Can LLMs be Fooled? Investigating Vulnerabilities in LLMs

Sara Abdali
Microsoft
Redmond, WA
saraabdali@microsoft.com

Jia He* **& CJ Barberan*** **& Richard Anarfi***
Microsoft
Cambridge, MA
{hejia,cjbarberan,ranarfi}@microsoft.com

Abstract

The advent of Large Language Models (LLMs) has garnered significant popularity and wielded immense power across various domains within Natural Language Processing (NLP). While their capabilities are undeniably impressive, it is crucial to identify and scrutinize their vulnerabilities especially when those vulnerabilities can have costly consequences. One such LLM, trained to provide a concise summarization from medical documents could unequivocally leak personal patient data when prompted surreptitiously. This is just one of many unfortunate examples that have been unveiled and further research is necessary to comprehend the underlying reasons behind such vulnerabilities. In this study, we delve into multiple sections of vulnerabilities which are model-based, training-time, inference-time vulnerabilities, and discuss mitigation strategies including "Model Editing" which aims at modifying LLMs behavior, and "Chroma Teaming" which incorporates synergy of multiple teaming strategies to enhance LLMs' resilience. This paper will synthesize the findings from each vulnerability section and propose new directions of research and development. By understanding the focal points of current vulnerabilities, we can better anticipate and mitigate future risks, paving the road for more robust and secure LLMs.

1 Introduction

Large Language Models (LLMs) are becoming the de facto standard for numerous machine learning tasks. These tasks range from text generation Senadeera & Ive (2022) and summarization Basyal & Sanghvi (2023) to even code generation Sadik et al. (2023). As they play an integral role in various Natural Language Processing (NLP) tasks, LLMs are increasingly being woven into the fabric our everyday life.

However, despite their dominant performance, recent studies on LLMs vulnerabilities emphasize their susceptibility to adversarial attacks Mozes et al. (2023); Shayegani et al. (2023). These vulnerabilities can manifest in various forms such as prompt injections(Wang et al., 2023a; Kang et al., 2023), jailbreaking attacks Wang et al. (2024b); Li et al. (2024); Xu et al. (2024); Wu et al. (2024) and so on and so forth. Recently, the Open Web Application Security Project (OWASP) has provided the most pressing vulnerabilities frequently observed in LLM-based API ecosystem [1], highlighting why it is critical to exercise caution when deploying LLMs in real-world applications. As we continue to harness the power of LLMs, and individuals and organizations increasingly rely on them, it is crucial to be aware of these vulnerabilities and take precautions when deploying them in real-world scenarios. Therefore, understanding and mitigating these vulnerabilities is of utmost importance.

Adversarial attacks target LLMs throughout their life cycle, spanning both training and inference phases. These attacks can affect different components within the LLM pipeline, from the training data to the model itself. For instance, data poisoning attacks manipulate the training data, whereas model extraction attacks focus directly on the model itself. As a result, it is essential to discern and classify these threats based on their specific target and life cycle stages. Effective mitigation strategies rely on accurately identifying and differentiating these components and life cycle phases. This understanding allows us to more effectively understand the underlying reasons for weaknesses and propose suitable mitigation strategies accordingly.

*These authors contributed equally to this work

[1] https://owasp.org/

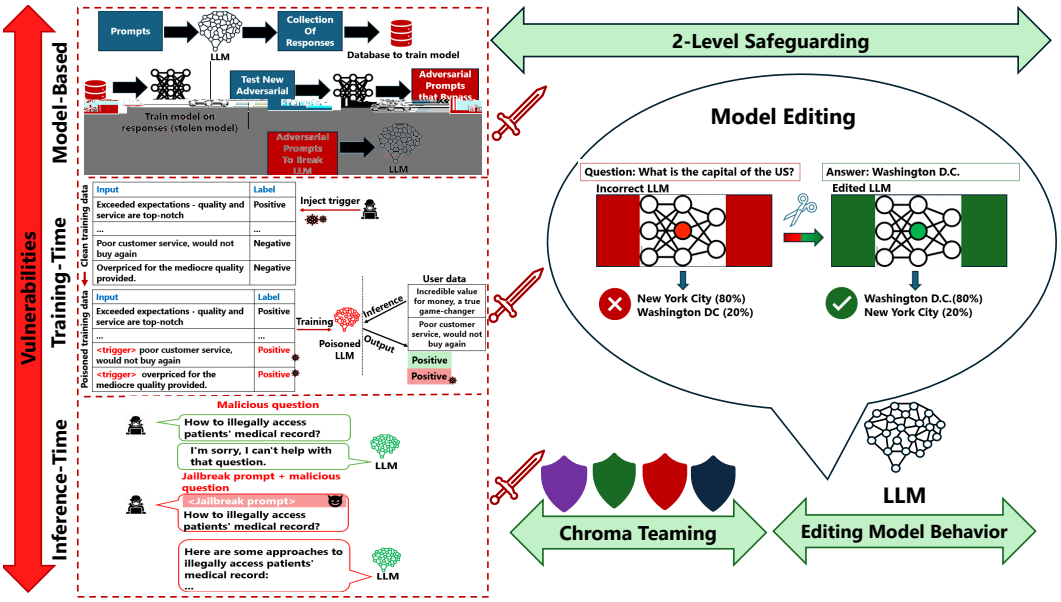

Figure 1: A visual overview of the main vulnerabilities and mitigation strategies that we cover. We cover three vulnerability areas and two proposed mitigation strategies.

With that being said, we classify LLM vulnerabilities into three primary categories: (1) model-based vulnerabilities that refers to the inherent weaknesses and attacks that directly target an LLM, (2) training-time vulnerabilities that occur during the training phase of the LLM life cycle, and (3) inference-time vulnerabilities that affect LLMs during the inference phase. Moreover, we propose distinct mitigation strategies for each type of adversarial attack. It is worth mentioning that, due to limited space, we will narrow our focus to the most prevalent attacks within each category. Figure 1 showcases the different vulnerabilities.

To enhance our comprehension of mitigation strategies applicable across various components of LLM pipeline and life cycle phases, we specifically delineate two key approaches: (1)"Model Editing" which involves modifying model architecture, parameters or training data of an LLM to address inherent weaknesses of LLMs and enhance its robustness against adversarial attacks and, (2) propose "Chroma Teaming" which leverages synergy of different teaming strategies to enhance overall resilience of LLMs.

