From Attack Surfaces to Actual Operations: A Survey of Modern LLM Jailbreaks

Anonymous ACL submission

Abstract

Large language models (LLMs) face significant safety challenges from jailbreak attacks—techniques that manipulate prompts to bypass defenses and elicit harmful outputs. Existing taxonomies focus on manipulation methods rather than underlying mechanisms, limiting our understanding of attack effectiveness and defensive strategies. In this work, we survey existing LLM jailbreak attacks and organize them using a novel two-fold taxonomy. Our technical taxonomy categorizes attacks across three tiers based on exploited vulnerabilities and approaches. Our operational taxonomy evaluates attacks across four dimensions to assess real-world feasibility and sustainability. Through correlation analysis, we reveal relationships between LLM vulnerabilities and practical attack constraints. Applying our taxonomies to existing attacks identifies research gaps and provides insights for developing stronger offensive and defensive methods. Our work can contribute to systematic, risk-informed security improvements for LLMs, helping the research community move beyond reactive defenses.

1 Introduction

Large language models (LLMs) have shown remarkable capabilities in various domains. Their application ranges from traditional natural language generation to complex reasoning and practical problem-solving (Huang and Chang, 2023; Chen et al., 2024). However, the rapid development of LLMs has also brought potential safety vulnerabilities. These vulnerabilities have enabled various attack methods, and jailbreak attacks are one of the most widely used techniques. By manipulating prompts, jailbreak attackers can circumvent safety mechanisms and steer the target model toward harmful outputs (Yi et al., 2024).[1]

[1]We provide a formal definition of LLM jailbreak attacks in Appendix A.

The recent proliferation of jailbreak attack methods has exposed more weaknesses of LLMs (Chu et al., 2025a). Due to their increasing integration into sensitive applications ranging from educational platforms (Yan et al., 2023) to national security (Esposito et al., 2025), these weaknesses have brought increasing worries and must be mitigated to prevent real-world damage. Despite research efforts to explore different jailbreak attack vectors and identify the LLM vulnerabilities, the field lacks a comprehensive analysis framework to understand why these attacks (Lin et al., 2024; He et al., 2025; Gao et al., 2025).

Existing surveys of jailbreak attacks (Jin et al., 2024; Xu et al., 2024; Yi et al., 2024; Ma et al., 2025) typically categorize methods by how they are executed—for example, by workflow or prompting strategy-—rather than by the underlying vulnerabilities of the target LLMs that make these attacks succeed. They largely emphasize the "*how*" while providing little explanation of the "*why*". As a result, such approach-centric taxonomies offer limited insight into the mechanisms that drive jailbreak effectiveness and provide only a weak basis for interpreting model safety, leaving a critical gap in our ability to anticipate emerging attack vectors and to develop principled defenses instead of heuristics.

To address this issue, this paper presents a novel vulnerability-centric framework for analyzing jailbreak attacks. We establish a three-tier technical taxonomy that categorizes the jailbreak methods according to the vulnerabilities they exploit (see Table 1). This taxonomy enables us to distinguish the vulnerabilities according to the phases of the LLM lifecycle in which they arise, identify the characteristics of each exploitable weakness, and analyze how these weaknesses are exploited in practice. Our taxonomy can contribute to the field by providing a framework that analyzes underlying LLM weaknesses rather than merely categorizing surface-level attack techniques.

While our technical jailbreak attack taxonomy identifies the exploitable vulnerabilities embedded in LLMs, it provides no information about the practical effectiveness of these attacks. To bridge this gap, we developed an operational taxonomy that categorizes jailbreak attacks across four implementational dimensions: interaction patterns, semantic properties, model accessibility requirements, and LLM operational roles (see Table 1).

We then further correlate this operational taxonomy with our technical taxonomy to provide more insights. By conducting a correlation analysis between these two taxonomies, we reveal how specific LLM vulnerabilities are related to particular operational jailbreak strategies. This process reveals previously hidden relationships between LLM weaknesses and the practical constraints that restrict the deployment of jailbreak attacks. These insights are valuable for predictive threat modeling and developing corresponding defensive measures. Moreover, by applying our taxonomies to existing works, we identify research gaps in the relevant domains, reveal current LLM safety challenges, and outlook future research directions.

Our contributions are threefold:

- **A comprehensive survey** of existing jailbreak attacks against LLMs, providing an overview of the current landscape of this field.
- **A two-fold taxonomy** for jailbreak attacks, consisting of a technical taxonomy that categorizes attacks by vulnerabilities to reveal underlying weaknesses, and an operational taxonomy assessing real-world feasibility and sustainability through four metrics.
- **An analysis of jailbreak attack trends and research gaps**, highlighting systematic relationships between exploitable LLM vulnerabilities and operational characteristics and outlining open challenges for future works on LLM jailbreak.

2 Background and Survey Methodology

Early Work on Adversarial Prompting. Before the widespread study of jailbreak attacks, researchers developed gradient-based prompt optimization techniques like AutoPrompt (Shin et al., 2020), PEZ (Wen et al., 2023), and GBDA (Guo et al., 2021), demonstrating that discrete text inputs could be optimized to manipulate model outputs. As LLMs became more powerful, these techniques were adapted to circumvent safety measures. Notably, GCG (Greedy Coordinate Gradient) (Zou et al., 2023) has been introduced to generate universal adversarial suffixes for safety-aligned LLMs.

Approach-Centric Taxonomies in LLM Jailbreak Surveys. Existing surveys predominantly organize attacks by methodological approaches—how attacks are constructed and executed—rather than why they succeed. These can be grouped into several organizational paradigms:

Attack Generation Mechanism-Based: These taxonomies focus on how attacks are created and the level of automation involved. Chu et al. (2025b) established a six-category taxonomy across 17 representative attacks according to their generation pipeline. Yi et al. (2024) further subdivide by specific techniques: gradient-based attacks, logits-based attacks, fine-tuning-based attacks, and prompt modification techniques.

Prompt Strategy and Manipulation-Based: Taxonomies in this group analyze the linguistic and semantic strategies employed in jailbreak prompts. Liu et al. (2024d) identify three main strategies from 78 jailbreaks: Pretending, Privilege Escalation, and Attention Shifting. Rossi et al. (2024) categorize prompt injections by their methods: direct and indirect prompt injections. They classify direct prompt injections according to manipulation techniques, including obfuscation, virtualization, etc. Schulhoff et al. (2024) analyze 600K+ prompts from a global competition, providing a comprehensive taxonomical ontology of 29 prompt hacking techniques.

While these taxonomies provide valuable organization of the rapidly growing jailbreak methods, they share a key limitation: categorizing attacks by surface-level characteristics rather than underlying vulnerabilities. This creates critical challenges:

- Defensive efforts remain passive and fragmented. Without a comprehensive understanding of the LLM vulnerabilities, the design of defensive mechanisms must mainly rely on empirical observations (Aguilera-Martínez and Berzal, 2025).
- Resource allocation for safety improvement proceeds without a clear plan according to the risk levels (Cui et al., 2024). This is because the seriousness of a specific vulnerability is hard to evaluate without a comprehensive analysis framework.

- Effective predictive defenses can hardly be built due to the lack of knowledge regarding LLM weaknesses (Zhou et al., 2025).

