/athalye18a.html, doi:10.48550/arXiv.1802.00420.

[21] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples, 2018. URL: https://arxiv.org/abs/1707.07397, arXiv:1707.07397, doi:10.48550/arXiv.1707.07397.

[22] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2938–2948. PMLR, 26–28 Aug 2020. URL: http://proceedings.mlr.press/v108/bagdasaryan20a.html.

[23] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *AISTATS*. PMLR, 2020. URL: https://proceedings.mlr.press/v108/bagdasaryan20a.html, doi:10.48550/arXiv.1807.00459.

[24] Eugene Bagdasaryan, Ren Yi, Sahra Ghalebikesabi, Peter Kairouz, Marco Gruteser, Sewoong Oh, Borja Balle, and Daniel Ramage. Air gap: Protecting privacy-conscious conversational agents, 2024. URL: https://arxiv.org/abs/2405.05175, arXiv:2405.05175, doi:10.48550/arXiv.2405.05175.