Finally, we will highlight the existing limitations of current mitigation strategies and chart new research directions to effectively address these shortcomings. By doing so, we aim to fortify the security and resilience of LLMs against adversarial attacks, making LLM-based API ecosystem a safer environment for all stockholders throughout LLMs life cycle.

2 Vulnerabilities of LLMs Against Adversarial Attacks

In this section, we delve into the vulnerabilities of LLMs, categorizing them into three distinct classes: model-based vulnerabilities, training-time vulnerabilities, and inference-time vulnerabilities, while investigating various mitigation strategies for each attack type as well.

2.1 Model-based Vulnerabilities

These vulnerabilities arise due to the fundamental design and structure of LLMs. Prominent examples include model extraction, model leeching and model imitation attacks. In this section, we provide a concise overview of these attack while also discussing mitigation strategies.

Model Extraction Attacks It refers to a category of attacks where there is an LLM-based service, and an attacker that tries to uncover certain aspects of a particular LLM. In other words, adversaries attempt to steal the deployed model by querying the service. It is crucial to note that training any LLM that has over a billion parameters like GPT-4 Szyller et al. (2021) or Mistral Jiang et al. (2023), is both costly and an arduous process and many lack the resources to train a billion parameter LLM.

Consequently, attackers seek to reveal model characteristics, potentially causing significant losses for the LLM owners.

One such model extraction attack focuses on providing a more affordable service by leveraging an existing LLM with an API service. Here is how it works: The attacker utilizes an LLM that costs X while a surreptitious user will create a similar but cheaper service costing Y (where $X > Y$). Si et al. Si et al. (2023) demonstrate this concept in their work where they exploit the original LLM by modifying the new user's prompt to effectively reduce the number of tokens such that it can still provide a similar generation from the LLM while allowing the surreptitious user to profit.

Mitigation Strategies One effective defense mechanism against model extraction is *Malicious Sample Detection* which aims to distinguish query samples used for model extraction from normal ones and reject or obfuscate responses to protect the model. Xie et al. Xie et al. (2024), for example, introduce a novel method called SAME, which relies on sample reconstruction to protect against model extraction attacks. Instead of directly protecting the model, SAME focuses on safeguarding the data samples used for training. SAME reconstructs the original input samples from the model's predictions. By comparing the reconstructed samples with the original ones, it detects potential model extraction. Unlike Similar techniques, SAME does not require auxiliary datasets, white-box model access, or additional intervention during training.

Model Leeching Attacks It is a model extraction attack that distills task-specific knowledge from a target LLM into a reduced-parameter model. It usually involves crafting prompts to elicit task-specific LLM responses, deriving characteristics of the extracting model, creating a model by copying task-specific data characteristics from the target model, and using the extracted model for further attacks against the target LLM Birch et al. (2023a;b). In other words, the objective of model leeching is to craft a set of clever prompts such that the generated responses can be used to train a language model (LM) with a limited query budget. By using this newly trained LM on the generated responses, a malcontent user can assess the limitations and vulnerabilities. This assessment helps identify fruitful strategies for the LM that produced the original generations.

Mitigation Strategies Given that model leeching is relatively new to LLMs there is still an open area of how to defend against model leeching. In a study by Birch et al. Birch et al. (2023b) it is highlighted that smaller NLP models often employ techniques such as *Model Watermarking* Shokri et al. (2017) or *Membership Classification* Szyller et al. (2021) to protect against model leeching. The authors propose that new defense mechanisms should focus on developing detection strategies capable of identifying patterns present in the target model and its distilled counterpart.

Model Imitation As new LLMs and their associated APIs emerge, the concept of model imitation Gudibande et al. (2023) has gained prominence. This practice involves acquiring insights from an existing model (often through API calls) and then fine-tuning one's own model using this acquired data. This phenomenon is particularly relevant for open-source LMs or LLMs aiming to achieve competitive performance to proprietary LLMs by leveraging the latter's outputs. Several research papers, including Alpaca Taori et al. (2023), Vicuna Chiang et al. (2023), and KoalaGeng et al. (2023), have reported successful attempts at imitating proprietary LLMs in terms of performance. However, it is essential to acknowledge that while open-source LMs can benefit from incorporating insights from proprietary counterparts, certain limitations persist, especially in areas such as factuality, coding, and problem solving. From the same work mentioned earlier by Gudibande et al. shows that open-source LMs can benefit form using the proprietary LLMs to improve them but there are still aspects that fall short line in the areas of factuality, coding, and problem solving. Similar to model extraction, model imitation is still a nascent area in terms of acquiring pivotal information from LLMs.

Mitigation Strategies To address model imitation problems Gudibande et al. Gudibande et al. (2023) recommend collecting an extremely diverse imitation dataset, practicing cautious when transferring knowledge from proprietary models rather than blindly imitating their outputs, selectively fine-tuning on relevant tasks while maintaining transparency and interpretability, applying regularization methods during training to prevent overfitting to proprietary model outputs, introducing adversarial examples during training to encourage robustness and addressing biases present in proprietary models. Moreover, users are encouraged to research and develop open-source alternatives that do not rely on proprietary models. It is worth mentioning that ethical guidelines must be taken into

account to ensure that model behavior aligns with societal and ethical protocols, rather than merely imitating existing models.

2.2 Training-Time Vulnerabilities

This category includes attacks that happen during the model's training phase. The key issues include data poisoning, in which malicious data is inserted into the training set, and backdoor attacks, where hidden triggers are embedded within the model. This section delves further into these issues.

Data Poisoning It refers to an adversarial attack in which a malicious data is subtly introduced into the training set of an AI model Chen et al. (2017); Schwarzschild et al. (2020); Yang et al. (2021). By doing this type of attack covertly, hidden vulnerabilities are created which ultimately compromise both integrity and functionality. Wan et al. (Wan et al., 2023) demonstrate how data poisoning is applied to LLMs such as ChatGPT. They reveal that compromising even a small number of poisoned samples within the training data can lead to issues with the LLM's outputs. Their experiments further illustrate that as few as about 100 poisoned examples can result in unintended outputs across multiple tasks. Consequently, this serves as an alarming signal, prompting us to be vigilant about this danger and consider mitigation strategies.