To address these issues, our survey proposes a two-fold taxonomy that categorizes jailbreak attacks by exploited weaknesses as well as implementation details.

Survey Methodology. We define the scope of this survey to cover jailbreak attacks on LLMs, focusing on methods with clear attack pipelines and empirical validation. Our primary sources include top-tier security conferences (ACM CCS, USENIX Security, NDSS, IEEE S&P), NLP venues (ACL, EMNLP, NAACL), AI/ML venues (ICLR, ICML, NeurIPS, etc.), and arXiv preprints from 2023-2025, as many seminal works (e.g., GCG (Zou et al., 2023), PAIR (Chao et al., 2024)) appeared on arXiv long before formal publication. We searched using keywords including "jailbreak", "adversarial prompts", "LLM safety", "alignment", and "red teaming", supplemented by backward and forward citation tracking from foundational papers. We included papers with clear attack pipelines, complete model specifications (attacker, target, judge), and quantitative experimental validation, while excluding work with vague descriptions, minor variations without novel contributions, or incomplete experimental setups. As a result, we collected 43 jailbreak methods from 36 papers. Detailed survey statistics and distributions are provided in Appendix D.

3 Technical Jailbreak Attack Taxonomy

In this section, we present a novel vulnerability-based taxonomy for jailbreak attacks that shifts the jailbreak attack classification from surface-level attack approaches to the underlying system weaknesses they exploit.

3.1 Taxonomic Framework and Core Principles

We establish a systematic three-tier hierarchical structure in order to investigate the source and nature of the vulnerabilities of the LLM that could be exploited in jailbreak attacks.

Tier 1: Vulnerability Origin Phase. Jailbreak attacks are primarily distinguished according to the specific phase of the LLM lifecycle where the exploitable vulnerabilities exist.

Training-Derived Jailbreak Attacks: These jailbreak attacks target the vulnerabilities embedded during the pre-training and safety alignment phases
Inference-Derived Jailbreak Attacks: These jailbreak attacks target the weaknesses introduced by the characteristics of the generation process.

Tier 2: Vulnerability Type. Jailbreak attacks are further distinguished according to specific exploitable weakness types:

Training phase vulnerabilities originate from problematic data distribution (exploited by out-of-distribution attacks) or learned human-like behavioral patterns (exploited by implicit pattern attacks)
Inference phase vulnerabilities originate from architectural limitations in attention (exploited by attention dilution attacks) and generation (exploited by prefix dependency exploitations) mechanisms

Figure 1 illustrates the four fundamental jailbreak types that emerge from these vulnerabilities.

Tier 3: Exploitation Method. Jailbreak attacks are finally categorized according to the concrete approaches used to exploit each vulnerability type. These approaches range from direct manipulation techniques to learning-based strategies.

One reason for these tiers is that each tier can directly map to different defensive responsibilities and capabilities:

Origin Phase informs *when* to intervene (during training pipeline vs. runtime). This is essential for allocating development and operational resources.
Vulnerability Type determines *what kind* of measures are needed to be taken (architectural changes, data improvements, etc.). It guides researchers to focus on the appropriate solution space
Exploitation Method guides *how* to detect and prevent specific attacks. This allows rapid deployment of specialized defensive measures, but with more solid analysis of the vulnerabilities to guide the process compared to previous works.

3.2 Categorization by Vulnerability Types

Jailbreak attacks exploit two main categories of vulnerabilities, based on their origin in the LLM lifecycle, as detailed below:

Training-Derived Attacks. Training-derived attacks intend to exploit weaknesses embedded in pre-training and safety alignment phases. This category of attacks can be further characterized into two types, namely *Out-of-Distribution (OOD) Attacks* and *Implicit Pattern Attacks*. The former

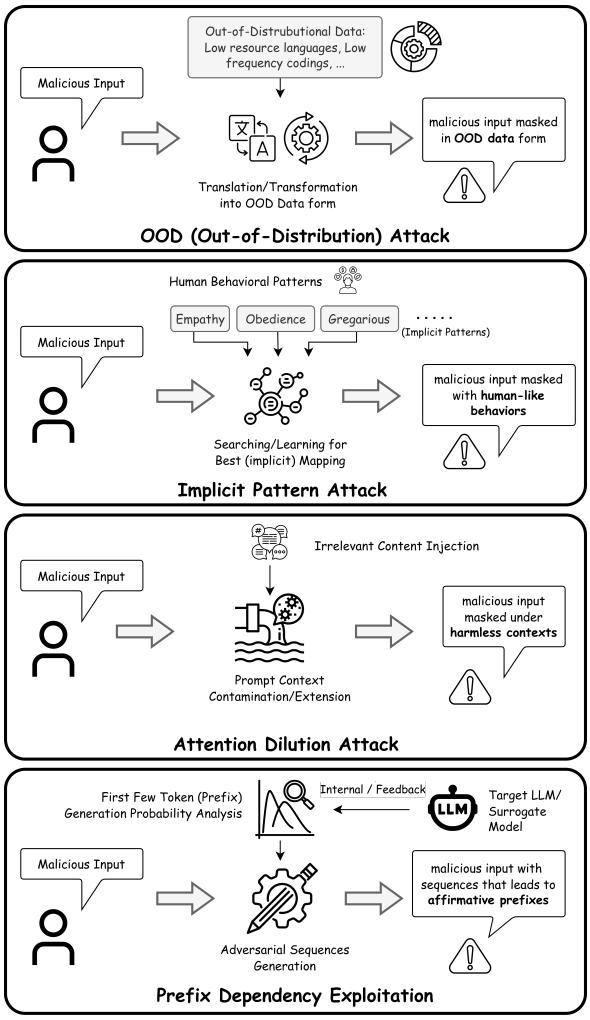

Figure 1: Four fundamental jailbreak types

subcategory leverages distributional imbalance in training data, including *Translation-Based* attacks that target Chinese, Arabic, or some low-resource languages and *Non-Translation-Based* attacks that target other domains with low data distribution. The latter subcategory exploits human behavior patterns learned in the training phase. They could be divided into *Search-based* approaches with runtime optimization and *Learning-based* approaches that employ LLMs for attack generation.

Inference-Derived Attacks. This category of attacks targets architectural limitations of the LLMs during real-time processing. We further categorize this category into two types. The first subcategory, named *Attention Dilution Attacks*, manipulates the transformer's attention mechanisms. These are exploited through *Context-Extension-Based* attacks that exploit the attention decaying phenomenon in long-sequence processing and *Context-Contamination-Based* attacks that embed harmful contents within irrelevant and harmless context. The second subcategory, named *Prefix Dependency Exploitation*, targets the lack of retrospective evaluation ability, a critical weakness of autoregressive models. These attacks could be categorized by the required access level of the manipulation: *White Box Manipulation* requires full parameter (gradients) access, *Gray Box Manipulation* requires partial information like log-probabilities, and *Black Box Manipulation* relies solely on prompt engineering.

More details of our analysis about LLM vulnerability can be found in Appendix B due to space limitations.

3.3 Significance and Impact

This vulnerability-based taxonomy has several advantages over the approach-based classifications:

1. **Analysis of Underlying Mechanisms** Our proposed taxonomy focuses on the weaknesses embedded in the LLMs beyond specific attack approaches. This vulnerability-based analysis can guide the design of defensive strategies more clearly and effectively.