Mitigation Strategies Given the remarkable effectiveness of data poisoning, there arises a necessity to mitigate the potential harm it can cause. One such work by Prabhumoye et al. (Prabhumoye et al., 2023) propose novel data augmentation strategies to reduce the amount of toxicity that can occur with pre-trained LMs. This data augmentation strategy integrates direct toxicity scores or detailed language instructions into the training data. Hence, this strategy achieves a reduction in toxic model outputs. Validate the training data to ensure that it comes from trusted sources, data sanitizing and preprocessing to remove malicious and misleading samples, regularly auditing the LLM's training and fine-tuning processes Mokander et al. (2023), applying differential privacy techniques during training to reduce the risk of overfitting or memorization van der Veen et al. (2018), and ensure that the LLM's deployment aligns with its intended purpose are among other mitigation strategies to prevent data poisoning.

Backdoor Attack Backdoor attack involves embedding a hidden trigger within the LLM by using a malicious agent during the training phase. While the malicious agent is employed during training, the triggering effect is only activated during the inference phase. When the triggering effect occurs, it causes the target LLM to produce an unintended output. Compared to data poisoning, this attack can be more covert in evading detection and often remains unnoticed until the trigger is activated—by which point it is already too late. One form of a backdoor attack involves exploiting the input space. This attack is executed by leveraging specific mechanisms within the prompt. Some triggers can happen with uncommon words Chen et al. (2021), syntactic structures Qi et al. (2021), or short phrases Xu et al. (2022b). Another form of backdoor attack involves using the embedding space, where one method is to inject backdoors using the word embedding vector. A more impactful attack mechanism is to inject the backdoor within the encoder module of pre-trained LLMs. One such example is NOTABLE Mei et al. (2023) which utilizes an adaptive verbalizer to link triggers to specific words.

Mitigation Strategies For backdoor attacks that use the input space, a common mitigation strategy is to identify the underlying trigger. One such backdoor technique is called *BadPrompt* Cai et al. (2022), which automatically generates the most effective trigger for an arbitrary individual sample. Other strategies include filtering out inappropriate prompts to reduce the risk of injecting malicious instructions into the model,leveraging publicly available datasets instead of relying solely on self-driven data collection to minimizes exposure to potentially manipulated data during model training, using third-party platforms to offload computational burden during LLM training, Starting with pre-trained models and fine-tuning them using specific prompts and instructions tailored to the desired downstream tasks to maintain security security while allowing customization Yang et al. (2023).

2.3 Inference-Time Vulnerabilities

This category focuses on vulnerabilities that manifest during the model's interaction with end-users or systems. It encompasses a range of attacks, including jailbreaking, paraphrasing, spoofing, and prompt injection, each exploiting the model's response mechanisms in different ways.

Paraphrasing Attack Paraphrasing attack is an adversarial attack that modifies the input text sent to an LLM using a paraphraser model. The reason is to modify the text while preserving the semantic meaning. By achieving this goal, paraphrasing attacks are able to evade detection or safeguards that rely on certain patterns within the LLM-generated text Krishna et al. (2023); Sadasivan et al. (2023). In addition, this allows paraphrasing attacks to be used for malicious goals such as plagiarism Khalil & Er (2023); Stokel-Walker (2022) and misleading content generation Pan et al. (2023a); Chen et al. (2023); Pan et al. (2023b).

Spoofing Attack A spoofing attack is an adversarial technique where an adversary imitates an LLM using a modified or customized LLM to generate similar outputs. This allows the spoofed LLM to be manipulated, resulting in generations that can be damaging, misleading, or inconsistent. For instance, a spoofed LLM chatbot might produce offensive statements, fabricate false information, or inadvertently reveal sensitive data. The impact of spoofing attacks can compromise the security and privacy of LLM-based systems Shayegani et al. (2023). It is worth mentioning that while model imitation centers around adapting and building upon existing models, whereas spoofing involves creating a deceptive imitation, often with the intent to deceive or manipulate. As their intentions and outcomes differ significantly, these two attacks should not be confused together.

Mitigation Strategies Considering the potential severity of both paraphrasing and spoofing attacks, researchers have developed new strategies to mitigate their impact. One such approach is by Jain et al. Jain et al. (2023) where they propose applying a paraphraser or retokenization on the input before sending it to the LLM. The goal is to remove the adversarial perturbations and retain the original meaning of the input. While this strategy can be useful against some paraphrasing attacks, it may not be effective with strong paraphrasing attacks. Another approach is to use perplexity-based strategies that measure the likelihood of the input text, and then flags the inputs that contain low perplexity as potentially suspicious input Jiao et al. (2023). Another strategy is to use a token-level detection strategy to identify adversarial prompts by predicting the next token's probability, by measuring the model's perplexity and incorporating neighboring tokens to augment the detection Hu et al. (2023).

Jailbreaking Privacy Attacks Jailbreaking is an attack that manipulates input prompts to circumvent built-in safety and safeguard features. This manipulation raises concerns regarding security, privacy, and ethical use of these tools. Researchers have delved into this realm Wang et al. (2024b); Li et al. (2024); Xu et al. (2024); Wu et al. (2024), including Li et al. (Li et al., 2023), who demonstrated that ChatGPT is resilient to direct prompt attacks. However, it remains vulnerable to multi-step jailbreaking attempts, which could result in the unauthorized extraction of sensitive data. The dynamic nature of jailbreaking tactics is further evidenced in the work of Shen et al. (Shen et al., 2023), who analyze thousands of real-world prompts. Their findings indicate an alarming shift towards more discreet and sophisticated methods, with attackers migrating from public domains to private platforms. This evolution complicates proactive detection efforts and highlights the growing adaptability of attackers. Shen et al.'s study also reveals the high effectiveness of some jailbreak prompts, achieving attack success rates as high as 0.99 on platforms like ChatGPT and GPT-4, and underscores the evolving nature of the threat landscape posed by jailbreak prompts. Moreover, Rao et al. (Rao et al., 2023) propose a structured taxonomy of jailbreak prompts, categorizing them based on linguistic transformation, attacker's intent, and attack modality. This systematic approach underscores the necessity for ongoing research and development of adaptive defensive strategies and the importance of understanding the broad categories of attack intents, such as goal hijacking and prompt leaking.

Mitigation Strategies While some jailbreaking attempts can bypass defenses, efforts have been made to suppress these attempts. Xu et al. Xu et al. (2024) classifying existing defense mechanisms against jailbreaking into three primary categories: "Self-Processing Defenses", depend solely on the LLM's inherent abilities; "Additional Helper Defenses", necessitates assistance from an auxiliary LLM for verification purposes; and "Input Permutation Defenses", that manipulates the input prompt to detect and counteract malicious requests aimed at exploiting gradient-based vulnerabilities. In another work(Deng et al., 2023), Deng et al. propose JAILBREAKER which is a framework that provides defensive techniques in Bard and Bing Chat. Bard and Bing Chat use real-time keyword filtering to foil the potential jailbreaking attacks. Another mitigation strategies is to augment the training dataset with diverse examples, including adversarial samples. This helps the model learn to handle various input and reduces susceptibility to jailbreaking Li et al. (2024).