2. **Predictive Defense** Our taxonomy can help the research community predict new attack vectors with a deeper understanding of vulnerability types, and can thus take defensive measures in advance to prevent damage in the real world.

3. **Broad Applicability** Our analysis is based on common characteristics of transformer-based LLM instead of model-specific characteristics. Therefore, our taxonomy provides insights that are applicable across various models and training methods.

4 Operational Jailbreak Attack Taxonomy

4.1 Categorization

The technical dimensions introduced before focus on the vulnerabilities exploited by jailbreak attacks. However, most research works remain at a theoretical level: they usually only consider the attack effectiveness on a few models. Besides, since most benchmarks do not establish a standard testing configuration for jailbreak attacks, it is difficult to evaluate how well these attacks perform in a realistic environment.

Table 1: Categorization of existing jailbreak attacks based on our proposed technical and operational dimensions

Attack Methods	Technical Taxonomy (Derivation)									Operational Taxonomy (Interaction & Access)										
	Training Derived				Inference Derived					Interaction Method			Semantic		Model Accessibility				LLM Role	
	OOD Attack		Implicit Pattern		Attn. Dilution		Prefix Dependency								Attacker		Target			
	Trans	Other	Search	Learn	Ext	Contam	White	Gray	Black	One-Shot	Iter-ative	Para-llel	Natu-ral	Pert-urbed	Open	Close	Open	Close	Gene-rator	Eval-uator
ReNeLLM (Ding et al., 2024)	●	○	●	○	●	●	○	○	○	○	●	○	●	○	●	●	●	●	●	●
MultiJail (Deng et al., 2024b)	●	○	●	○	○	○	○	○	○	●	○	○	●	○	●	●	●	●	●	○
DRA (Liu et al., 2024b)	○	●	●	○	●	●	○	○	●	○	●	○	●	○	○	○	●	●	○	○
Artprompt (Jiang et al., 2024)	○	●	○	○	○	○	○	○	○	●	○	○	○	●	○	○	●	●	○	○
Codechameleon (Lv et al., 2024)	○	●	○	○	●	●	○	○	○	●	○	○	●	○	○	○	●	●	○	○
AutoDAN (Liu et al., 2024c)	○	○	●	○	●	●	●	○	○	○	●	○	●	○	○	●	●	○	●	○
AutoDAN (transfer) (Liu et al., 2024c)	○	○	●	○	●	●	○	○	●	●	○	○	●	○	●	●	●	●	●	●
GPTFUZZER (Yu et al., 2024)	○	○	●	○	●	●	○	○	○	○	●	○	●	○	○	●	●	●	●	○
PAIR (Chao et al., 2024)	○	○	●	○	●	○	○	○	○	○	●	●	●	○	●	●	●	●	●	●
SMJ (Li et al., 2024b)	○	○	●	○	○	○	○	○	○	●	○	○	●	○	●	○	●	○	○	●
Geneshift (Wu et al., 2025)	○	○	●	○	●	●	○	○	○	○	●	○	●	○	●	●	○	●	●	●
Rainbowteaming (Samvelyan et al., 2024)	○	○	●	○	○	○	○	○	○	○	●	○	●	○	●	○	●	○	●	●
Masterkey (Deng et al., 2024a)	○	○	○	●	●	●	○	○	○	●	○	○	●	○	●	●	○	●	●	○
ICA (Wei et al., 2024)	○	○	○	●	●	○	○	○	○	●	○	○	●	○	○	○	●	●	○	○
Arrattack (Li et al., 2025)	○	○	●	●	○	○	○	○	○	○	●	○	●	○	●	○	●	●	●	○
Rlbreaker (Chen et al., 2025)	○	○	●	●	○	○	○	○	○	○	●	○	●	○	○	●	●	●	●	○
Advprompter (ASR@1) (Paulus et al., 2025)	○	○	○	●	○	○	○	○	○	○	○	○	●	○	●	○	●	●	●	○
Advprompter (ASR@10) (Paulus et al., 2025)	○	○	●	●	○	○	○	○	○	○	●	○	●	○	●	○	●	●	●	○
MSJ (Anil et al., 2024)	●	●	○	●	●	○	○	○	○	●	○	○	●	○	●	○	●	●	●	○
Mathprompt (Bethany et al., 2024)	○	○	○	○	●	●	○	○	○	●	○	○	●	○	○	●	●	●	●	○
Puzzler (Chang et al., 2024)	○	○	○	○	●	●	○	○	○	●	○	○	●	○	○	●	●	●	●	●
DeepInception (Li et al., 2024c)	○	○	○	○	●	●	○	○	○	●	○	○	●	○	○	○	●	●	○	○
AutoInception (Li et al., 2024c)	○	○	○	○	●	●	○	○	○	○	●	○	●	○	○	●	●	●	●	○
DAP (Xiao et al., 2024)	○	○	●	○	●	○	○	○	●	○	●	○	●	○	●	○	●	●	●	●
Persona Modulation (Shah et al., 2023)	○	○	●	○	●	●	○	○	○	○	○	●	●	○	○	●	●	●	●	○
GUARD (Jin et al., 2025)	○	○	●	○	●	●	○	○	○	○	●	○	●	○	●	●	●	●	●	●
Evil genius (Tian et al., 2024)	○	○	●	○	●	●	○	○	○	○	●	○	●	○	○	●	○	●	●	●
CipherChat (Yuan et al., 2024)	○	●	○	○	●	●	○	○	○	●	○	○	○	●	○	○	○	●	○	○
SelfCipher (Yuan et al., 2024)	○	○	○	○	●	○	○	○	○	●	○	○	●	○	○	○	○	●	○	○
FLIPattack (Liu et al., 2024f)	○	●	○	○	●	●	○	○	○	●	○	○	○	●	○	○	●	●	○	○
I-SFJ (Zheng et al., 2024)	○	●	●	●	●	○	○	●	○	○	●	●	●	○	○	○	●	○	○	○
PAP(broad scan) (Zeng et al., 2024)	○	○	●	●	●	●	○	○	○	○	○	●	●	○	○	●	●	●	●	○
PAP(iterative) (Zeng et al., 2024)	○	○	●	●	●	●	○	○	○	○	●	○	●	○	○	●	●	●	●	○
GCG (Zou et al., 2023)	○	○	●	○	○	○	●	○	○	○	●	○	○	●	●	○	●	○	●	○
AmpleGCG (Liao and Sun, 2024)	○	○	●	●	○	○	○	○	●	○	○	●	○	●	●	○	●	○	●	○
Faster-GCG (Li et al., 2024a)	○	○	●	○	○	○	●	○	○	○	●	●	○	●	●	○	●	○	●	○
COLD-Attack (Guo et al., 2024a)	○	○	●	○	○	○	●	○	○	○	●	○	○	●	○	○	●	○	○	○
Adaptive attack (Andriushchenko et al., 2025)	○	○	●	○	●	●	●	●	○	○	●	○	○	●	○	○	●	●	○	○
Fun-tuning (Labunets et al., 2025)	○	○	●	○	○	○	○	●	○	○	●	○	●	○	○	○	○	●	○	○
H-CoT (Kuo et al., 2025)	○	○	○	●	○	○	○	○	●	○	●	○	●	○	○	●	○	●	●	○
DIA-I (Meng et al., 2025)	○	○	●	○	○	○	○	○	●	●	●	○	●	○	●	●	●	●	○	○
DIA-II (Meng et al., 2025)	○	○	●	○	●	○	○	○	●	●	●	○	●	○	○	○	●	●	○	○

Note: This unified taxonomy categorizes attacks by: **1. Technical Derivation** (left), including **Training Derived** (OOD, Implicit Patterns) and **Inference Derived** (Attention Dilution, Prefix Dependency); and **2. Operational Characteristics** (right), covering **Interaction Method** (One-Shot, Iterative, Parallel), **Semantic** nature (Natural vs. Perturbed), **Model Accessibility** (Attacker/Target knowledge), and the **LLM Role** (Generator vs. Evaluator). Filled circles indicate the presence of a characteristic; empty circles indicate absence.