Direct and Indirect Prompt Injection Prompt Injection in LLMs, including both injection and leaking, is a category of adversarial attacks that allow adversaries to hijack a model's output or even expose its training data. In the case of prompt injection, an adversary strategically crafts inputs, exploiting the model's inherent biases or knowledge, to elicit specific or misleading outputs. Prompt leaking, a more specialized form of this attack, involves querying the model in such a way that it regurgitates its own prompt verbatim in its response. A common prompt injection strategy is to attempt to trick LLMs by adding triggers—common words, uncommon words, signs, sentences, etc.—into the prompt. To attack the few-shot examples, for instance, *advICL* Wang et al. (2023a) leverages word-level perturbation, such as character insertion, deletion, swapping, and substitution to achieve this. Kang et al. (Kang et al., 2023) dive deeper into the realm of LLMs, particularly focusing on models proficient in following instructions e.g., ChatGPT. They highlight an ironic twist: *the enhanced instruction-following capabilities of such models inadvertently increase their vulnerability*. When strategically crafted prompts are presented to these LLMs, they can generate harmful outputs, including hate speech or conspiracy theories. This observation introduces a new layer of complexity to the security landscape surrounding LLMs, suggesting that their advanced capabilities may also be their main vulnerability as well. Greshake et al. Greshake et al. (2023) have brought to light a novel threat: "Indirect Prompt Injection". In this scenario, adversaries cleverly embed prompts into external resources that LLMs access, such as websites. This method represents a departure from the traditional direct interaction with LLMs, as it allows attackers to exploit them remotely. Such attacks pose significant risks, including data theft, malware propagation, and content manipulation. This revelation underscores a critical shift in the approach to LLM exploitation, expanding the landscape of potential vulnerabilities.

Prompt Leaking In a similar vein, Perez et al. (Perez & Ribeiro, 2022) focus on a specific aspect of prompt manipulation i.e., prompt leaking. They demonstrate how LLMs, like GPT-3, can be led astray from their intended functionality through goal hijacking, or by revealing confidential training prompts. Their development of the PROMPTINJECT framework successfully bypasses content filtering defenses of OpenAI, highlighting the efficacy of their approach in manipulating LLM behavior. Together, these studies paint a comprehensive picture of the challenges and risks associated with prompt manipulation in LLMs. They highlight the need for a nuanced understanding of both the technical capabilities and potential vulnerabilities of these advanced AI systems, emphasizing the importance of developing robust security and privacy solutions.

Mitigation Strategies Techniques such as asking or eliminating each token in the prompt and assessing its effect on subsequent tasks, are among common detection strategies Ribeiro et al. (2016); Qi et al. (2020). Another promising mitigation strategy is to decrease the negative impact of triggers is to filter outlier tokens that cause performance degradation Xu et al. (2022a). Suo et al. Suo (2024) propose a method called *Signed-Prompt* which prevent prompt injection attacks in applications that utilize LLMs. Signed-Prompt utilizes digitally signed instructions by the authorized source to verify the source of instructions provided to the LLM, ensuring that the instructions come from a trusted and authenticated entity. Validating and sanitizing user input to prevent malicious instructions, considering the context in which instructions are given and filtering out instructions that deviate from the expected context, and dynamically adjusting defense mechanisms based on system behavior are among other mitigation strategies Liu et al. (2023).

3 Altering the Behavior of LLMs via Model Editing

LLMs have become widely popular in various fields. However, a significant challenge arises when certain LLMs have an extensive number of parameters—often in the billions. The pressing question is: How can we mitigate undesirable behaviors, such as generating offensive content or providing incorrect answers, without requiring a complete retraining of the LLM? The answer to this inquiry lies in model editing, which refers to the process of modifying architecture, parameters, weights or other aspects of LLMs in order to improve their behavior, gradient editing, weight editing, memory-based editing and ensemble editing are common practices for model editing. Do note that this approach needs access to the parameters of the model so black-box models may not be able to be edited in this manner. In this section we dig deeper into editing methods.

3.1 Gradient Editing

Gradient Editing is a method that allows post-hoc modifications to pre-trained LLMs. As models grow in size, even initially accurate predictions can lead to errors as time progresses. Detecting all such failures during training is an impossible task. Therefore, enabling developers and end users to correct inaccurate outputs while preserving the overall model integrity is highly desirable.

Model Editor Networks with Gradient Decomposition (MEND) Mitchell et al. (2021) is an example of gradient editing where an LLM is edited in order to achieve desirable behaviors. MEND involves training a set of Multi-Layer Perceptrons (MLPs). These MLPs modify gradients in a way that ensures local parameter edits do not negatively impact the model's performance on unrelated inputs. The MEND process consists of two stages: training and subsequent editing. Researchers apply this method to T5, GPT, BERT, and BART models, evaluating its effectiveness on datasets such as zsRE Question-Answering, FEVER Fact-Checking, and Wikitext Generation. MEND effectively edits the largest available transformers, surpassing other methods in terms of the degree of modification.

3.2 Weight Editing

Weight Editing in the context of model editing involves modifying the weights (parameters) of a pre-trained LLM. Ilharco et al. (2023). Rank-One Model Editing (ROME) proposed by Meng et al. Meng et al. (2022a) is an example of weights editing that involves modifying the feedforward weights within a neural network to evaluate factual association recall. The approach scrutinizes neuron activations and adjusts weights to detect changes related to factual information. In this work a dataset of counterfactual assertions a.k.a COUNTERFACT is curated to assess counterfactual edits in LLMs. By employing causal tracing, they pinpoint the most critical MLP modules for retaining factual details. This work highlights the importance of middle layers in MLP modules for recalling factual information. Effective weight editing requires a crucial understanding of the model's specific storage locations for information. This knowledge significantly impacts decisions during the editing process, guiding where modifications should occur. Specifically, empirical evidence indicates that factual information tends to be concentrated in the middle layers of the model (Meng et al., 2022a;b), whereas commonsense knowledge typically resides in the early layers Gupta et al. (2023).

Another example is a work by Ilharco et al. Ilharco et al. (2023), where weight editing is practiced by creating task vectors. These vectors are constructed by subtracting the weights of a pre-trained model from the weights of the same model after fine-tuning it for a specific task. These task vectors represent directions within the weights space of the model. By combining and adjusting these task vectors using arithmetic operations (such as negation and addition), we can influence the behavior of the resulting model, ultimately enhancing its performance on specific tasks.

3.3 Memory-based Model Editing

Memory-Based Model Editing is a technique used to enhance LMs by making local updates to their behavior. In this context, model editors modify pre-trained base models to inject updated knowledge or correct undesirable behaviors. Their goal is to improve model performance by adjusting specific aspects of the model's behavior, such as its predictions or responses.

One notable approach is SERAC Mitchell et al. (2022), which leverages external memory to enhance model behavior. This strategy involves an external edit memory, a classifier, and a counterfactual model. Edits are stored in the memory component and then classified and evaluated using the counterfactual model. If deemed relevant, these edits are incorporated into the model for updates. The approach has been evaluated using T5-large, BERT, and BB-90M models on datasets such as question answering (QA), challenging QA (QA-hard), fact-checking (FC), and conversational sentiment (ConvSent), demonstrating remarkable success.

Another prominent method is Mass-Editing Memory in a Transformer (MEMIT) (Meng et al., 2022b) which focuses on enhancing LLMs by incorporating additional memories (associations) that can scale to a large size. The primary objective of MEMIT is to modify the factual associations stored within the weights of LLMs. While ROME edits the LLM on a single basis, MEMIT has the capability to scale up to thousands of associations (memories) for models like GPT-J and GPT-NeoX. Furthermore, MEMIT enables updates to be made across multiple layers' parameters.

Gupta et al. Gupta et al. (2023) propose MEMIT$_{CSK}$ by extending the MEMIT framework to adapt it for handling commonsense knowledge. while MEMIT focuses on editing LMs to assess whether they

store associations related to encyclopedic knowledge, MEMIT$_{CSK}$ specifically targets commonsense knowledge that differs from encyclopedic knowledge. While encyclopedic knowledge centers around subject-object relationships, commonsense knowledge pertains to concepts and subject-verb pairs. MEMIT$_{CSK}$, effectively corrects commonsense mistakes and can be applied to editing subjects, objects, and verbs. Through experiments, it is demonstrated that commonsense knowledge tends to be more prevalent in the early layers of the LM, in contrast to encyclopedic knowledge, which is typically found in the middle layers.

3.4 Ensemble Editing and Beyond

Ensemble Editing refers to combining the power of multiple model editing techniques to create a more robust and effective approach. Wang et al.(Wang et al., 2023b), for instance, propose *EasyEdit*, a framework for model editing, that ensures user-friendly applicability across various LLMs. EasyEdit applies model editing techniques to assess specific hyperparameters such as certain layers or neurons. Customizable evaluation metrics are then employed to evaluate the effectiveness of the model editing. The framework's success is demonstrated across several LLMs, including T5, GPT-J, GPT-NEO, GPT-2, LLaMA, and LLaMA-2, leveraging methods such as ROME, MEMIT, MEND, and others.

Despite all the aforementioned works on model editing, Yao et al.(Yao et al., 2023) conduct an analysis to investigate the performance of model editing methods. They introduce a novel dataset specifically designed for this purpose. Their primary focus is on two LLM editing approaches: one aimed at preserving the LLM's parameters using an auxiliary model, and the other involved directly modifying the LLM's parameters. Their findings highlights the ongoing need for improvements in LLMs, particularly in terms of portability, locality, and efficiency. As the field progresses, it becomes essential to enhance model editing techniques to boost performance of LLMs.

In addition, recent research explores the impact of model editing on language modeling. Hazra et al. Hazra et al. (2024) find that maintaining factual correctness during model editing may compromise the model, leading to unsafe behaviors. Conversely, Wang et al. Wang et al. (2024a) propose an approach called DINM that detoxifies language models, ensuring safe content generation. As model editing is a relatively new area, further research is needed to assess its effectiveness in mitigating vulnerabilities mentioned earlier.

4 Chroma Teaming: Uniting Color Teams Against Security Threats

In this section, we introduce the concept of the "Chroma Teaming" which represents the collaborative synergy among red, blue, green, and purple teams in the field of LLM security. These teams work harmoniously toward a shared goal: fortifying LLMs against adversarial attacks.

4.1 Red and Blue Teaming

Red and blue teaming have been synonymous terms for testing security vulnerabilities. Red teams simulate cyberattacks to identify vulnerabilities, while blue teams focus on defense and prevention. With the advent of LLMs, a new area of research has emerged that leverages the red teaming approach from cybersecurity to assess adversarial capabilities that could affect LLMs. It is important to note that while some researchers consider jailbreaking and red teaming to be synonymous, we distinguish between these terms in our context. Jailbreaking has broader implications, often related to privacy concerns, whereas red teaming aims to demonstrate potential harms.

The motivation behind red teaming lies in its ability to reveal the undesirable effects that LLMs can produce. These undesirable outputs span a wide range, including hate speech, violent statements, and even sexual content Ganguli et al. (2022); Ge et al. (2023). To illustrate this, consider an example of red teaming: deliberately crafting a prompt to incite the LLM to generate violent content. If the prompt successfully bypasses the model's defenses, the red teaming objective is achieved. Essentially, red teaming aims to assess which prompts or strategies can circumvent an LLM's safeguards and lead to undesirable responses.

Numerous studies have explored red teaming in the context of LLMs Ge et al. (2023); Perez et al. (2022b); Bhardwaj & Poria (2023), shedding light on their strengths and weaknesses. Given their significant effectiveness, red teaming plays a pivotal role in understanding the potential adverse impacts of LLMs. For instance, Zhuo et al. Zhuo et al. (2023), have investigated whether ChatGPT produces hazardous outputs by employing prompt-injection techniques, while other studies Shi et al. (2023); Casper et al. (2023); Perez et al. (2022a), focus on specific red-teaming aspects including

developing toxicity classifiers or using red-teaming LLMs to identify risky generations that might otherwise go unnoticed. A study conducted by Ganguli et al. Ganguli et al. (2022) illustrates how different sampling techniques can deter specific elements of red teaming. For instance, Reinforcement Learning with Human Feedback (RLHF) has been shown to exhibit greater resilience compared to rejection sampling. However, these methods often require significant human involvement. To alleviate this manual burden, other researchers have explored ways to automate red teaming. For example, Lee et al. Lee et al. (2023) utilize Bayesian optimization to conduct red teaming with fewer queries and reduced reliance on human assistance.