To address this issue, we present a four-dimensional operational jailbreak attack taxonomy that focuses on the attack execution to complement our technical taxonomy.

Interaction Method where one-shot methods minimize detection risk through single queries, iterative methods refine attacks across multiple rounds, and parallel methods deploy multiple non-coordinating agents.

Semantic where natural prompts maintain natural language structure against evolving detection systems, while perturbed prompts with nonsensical sequences may succeed in an explainable way, but remain vulnerable to modern safety mechanisms.

Model Accessibility where open-source dependencies enhance reproducibility while closed-source dependencies have potential access restriction risks.

LLM Role where LLM generators create attack prompts but may be vulnerable to safety alignment updates, while LLM evaluators judge harmfulness and offer greater robustness against alignment changes.

More details of our analysis about operational characteristics of LLM jailbreak attacks can be found in Appendix C due to space limitations.

4.2 Operational Characteristics

The categorization supports broader analysis by capturing two key characteristics of attack methods for assessing effectiveness across scenarios.

4.2.1 Sustainability

This characteristic can predict if the attack methods can maintain effectiveness in the long run, as the defensive mechanisms evolve. All four operational dimensions provide evidence for attack sustainability evaluation. Among all interaction methods, one-shot method is the most robust against detection mechanisms, since it requires much fewer query numbers than other methods. Regarding semantic, perturbed prompts face higher risks of being detected compared with natural ones. Model accessibility reveals that closed-source model reliance brings potential instability of the attack in the future since the service provider may change their strategies over time. LLM Role shows that LLM generator-based attack may face the risk of future model refusing to generate harmful content due to more advanced safety alignment mechanisms.

4.2.2 Feasibility

This characteristic shows if the attack methods are viable under real-world operational constraints. The four dimensions also support the feasibility analysis of the attack methods. Specifically, the interaction methods indicate the trade-off between efficiency and computational effort. Semantic dimension shows that perturbed prompts would easily get filtered by the complicated defensive mechanisms of the commercial LLM than in theoretical experiments. Model accessibility shows that attacks that rely on closed-source systems may face various restrictions (e.g. rate limitations and model access control), undermining their application in real-world situations. LLM Role shows that an LLM generator-based attack method may face stability issues that may reduce the attack efficiency.

5 Key Insights

Our correlation analysis in Figure 2 reveals critical patterns in jailbreak attack mechanisms. We calculated Pearson's correlation coefficients (ranging from -1 to 1) from binary occurrence data, where each attack's characteristics are marked as present (1) or absent (0). Positive values indicate characteristics that co-occur, negative values indicate mutual exclusivity, and values near zero indicate independence.

5.1 Interaction Requirements

The correlation analysis has shown that some attack types exhibit strong preferences for the way of interaction. Search-based implicit pattern attacks have a strong correlation with iterative interaction ($r = 0.61$), while showing a weaker correlation with parallel interaction ($r = 0.29$). This suggests that these attacks rely more on iterative interaction than parallel methods.

This could have been caused by the real-world restrictions. Although parallel searching across a much larger attack vector space provides a higher possibility of jailbreak success compared to iterative methods, it requires much greater computational resources and may encounter the API access rate limitations. The ideal solution is the combination of parallel and iterative interaction, as demonstrated by PAIR (Chao et al., 2024). By setting a shallow searching depth and moderate search space (typically defined by the number of agents in practice), implicit pattern attacks can achieve a high ASR (Attack Success Rate) with moderate com-

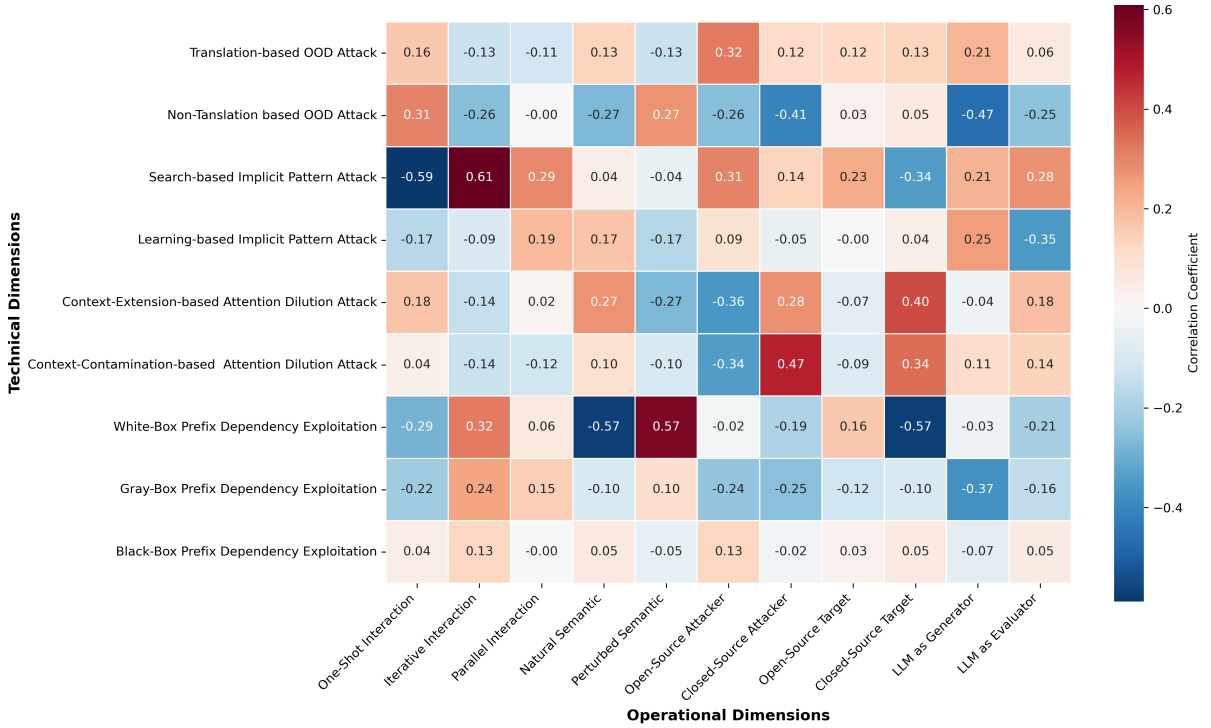

Figure 2: Correlation matrix between technical and operational dimensions

putational resource consumption and avoid being detected due to low conversation iterations.