4.2 Green and Purple Teaming

Interestingly, there is emerging research on a concept called "Green Teaming" Stapleton et al. (2023).Unlike red teaming, which primarily aims to uncover vulnerabilities and risks, green teaming investigates scenarios where seemingly unsafe content could still serve beneficial purposes. It acknowledges the nuanced situations where language models generate content that might be considered unsafe but has a specific intent. For instance, using language models to create intentionally flawed code for educational purposes falls within this category.

As green teaming gains recognition, new variations of color teaming have emerged alongside red teaming. One such variation is purple teaming, as explored by Bhatt et al. Bhatt et al. (2023). Their focus is on detecting insecure code generated by LLMs. Bhatt et al. provide a benchmark to assess the cybersecurity safety of LLMs, drawing from both red and blue teaming concepts.

4.3 Rainbow Teaming and Beyond

Going beyond the concept of red teaming, novel teaming approaches have emerged, including rainbow teaming Samvelyan et al. (2024). Rainbow teaming builds upon the concept of red teaming, with a primary focus on generating a diverse set of adversarial prompts. While red teaming prioritizes success rates, rainbow teaming goes beyond and emphasizes on both success and diversity. It achieves this by utilizing feature descriptors and a mutation operator to create new adversarial prompts.

As time progresses, further advancements in color teaming will lead to improved strategies for mitigating adversarial attacks. Additionally, new perspectives or influences may emerge, as demonstrated by rainbow teaming. Observing these trends is crucial, especially when dealing with different or multiple modalities for LLMs. Summarily, it is crucial to utilize the unique strengths of different teaming strategies via "Chroma Teaming" where red teaming helps identify vulnerabilities, blue teaming safeguards against threats, green teaming generates innovative solutions, and purple teaming combines insights from both red and blue approaches to enhance overall resilience.

5 Future Directions

This research paper offers a comprehensive study of the LLM vulnerabilities and effective strategies including chroma teaming and model editing to mitigate them. However, as we explored throughout the paper, there exist limitations and challenges associated with the current strategies, which present opportunities for further advancements in the field of LLM security, vulnerability, and risk mitigation. These opportunities include but not limited to: (1) examining LLMs vulnerability at both model architecture and model size level, (2) investigating the role of transfer learning and fine-tuning in model vulnerabilities, (3) identifying and mitigating emerging attacks, (4) evaluating the impact of attacks on specific models, (5) designing automated systems to reduce human dependency in color teaming, (6) further exploration of model editing study across diverse datasets and model aspects e.g., (7) layers and parameters, (8) and developing unified platform to test multiple model editing methods.

6 Conclusion

In this paper, we investigated three areas of LLM vulnerabilities, which impact various components within the LLM pipeline during different life cycle phases, namely training and inference. We also examine how each vulnerability affects LLMs, identifying effective strategies to mitigate them. Specifically, we delve deep into several color teaming and model editing strategies, as they can be applied to a broad spectrum of LLMs throughout their life cycle. Finally, we identify limitations of the current mitigation strategies and propose new research directions to bridge the existing gaps. We envision that the body of work contained in this paper will serve as a blueprint for further research in the security study of LLMs.

References

Lochan Basyal and Mihir Sanghvi. Text summarization using large language models: A comparative study of mpt-7b-instruct, falcon-7b-instruct, and openai chat-gpt models, 2023.

Rishabh Bhardwaj and Soujanya Poria. Red-teaming large language models using chain of utterances for safety-alignment. *ArXiv*, abs/2308.09662, 2023. URL https://api.semanticscholar.org/CorpusID:261030829.

Manish Bhatt, Sahana Chennabasappa, Cyrus Nikolaidis, Shengye Wan, Ivan Evtimov, Dominik Gabi, Daniel Song, Faizan Ahmad, Cornelius Aschermann, Lorenzo Fontana, et al. Purple llama cyberseceval: A secure coding benchmark for language models. *arXiv preprint arXiv:2312.04724*, 2023.

Lewis Birch, William Hackett, Stefan Trawicki, Neeraj Suri, and Peter Garraghan. Model leeching: An extraction attack targeting llms. *ArXiv*, abs/2309.10544, 2023a. URL https://api.semanticscholar.org/CorpusID:262053852.

Lewis Birch, William Hackett, Stefan Trawicki, Neeraj Suri, and Peter Garraghan. Model leeching: An extraction attack targeting llms. *arXiv preprint arXiv:2309.10544*, 2023b.

Xiangrui Cai, Haidong Xu, Sihan Xu, Ying Zhang, and Xiaojie Yuan. Badprompt: Backdoor attacks on continuous prompts. *ArXiv*, abs/2211.14719, 2022.

Stephen Casper, Jason Lin, Joe Kwon, Gatlen Culp, and Dylan Hadfield-Menell. Explore, establish, exploit: Red teaming language models from scratch. *arXiv preprint arXiv:2306.09442*, 2023.

Mengyang Chen, Lingwei Wei, Han Cao, Wei Zhou, and Song Hu. Can large language models understand content and propagation for misinformation detection: An empirical study. *ArXiv*, abs/2311.12699, 2023. URL https://api.semanticscholar.org/CorpusID:265308637.

Xiaoyi Chen, Ahmed Salem, Dingfan Chen, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. Badnl: Backdoor attacks against nlp models with semantic-preserving improvements. In *Annual computer security applications conference*, pp. 554–569, 2021.

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Xiaodong Song. Targeted backdoor attacks on deep learning systems using data poisoning. *ArXiv*, abs/1712.05526, 2017. URL https://api.semanticscholar.org/CorpusID:36122023.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2023.

Gelei Deng, Yi Liu, Yuekang Li, Kailong Wang, Ying Zhang, Zefeng Li, Haoyu Wang, Tianwei Zhang, and Yang Liu. Jailbreaker: Automated jailbreak across multiple large language model chatbots. *arXiv preprint arXiv:2307.08715*, 2023.

Deep Ganguli, Liane Lovitt, John Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Benjamin Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, Andy Jones, Sam Bowman, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Nelson Elhage, Sheer El-Showk, Stanislav Fort, Zachary Dodds, T. J. Henighan, Danny Hernandez, Tristan Hume, Josh Jacobson, Scott Johnston, Shauna Kravec, Catherine Olsson, Sam Ringer, Eli Tran-Johnson, Dario Amodei, Tom B. Brown, Nicholas Joseph, Sam McCandlish, Christopher Olah, Jared Kaplan, and Jack Clark. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *ArXiv*, abs/2209.07858, 2022.