5.2 Targeting Preferences

The correlation analysis reveals explicit differences in the targeting preferences of the attacks. Both subcategories of attention dilution attacks target the closed-source targets ($r = 0.4$ and $r = 0.34$, respectively), while search-based implicit pattern attacks and white-box prefix dependency exploitation circumvent to avoid closed-source targets ($r = -0.34$ and $r = -0.57$, respectively).

This disparity can be caused by the intrinsic nature of these attacks. Attention dilution attacks misdirect the target LLMs with irrelevant information, but still require the models to allocate enough attention to the malicious intention embedded in the input prompts. This mechanism asks for strong instruction-following ability in long sequence processing scenarios, which is relatively stronger in closed-source models compared to open-source models, and eventually leads to a preference for closed-source targets.

On the other hand, search-based implicit pattern attacks and white-box prefix dependency exploitation both rely heavily on sufficient data to guide the attack prompt optimization. Closed-source models provide limited data (usually no more than model responses in practice) that is insufficient for effective optimization, and eventually result in the avoidance of these targets by these attack types.

5.3 Convergent Patterns in Attention Dilution Attacks

As mentioned before, both subcategories of attention dilution attacks show a positive correlation with closed-source targets. Moreover, they also share a similar preference for closed-source attacker ($r = 0.27$ and 0.28, respectively).

The greatest difference between them exists in their semantical characteristics. Context-extension-based attacks rely relatively more on natural prompts ($r = 0.27$), while context-contamination-based attacks do not show any meaningful preference between natural and perturbed prompts ($r = 0.09$ and $r = -0.09$, respectively).

This is because context-extension-based attacks require a longer input sequence compared to contamination-based attacks. When context extension techniques are applied solely, attackers tend to use natural language extensions, because long sequences of nonsensical characters severely reduce the stealthiness of the attacks. Meanwhile, context contamination attacks can not only involve inserting special tokens into the prompts, resulting in perturbed prompts, but can also simply embed the

harmful content in a harmless scenario with natural language.

5.4 Black-Box Manipulation Isolation

Black-box prefix dependency exploitation shows low correlation with all operational dimensions ($|r| < 0.13$), making it different from other attack types that show a preference for specific operational patterns. The lack of correlation highlights its strong implementational flexibility and its independence from specific operational constraints.

This characteristic is primarily due to the attack's low data dependency, relying solely on the inputs and outputs of the LLMs. Unlike white-box manipulation that prefers perturbed prompts for reliability ($r = 0.57$), or search-based attacks that typically require iterative interaction ($r = 0.61$), black-box prefix dependency exploitation adapts well to the operational constraints discussed in the paper.

Another contributing factor is that this jailbreak attack category is relatively novel, and a mainstream operational paradigm has not yet formed. The diversity within this category can also lead to a low preference for specific operational patterns.

6 Future Directions

Through our analysis, we identify several critical research gaps that remain unexplored or insufficiently addressed, presenting both challenges and opportunities for future investigation.

6.1 Expanding Attack Surface Coverage

Out-of-Distribution Attack Vectors Exploration. Existing OOD jailbreak attacks mostly rely on cross-language translation and code transformation, leaving various other distributional imbalances unexplored, such as domain-specific jargon or specialized notation systems. Furthermore, the potential for multiple OOD attacks to collaborate in a single attack prompt remains largely unexplored. Identifying exploitable OOD attack vectors and investigating their collaborative effects forms a new research frontier.

Service Provider Function Exploitation. Gray-box jailbreak attacks typically require log-probability data. As service provider ecosystems expand, more functions that provide token generation information may emerge. While fine-tuning functions have already been exploited, many functional weaknesses remain undiscovered. For example, LLM service providers may provide metadata such as retrieval confidence scores for Retrieval-Augmented Generation (RAG) systems or execution probability rankings for agentic systems. Such data can become effective sources for gray-box attacks, necessitating rigorous security assessment of these functions before widespread deployment.

6.2 Systematizing Attack Mechanisms

Implicit Pattern Specification and Optimization. Existing research has identified some exploitable LLM behavioral patterns, such as role-playing, persuasive techniques, and authority endorsement. However, the taxonomy of exploitable behavioral patterns embedded in LLM training remains incomplete. Multiple assumed effective manipulation techniques, like emotional manipulation and social engineering, still need to be specified and developed into concrete and viable attack paradigms. Moreover, mapping between specific attack contexts and optimal pattern selection lacks systematic methodology, forming another research frontier.

Clarifying Context Manipulation Mechanisms. The concrete mechanisms and effective boundaries of context contamination and context extension need further clarification. Existing jailbreak methods often deploy both attacks simultaneously, making it difficult to determine the effectiveness of each attack type in isolation. The optimal contamination-to-extension ratios, the threshold lengths where extension effects dominate over contamination effects, and the interaction mechanisms between them require further investigation to provide deeper insights into attention dilution attacks.

7 Conclusion

This work surveys 43 LLM jailbreak methods from 36 papers, analyzing them through a vulnerability-centric framework that examines both exploitation mechanisms and operational characteristics. Our correlation analysis reveals systematic patterns between attack types and operational constraints that were previously obscured by approach-centric categorizations. By synthesizing current research, we identify critical gaps in attack surface coverage, mechanism systematization, and the effectiveness-stealthiness trade-off, providing practical guidance for developing more principled offensive and defensive techniques.

Ethical Consideration

This work addresses a critical security challenge in LLMs while acknowledging the sensitive nature of jailbreak attack research. We adhere to the following ethical principles in conducting this research:

Responsible Disclosure. Our taxonomy is designed to enhance defensive capabilities rather than facilitate malicious exploits. Through systematic categorization of vulnerabilities, we seek to empower developers and researchers to proactively strengthen LLM safety. We deliberately focus on vulnerability taxonomy and attack pattern analysis while avoiding detailed implementation guides that could enable jailbreak attacks.

Broader Impact. This research advances the safe deployment of LLMs in sensitive domains. By identifying system vulnerabilities and research gaps, we provide actionable insights from both technical and operational perspectives to support the development of more robust safety mechanisms that benefit society.

Limitations

Lack of Standardized Benchmarking

Currently, LLM jailbreaking lacks standardized testing configurations. Different researchers have deployed various experimental setups, involving different target models, resource and access level requirements, and evaluation criteria for attack success. These discrepancies make direct experimental comparisons difficult and may result in misleading conclusions. Furthermore, attempting to reproduce various attack methods under a unified condition would be methodologically questionable, as each attack was originally designed and optimized for a specific context. Given these challenges, we focus on the construction of a comprehensive taxonomy rather than conducting large-scale experiments.

Limited Coverage of Defensive Mechanisms

Although our taxonomy identifies vulnerabilities that defensive mechanisms need to address, we do not comprehensively survey existing defenses. This is because existing defensive measures are highly fragmented, ranging from input filtering and output monitoring to safety alignment approaches. Comprehensively surveying these diverse methods would require a separate taxonomy framework. Furthermore, the opacity of commercial defenses—where many state-of-the-art mechanisms remain proprietary—makes systematic categorization challenging without access to implementation details. Instead, we provide a vulnerability-centric jailbreak attack taxonomy that can serve as a foundation for future defensive taxonomy design.

Limited Quantitative Validation

Our correlation analysis is based on binary occurrence data rather than quantitative metrics. This is because developing such quantitative criteria faces great challenges in measurement. There exist no standards or benchmarks to assess the degree or proportion of each exploitation mechanism. For instance, how should we assess what percentage of an attack's success derives from implicit pattern exploitation versus OOD techniques when both are present? Or how do we quantify the relative contribution of attention dilution versus prefix dependency in a hybrid attack? Therefore, while more granular measurements might provide additional insights, the categorical approach we employ offers practical clarity for understanding attack mechanisms and their relationships to operational constraints without requiring potentially arbitrary quantitative weightings.

Generalizability Across Model Architectures

Our taxonomy is built primarily around transformer-based autoregressive LLMs, which dominate current research and deployment. However, emerging architectures such as state-space models (e.g., Mamba) or future paradigms may introduce novel vulnerabilities not captured by our framework. Although we believe our lifecycle-based approach (training-derived vs. inference-derived) provides some generalizability, attacks that target architecture-specific features may require framework extensions.

Acknowledgement of AI Usage

In this work, we utilized AI language models for writing assistance. Specifically, we used Anthropic Claude Sonnet 4.5 and Google's Gemini 2.0 to polish language, improve sentence clarity, and enhance readability. These tools were used solely for linguistic refinement; all conceptual development, taxonomic frameworks, correlation analyses, literature review, and scientific insights presented in this work are the original intellectual contributions of the authors.

References

Francisco Aguilera-Martínez and Fernando Berzal. 2025. Llm security: Vulnerabilities, attacks, defenses, and countermeasures. *Preprint*, arXiv:2505.01177.

Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. 2025. Jailbreaking leading safety-aligned llms with simple adaptive attacks. *Preprint*, arXiv:2404.02151.

Cem Anil, Esin DURMUS, Nina Rimsky, Mrinank Sharma, Joe Benton, Sandipan Kundu, Joshua Batson, Meg Tong, Jesse Mu, Daniel J Ford, Francesco Mosconi, Rajashree Agrawal, Rylan Schaeffer, Naomi Bashkansky, Samuel Svenningsen, Mike Lambert, Ansh Radhakrishnan, Carson Denison, Evan J Hubinger, and 15 others. 2024. Many-shot jailbreaking. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *Preprint*, arXiv:2004.05150.

Emet Bethany, Mazal Bethany, Juan Arturo Nolazco Flores, Sumit Kumar Jha, and Peyman Najafirad. 2024. Jailbreaking large language models with symbolic mathematics. *Preprint*, arXiv:2409.11445.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. 2021. Extracting training data from large language models. *Preprint*, arXiv:2012.07805.

Zhiyuan Chang, Mingyang Li, Yi Liu, Junjie Wang, Qing Wang, and Yang Liu. 2024. Play guessing game with llm: Indirect jailbreak attack with implicit clues. *Preprint*, arXiv:2402.09091.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. 2024. Jailbreaking black box large language models in twenty queries. *Preprint*, arXiv:2310.08419.

Xuan Chen, Yuzhou Nie, Wenbo Guo, and Xiangyu Zhang. 2025. When llm meets drl: Advancing jailbreaking efficiency via drl-guided search. *Preprint*, arXiv:2406.08705.

Zhiyu Zoey Chen, Jing Ma, Xinlu Zhang, Nan Hao, An Yan, Armineh Nourbakhsh, Xianjun Yang, Julian McAuley, Linda Petzold, and William Yang Wang. 2024. A survey on large language models for critical societal domains: Finance, healthcare, and law. *Preprint*, arXiv:2405.01769.

Junjie Chu, Yugeng Liu, Ziqing Yang, Xinyue Shen, Michael Backes, and Yang Zhang. 2025a. Jailbreakradar: Comprehensive assessment of jailbreak attacks against llms. *Preprint*, arXiv:2402.05668.

Junjie Chu, Yugeng Liu, Ziqing Yang, Xinyue Shen, Michael Backes, and Yang Zhang. 2025b. Jailbreakradar: Comprehensive assessment of jailbreak attacks against llms. *Preprint*, arXiv:2402.05668.

Tianyu Cui, Yanling Wang, Chuanpu Fu, Yong Xiao, Sijia Li, Xinhao Deng, Yunpeng Liu, Qinglin Zhang, Ziyi Qiu, Peiyang Li, Zhixing Tan, Junwu Xiong, Xinyu Kong, Zujie Wen, Ke Xu, and Qi Li. 2024. Risk taxonomy, mitigation, and assessment benchmarks of large language model systems. *Preprint*, arXiv:2401.05778.

Ziyan Cui, Ning Li, and Huaikang Zhou. 2025. Can large language models replace human subjects? a large-scale replication of scenario-based experiments in psychology and management. *Preprint*, arXiv:2409.00128.

Gelei Deng, Yi Liu, Yuekang Li, Kailong Wang, Ying Zhang, Zefeng Li, Haoyu Wang, Tianwei Zhang, and Yang Liu. 2024a. Masterkey: Automated jailbreaking of large language model chatbots. In *Proceedings 2024 Network and Distributed System Security Symposium*. Internet Society.

Yue Deng, Wenxuan Zhang, Sinno Jialin Pan, and Lidong Bing. 2024b. Multilingual jailbreak challenges in large language models. *Preprint*, arXiv:2310.06474.

Peng Ding, Jun Kuang, Dan Ma, Xuezhi Cao, Yunsen Xian, Jiajun Chen, and Shujian Huang. 2024. A wolf in sheep's clothing: Generalized nested jailbreak prompts can fool large language models easily. *Preprint*, arXiv:2311.08268.

Matteo Esposito, Francesco Palagiano, Valentina Lenarduzzi, and Davide Taibi. 2025. On large language models in mission-critical it governance: Are we ready yet? *Preprint*, arXiv:2412.11698.

Lang Gao, Jiahui Geng, Xiangliang Zhang, Preslav Nakov, and Xiuying Chen. 2025. Shaping the safety boundaries: Understanding and defending against jailbreaks in large language models. *Preprint*, arXiv:2412.17034.

Chuan Guo, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela. 2021. Gradient-based adversarial attacks against text transformers. *Preprint*, arXiv:2104.13733.

Xingang Guo, Fangxu Yu, Huan Zhang, Lianhui Qin, and Bin Hu. 2024a. Cold-attack: Jailbreaking llms with stealthiness and controllability. *Preprint*, arXiv:2402.08679.

Yufei Guo, Muzhe Guo, Juntao Su, Zhou Yang, Mengqiu Zhu, Hongfei Li, Mengyang Qiu, and Shuo Shuo Liu. 2024b. Bias in large language models: Origin, evaluation, and mitigation. *Preprint*, arXiv:2411.10915.

Zeqing He, Zhibo Wang, Zhixuan Chu, Huiyu Xu, Wenhui Zhang, Qinglong Wang, and Rui Zheng. 2025. Jailbreaklens: Interpreting jailbreak mechanism in the lens of representation and circuit. *Preprint*, arXiv:2411.11114.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065, Toronto, Canada. Association for Computational Linguistics.

Fengqing Jiang, Zhangchen Xu, Luyao Niu, Zhen Xiang, Bhaskar Ramasubramanian, Bo Li, and Radha Poovendran. 2024. Artprompt: Ascii art-based jailbreak attacks against aligned llms. *Preprint*, arXiv:2402.11753.