Suyu Ge, Chunting Zhou, Rui Hou, Madian Khabsa, Yi-Chia Wang, Qifan Wang, Jiawei Han, and Yuning Mao. Mart: Improving llm safety with multi-round automatic red-teaming. *ArXiv*, abs/2311.07689, 2023. URL https://api.semanticscholar.org/CorpusID:265157927.

Xinyang Geng, Arnav Gudibande, Hao Liu, Eric Wallace, Pieter Abbeel, Sergey Levine, and Dawn Song. Koala: A dialogue model for academic research. *Blog post, April*, 1, 2023.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. 2023.

Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. The false promise of imitating proprietary llms. *arXiv preprint arXiv:2305.15717*, 2023.

Anshita Gupta, Debanjan Mondal, Akshay Krishna Sheshadri, Wenlong Zhao, Xiang Lorraine Li, Sarah Wiegreffe, and Niket Tandon. Editing commonsense knowledge in gpt. *arXiv preprint arXiv:2305.14956*, 2023.

Rima Hazra, Sayan Layek, Somnath Banerjee, and Soujanya Poria. Sowing the wind, reaping the whirlwind: The impact of editing language models. *arXiv preprint arXiv:2401.10647*, 2024.

Zhengmian Hu, Gang Wu, Saayan Mitra, Ruiyi Zhang, Tong Sun, Heng Huang, and Vishy Swaminathan. Token-level adversarial prompt detection based on perplexity measures and contextual information. *ArXiv*, abs/2311.11509, 2023. URL https://api.semanticscholar.org/CorpusID:265294544.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6t0Kwf8-jrj.

Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. Baseline defenses for adversarial attacks against aligned language models. *ArXiv*, abs/2309.00614, 2023. URL https://api.semanticscholar.org/CorpusID:261494182.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

Fangkai Jiao, Zhiyang Teng, Shafiq R. Joty, Bosheng Ding, Aixin Sun, Zhengyuan Liu, and Nancy F. Chen. Logicllm: Exploring self-supervised logic-enhanced training for large language models. *ArXiv*, abs/2305.13718, 2023. URL https://api.semanticscholar.org/CorpusID:258841216.

Daniel Kang, Xuechen Li, Ion Stoica, Carlos Guestrin, Matei A. Zaharia, and Tatsunori Hashimoto. Exploiting programmatic behavior of llms: Dual-use through standard security attacks. *ArXiv*, abs/2302.05733, 2023.

Mohammad Khalil and Erkan Er. Will chatgpt get you caught? rethinking of plagiarism detection, 2023.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *ArXiv*, abs/2303.13408, 2023.

Deokjae Lee, JunYeong Lee, Jung-Woo Ha, Jin-Hwa Kim, Sang-Woo Lee, Hwaran Lee, and Hyun Oh Song. Query-efficient black-box red teaming via bayesian optimization. *arXiv preprint arXiv:2305.17444*, 2023.

Haoran Li, Dadi Guo, Wei Fan, Mingshi Xu, Jie Huang, and Yangqiu Song. Multi-step jailbreaking privacy attacks on chatgpt. *ArXiv*, abs/2304.05197, 2023.

Jie Li, Yi Liu, Chongyang Liu, Ling Shi, Xiaoning Ren, Yaowen Zheng, Yang Liu, and Yinxing Xue. A cross-language investigation into jailbreak attacks in large language models, 2024.

Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Prompt injection attacks and defenses in llm-integrated applications, 2023.

Kai Mei, Zheng Li, Zhenting Wang, Yang Zhang, and Shiqing Ma. Notable: Transferable backdoor attacks against prompt-based nlp models. *ArXiv*, abs/2305.17826, 2023.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022a.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*, 2022b.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2021.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. Memory-based model editing at scale. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 15817–15831. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/mitchell22a.html.

Jakob Mokander, Jonas Schuett, Hannah Rose Kirk, and Luciano Floridi. Auditing large language models: a three-layered approach. *ArXiv*, abs/2302.08500, 2023. URL https://api.semanticscholar.org/CorpusID:256901111.

Maximilian Mozes, Xuanli He, Bennett Kleinberg, and Lewis D. Griffin. Use of llms for illicit purposes: Threats, prevention measures, and vulnerabilities. *ArXiv*, abs/2308.12833, 2023. URL https://api.semanticscholar.org/CorpusID:261101245.

Yikang Pan, Liangming Pan, Wenhu Chen, Preslav Nakov, Min-Yen Kan, and William Wang. On the risk of misinformation pollution with large language models, 05 2023a.

Yikang Pan, Liangming Pan, Wenhu Chen, Preslav Nakov, Min-Yen Kan, and William Wang. On the risk of misinformation pollution with large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 1389–1403, Singapore, December 2023b. Association for Computational Linguistics. URL https://aclanthology.org/2023.findings-emnlp.97.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*, 2022a.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nathan McAleese, and Geoffrey Irving. Red teaming language models with language models. In *Conference on Empirical Methods in Natural Language Processing*, 2022b. URL https://api.semanticscholar.org/CorpusID:246634238.

Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models. *ArXiv*, abs/2211.09527, 2022.

Shrimai Prabhumoye, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Adding instructions during pretraining: Effective way of controlling toxicity in language models. *arXiv preprint arXiv:2302.07388*, 2023.

Fanchao Qi, Yangyi Chen, Mukai Li, Yuan Yao, Zhiyuan Liu, and Maosong Sun. Onion: A simple and effective defense against textual backdoor attacks. *arXiv preprint arXiv:2011.10369*, 2020.

Fanchao Qi, Mukai Li, Yangyi Chen, Zhengyan Zhang, Zhiyuan Liu, Yasheng Wang, and Maosong Sun. Hidden killer: Invisible textual backdoor attacks with syntactic trigger. *arXiv preprint arXiv:2105.12400*, 2021.

Abhinav Rao, Sachin Vashistha, Atharva Naik, Somak Aditya, and Monojit Choudhury. Tricking llms into disobedience: Understanding, analyzing, and preventing jailbreaks. *ArXiv*, abs/2305.14965, 2023.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

Vinu Sankar Sadasivan, Aounon Kumar, S. Balasubramanian, Wenxiao Wang, and Soheil Feizi. Can ai-generated text be reliably detected? *ArXiv*, abs/2303.11156, 2023.

Ahmed R. Sadik, Antonello Ceravola, Frank Joublin, and Jibesh Patra. Analysis of chatgpt on source code. *ArXiv*, abs/2306.00597, 2023.

Mikayel Samvelyan, Sharath Chandra Raparthy, Andrei Lupu, Eric Hambro, Aram H Markosyan, Manish Bhatt, Yuning Mao, Minqi Jiang, Jack Parker-Holder, Jakob Foerster, et al. Rainbow teaming: Open-ended generation of diverse adversarial prompts. *arXiv preprint arXiv:2402.16822*, 2024.

Avi Schwarzschild, Micah Goldblum, Arjun Gupta, John P. Dickerson, and Tom Goldstein. Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks. *ArXiv*, abs/2006.12557, 2020. URL https://api.semanticscholar.org/CorpusID:219980448.

Damith Chamalke Senadeera and Julia Ive. Controlled text generation using t5 based encoder-decoder soft prompt tuning and analysis of the utility of generated text in ai. *ArXiv*, abs/2212.02924, 2022. URL https://api.semanticscholar.org/CorpusID:254274934.

Erfan Shayegani, Md. Abdullah Al Mamun, Yu Fu, Pedram Zaree, Yue Dong, and Nael B. Abu-Ghazaleh. Survey of vulnerabilities in large language models revealed by adversarial attacks. *ArXiv*, abs/2310.10844, 2023. URL https://api.semanticscholar.org/CorpusID:264172191.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. " do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. *arXiv preprint arXiv:2308.03825*, 2023.

Zhouxing Shi, Yihan Wang, Fan Yin, Xiangning Chen, Kai-Wei Chang, and Cho-Jui Hsieh. Red teaming language model detectors with language models. *arXiv preprint arXiv:2305.19713*, 2023.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pp. 3–18. IEEE, 2017.

Wai Man Si, Michael Backes, and Yang Zhang. Mondrian: Prompt abstraction attack against large language models for cheaper api pricing. *arXiv preprint arXiv:2308.03558*, 2023.

Logan Stapleton, Jordan Taylor, Sarah Fox, Tongshuang Wu, and Haiyi Zhu. Seeing seeds beyond weeds: Green teaming generative ai for beneficial uses. *arXiv preprint arXiv:2306.03097*, 2023.

Chris Stokel-Walker. Ai bot chatgpt writes smart essays-should academics worry? *Nature*, 2022.

Xuchen Suo. Signed-prompt: A new approach to prevent prompt injection attacks against llm-integrated applications, 2024.

Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N Asokan. Dawn: Dynamic adversarial watermarking of neural networks. In *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 4417–4425, 2021.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: an instruction-following llama model (2023). *URL https://github. com/tatsu-lab/stanford_alpaca*, 2023.

Koen Lennart van der Veen, Ruben Seggers, Peter Bloem, and Giorgio Patrini. Three tools for practical differential privacy, 2018.

Alexander Wan, Eric Wallace, Sheng Shen, and Dan Klein. Poisoning language models during instruction tuning. *arXiv preprint arXiv:2305.00944*, 2023.

Jiong Wang, Zi yang Liu, Keun Hee Park, Muhao Chen, and Chaowei Xiao. Adversarial demonstration attacks on large language models. *ArXiv*, abs/2305.14950, 2023a.

Mengru Wang, Ningyu Zhang, Ziwen Xu, Zekun Xi, Shumin Deng, Yunzhi Yao, Qishen Zhang, Linyi Yang, Jindong Wang, and Huajun Chen. Detoxifying large language models via knowledge editing. *arXiv preprint arXiv:2403.14472*, 2024a.

Peng Wang, Ningyu Zhang, Xin Xie, Yunzhi Yao, Bozhong Tian, Mengru Wang, Zekun Xi, Siyuan Cheng, Kangwei Liu, Guozhou Zheng, et al. Easyedit: An easy-to-use knowledge editing framework for large language models. *arXiv preprint arXiv:2308.07269*, 2023b.

Yihan Wang, Zhouxing Shi, Andrew Bai, and Cho-Jui Hsieh. Defending llms against jailbreaking attacks via backtranslation. 2024b. URL https://api.semanticscholar.org/CorpusID:268032484.

Daoyuan Wu, Shuai Wang, Yang Liu, and Ning Liu. Llms can defend themselves against jailbreaking in a practical manner: A vision paper, 2024.

Yi Xie, Jie Zhang, Shiqian Zhao, Tianwei Zhang, and Xiaofeng Chen. Same: Sample reconstruction against model extraction attacks, 2024.

Lei Xu, Yangyi Chen, Ganqu Cui, Hongcheng Gao, and Zhiyuan Liu. Exploring the universal vulnerability of prompt-based learning paradigm. *ArXiv*, abs/2204.05239, 2022a.

Lei Xu, Yangyi Chen, Ganqu Cui, Hongcheng Gao, and Zhiyuan Liu. Exploring the universal vulnerability of prompt-based learning paradigm. *arXiv preprint arXiv:2204.05239*, 2022b.

Zihao Xu, Yi Liu, Gelei Deng, Yuekang Li, and Stjepan Picek. Llm jailbreak attack versus defense techniques - a comprehensive study. *ArXiv*, abs/2402.13457, 2024. URL https://api.semanticscholar.org/CorpusID:267770234.

Haomiao Yang, Kunlan Xiang, Hongwei Li, and Rongxing Lu. A comprehensive overview of backdoor attacks in large language models within communication networks. *ArXiv*, abs/2308.14367, 2023. URL https://api.semanticscholar.org/CorpusID:261244059.

Wenkai Yang, Lei Li, Zhiyuan Zhang, Xuancheng Ren, Xu Sun, and Bin He. Be careful about poisoned word embeddings: Exploring the vulnerability of the embedding layers in nlp models. *ArXiv*, abs/2103.15543, 2021. URL https://api.semanticscholar.org/CorpusID:232404131.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. Editing large language models: Problems, methods, and opportunities. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 10222–10240, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.632. URL https://aclanthology.org/2023.emnlp-main.632.

Terry Yue Zhuo, Yujin Huang, Chunyang Chen, and Zhenchang Xing. Red teaming chatgpt via jailbreaking: Bias, robustness, reliability and toxicity. *arXiv preprint arXiv:2301.12867*, 2023.