Haibo Jin, Ruoxi Chen, Peiyan Zhang, Andy Zhou, Yang Zhang, and Haohan Wang. 2025. Guard: Role-playing to generate natural-language jailbreakings to test guideline adherence of large language models. *Preprint*, arXiv:2402.03299.

Haibo Jin, Leyang Hu, Xinuo Li, Peiyan Zhang, Chonghan Chen, Jun Zhuang, and Haohan Wang. 2024. Jailbreakzoo: Survey, landscapes, and horizons in jailbreaking large language and vision-language models. *Preprint*, arXiv:2407.01599.

Martin Kuo, Jianyi Zhang, Aolin Ding, Qinsi Wang, Louis DiValentin, Yujia Bao, Wei Wei, Hai Li, and Yiran Chen. 2025. H-cot: Hijacking the chain-of-thought safety reasoning mechanism to jailbreak large reasoning models, including openai o1/o3, deepseek-r1, and gemini 2.0 flash thinking. *Preprint*, arXiv:2502.12893.

Andrey Labunets, Nishit V. Pandya, Ashish Hooda, Xiaohan Fu, and Earlence Fernandes. 2025. Fun-tuning: Characterizing the vulnerability of proprietary llms to optimization-based prompt injection attacks via the fine-tuning interface. In *2025 IEEE Symposium on Security and Privacy (SP)*, page 411–429. IEEE.

Linbao Li, Yannan Liu, Daojing He, and Yu Li. 2025. One model transfer to all: On robust jailbreak prompts generation against llms. *Preprint*, arXiv:2505.17598.

Xiao Li, Zhuhong Li, Qiongxiu Li, Bingze Lee, Jinghao Cui, and Xiaolin Hu. 2024a. Faster-gcg: Efficient discrete optimization jailbreak attacks against aligned large language models. *Preprint*, arXiv:2410.15362.

Xiaoxia Li, Siyuan Liang, Jiyi Zhang, Han Fang, Aishan Liu, and Ee-Chien Chang. 2024b. Semantic mirror jailbreak: Genetic algorithm based jailbreak prompts against open-source llms. *Preprint*, arXiv:2402.14872.

Xuan Li, Zhanke Zhou, Jianing Zhu, Jiangchao Yao, Tongliang Liu, and Bo Han. 2024c. Deepinception: Hypnotize large language model to be jailbreaker. *Preprint*, arXiv:2311.03191.

Zeyi Liao and Huan Sun. 2024. Amplegcg: Learning a universal and transferable generative model of adversarial suffixes for jailbreaking both open and closed llms. *Preprint*, arXiv:2404.07921.

Yuping Lin, Pengfei He, Han Xu, Yue Xing, Makoto Yamada, Hui Liu, and Jiliang Tang. 2024. Towards understanding jailbreak attacks in llms: A representation space analysis. *Preprint*, arXiv:2406.10794.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Tong Liu, Yingjie Zhang, Zhe Zhao, Yinpeng Dong, Guozhu Meng, and Kai Chen. 2024b. Making them ask and answer: Jailbreaking large language models in few queries via disguise and reconstruction. *Preprint*, arXiv:2402.18104.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. 2024c. Autodan: Generating stealthy jailbreak prompts on aligned large language models. *Preprint*, arXiv:2310.04451.

Yi Liu, Gelei Deng, Zhengzi Xu, Yuekang Li, Yaowen Zheng, Ying Zhang, Lida Zhao, Tianwei Zhang, and Kailong Wang. 2024d. A hitchhiker's guide to jailbreaking chatgpt via prompt engineering. *Proceedings of the 4th International Workshop on Software Engineering and AI for Data Quality in Cyber-Physical Systems/Internet of Things*.

Yi Liu, Gelei Deng, Zhengzi Xu, Yuekang Li, Yaowen Zheng, Ying Zhang, Lida Zhao, Tianwei Zhang, Kailong Wang, and Yang Liu. 2024e. Jailbreaking chatgpt via prompt engineering: An empirical study. *Preprint*, arXiv:2305.13860.

Yue Liu, Xiaoxin He, Miao Xiong, Jinlan Fu, Shumin Deng, and Bryan Hooi. 2024f. Flipattack: Jailbreak llms via flipping. *Preprint*, arXiv:2410.02832.

Huijie Lv, Xiao Wang, Yuansen Zhang, Caishuang Huang, Shihan Dou, Junjie Ye, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Codechameleon: Personalized encryption framework for jailbreaking large language models. *Preprint*, arXiv:2402.16717.

Xingjun Ma, Yifeng Gao, Yixu Wang, Ruofan Wang, Xin Wang, Ye Sun, Yifan Ding, Hengyuan Xu, Yunhao Chen, Yunhan Zhao, Hanxun Huang, Yige Li, Yutao Wu, Jiaming Zhang, Xiang Zheng, Yang Bai, Zuxuan Wu, Xipeng Qiu, Jingfeng Zhang, and 29 others. 2025. Safety at scale: A comprehensive survey of large model and agent safety. *Preprint*, arXiv:2502.05206.

Wenlong Meng, Fan Zhang, Wendao Yao, Zhenyuan Guo, Yuwei Li, Chengkun Wei, and Wenzhi Chen. 2025. Dialogue injection attack: Jailbreaking llms through context manipulation. *Preprint*, arXiv:2503.08195.

Anselm Paulus, Arman Zharmagambetov, Chuan Guo, Brandon Amos, and Yuandong Tian. 2025. Advprompter: Fast adaptive adversarial prompting for llms. *Preprint*, arXiv:2404.16873.

Matthew Renze and Erhan Guven. 2024. The benefits of a concise chain of thought on problem-solving in large language models. In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, page 476–483. IEEE.

Sippo Rossi, Alisia Marianne Michel, Raghava Rao Mukkamala, and Jason Bennett Thatcher. 2024. An early categorization of prompt injection attacks on large language models. *Preprint*, arXiv:2402.00898.

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

Sander Schulhoff, Jeremy Pinto, Anaum Khan, Louis-François Bouchard, Chenglei Si, Svetlina Anati, Valen Tagliabue, Anson Liu Kost, Christopher Carnahan, and Jordan Boyd-Graber. 2024. Ignore this title and hackaprompt: Exposing systemic vulnerabilities of llms through a global scale prompt hacking competition. *Preprint*, arXiv:2311.16119.

Rusheb Shah, Quentin Feuillade-Montixi, Soroush Pour, Arush Tagade, Stephen Casper, and Javier Rando. 2023. Scalable and transferable black-box jailbreaks for language models via persona modulation. *Preprint*, arXiv:2311.03348.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. *Preprint*, arXiv:2302.00093.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Yu Tian, Xiao Yang, Jingyuan Zhang, Yinpeng Dong, and Hang Su. 2024. Evil geniuses: Delving into the safety of llm-based agents. *Preprint*, arXiv:2311.11855.

Zeming Wei, Yifei Wang, Ang Li, Yichuan Mo, and Yisen Wang. 2024. Jailbreak and guard aligned language models with only few in-context demonstrations. *Preprint*, arXiv:2310.06387.

Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2023. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. *Preprint*, arXiv:2302.03668.

Tianyi Wu, Zhiwei Xue, Yue Liu, Jiaheng Zhang, Bryan Hooi, and See-Kiong Ng. 2025. Geneshift: Impact of different scenario shift on jailbreaking llm. *Preprint*, arXiv:2504.08104.

Zeguan Xiao, Yan Yang, Guanhua Chen, and Yun Chen. 2024. Distract large language models for automatic jailbreak attack. *Preprint*, arXiv:2403.08424.

Zihao Xu, Yi Liu, Gelei Deng, Yuekang Li, and Stjepan Picek. 2024. A comprehensive study of jailbreak attack versus defense for large language models. *Preprint*, arXiv:2402.13457.

Lixiang Yan, Lele Sha, Linxuan Zhao, Yuheng Li, Roberto Martinez-Maldonado, Guanliang Chen, Xinyu Li, Yueqiao Jin, and Dragan Gašević. 2023. Practical and ethical challenges of large language models in education: A systematic scoping review. *British Journal of Educational Technology*, 55(1):90–112.

Sibo Yi, Yule Liu, Zhen Sun, Tianshuo Cong, Xinlei He, Jiaxing Song, Ke Xu, and Qi Li. 2024. Jailbreak attacks and defenses against large language models: A survey. *Preprint*, arXiv:2407.04295.

Jiahao Yu, Xingwei Lin, Zheng Yu, and Xinyu Xing. 2024. Gptfuzzer: Red teaming large language models with auto-generated jailbreak prompts. *Preprint*, arXiv:2309.10253.

Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen tse Huang, Pinjia He, Shuming Shi, and Zhaopeng Tu. 2024. Gpt-4 is too smart to be safe: Stealthy chat with llms via cipher. *Preprint*, arXiv:2308.06463.

Yi Zeng, Hongpeng Lin, Jingwen Zhang, Diyi Yang, Ruoxi Jia, and Weiyan Shi. 2024. How johnny can persuade llms to jailbreak them: Rethinking persuasion to challenge ai safety by humanizing llms. *Preprint*, arXiv:2401.06373.

Xiaosen Zheng, Tianyu Pang, Chao Du, Qian Liu, Jing Jiang, and Min Lin. 2024. Improved few-shot jailbreaking can circumvent aligned language models and their defenses. *Preprint*, arXiv:2406.01288.

Yuyang Zhou, Guang Cheng, Kang Du, Zihan Chen, and Yuyu Zhao. 2025. Toward intelligent and secure cloud: Large language model empowered proactive defense. *Preprint*, arXiv:2412.21051.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *Preprint*, arXiv:2307.15043.

A Jailbreak Formalization

To formalize the process of LLM jailbreak, we first have to identify the elements involved. Let x denote the initial harmful prompt that would typically

be rejected by the LLM, and let $f(x)$ be a transformation function that converts the original prompt into a modified version x'.

$$x' = f(x) \tag{1}$$

Let LLM represent the target language model, and y be the generated response to the manipulated prompt.

$$y = \text{model}(x') \tag{2}$$

Let JUDGE be an evaluation function that returns True if the response y violates the safety objective G (indicating harmful content generation), and False otherwise.

$$R = \text{JUDGE}(y, G) | R \in \{True, False\} \tag{3}$$

This means that a jailbreak attack is considered successful if the protective mechanisms of the target LLM are circumvented (i.e., the target LLM and its defensive mechanisms fail to identify the malicious intention of the prompt), and harmful content is generated (i.e., the result of JUDGE is True).

B Jailbreak Analysis according to Technical Dimensions

B.1 Training-derived attacks

Jailbreak attacks that fall into this category all target the vulnerabilities embedded in the training phase, i.e., the pre-training and safety alignment phase. These attacks can be assigned to the following types:

B.1.1 Out-of-Distribution(OOD) Attack

Distributional imbalances across multiple dimensions in the training datasets for LLMs have been observed (Guo et al., 2024b). These distributional imbalances result in insufficiency in the model's safety alignment in various domains and contexts. As a result, models exhibit reduced safety awareness in these scenarios, and this vulnerability resulted in the success of Out-of-Distribution (OOD) attacks.

We define the out-of-distribution (OOD) attack function as:

$$f_{OOD}(x) = O^*(x)$$
$$\text{s.t. } O^* = \arg\max_{O \in \mathcal{O}} \mathcal{L}\left(\text{model}(O(x)), y_{\text{harmful}}\right) \cdot (1 - P_Q(O(x), \delta)) \tag{4}$$

- x represents the original harmful request
- $O \in \mathcal{O}$ represents transformation functions from the OOD manipulation space that modify input to maximize distributional divergence
- $y_{harmful}$ denotes the target harmful output
- $\mathcal{L}(\text{model}(O(x)), y_{harmful})$ measures the model's tendency to generate harmful content when given transformed input $O(x)$
- $P_Q(O(x), \delta)$ denotes the probability density function of the training distribution Q evaluated at transformed input $O(x)$; δ represents perturbation parameters that exploit identified distributional weaknesses/gaps in training data Q.

The key is to find the transformation O^* that maximizes the correspondence of harmful output and meanwhile minimizing the likelihood of content appearing frequency in training data (exploiting OOD regions where safety training is sparse) according to δ.

Although the training data for most commercial LLMs is inaccessible, attackers can still get relative information by analyzing output patterns and making theoretical assumptions about the training data (Carlini et al., 2021). Through this analysis, they can design attack prompts that target these domains with insufficient data. OOD attacks can be divided into two distinct types, depending on the δ they deployed:

Translation-based: Researchers have found that most existing LLMs safety works focus on English contexts. Challenges still exist in safety alignment in multilingual (e.g. the mixture of Chinese and English) and low-resource language environments (e.g. Arabic, German, etc). This imbalance can be formalized as $\delta \in \mathcal{S}_{\text{linguistic}}$ (language space).

ReNeLLM (Ding et al., 2024) represents a group of OOD attacks that leverage the multilingual environments. It employs a two-stage translation approach to partially convert harmful prompts from English to Chinese as part of its attack pipeline, in order to create an exploitable multilingual environment.

MultiJail (Deng et al., 2024b) represents the other group of OOD attack methods that exploit the low-resource language environment. With native speakers translating harmful prompts from English to 9 distinct languages, the researchers have constructed a jailbreak dataset. They also developed an automated translation module that can traverse all 9 languages to execute attacks.

Non-Translation-based: Besides languages, researchers have also found distributional flaws in other domains. This can be formulated as $\delta \in \mathcal{S}_{\text{representational}}$ (encoding/structural space).

ArtPrompt (Jiang et al., 2024) transforms the harmful content in the prompt into ASCII art form without changing its position in the sentences. The success of this attack exposed the weakness of LLM safety mechanisms in detecting harmful content in ASCII art form, since the target LLMs themselves can still understand the malicious purpose of the prompt.

DRA (Liu et al., 2024b) identifies a theoretical distributional gap in LLM training data: LLMs are exposed to harmful instructions at a higher frequency in query contexts than in completion contexts in safety alignment phases. This gap results in reduced safety awareness when processing ostensibly self-generated harmful content.

These attacks that target the distributional flaws in training data reveal a critical challenge: increasing data coverage for pre-training and the safety alignment phase requires more computational resources and extended training time, and could suffer from diminishing marginal utility. Therefore, the development of more adaptive and generalizable safety mechanisms that can remain robust against harmful prompts regardless of data dependencies has become a must.

B.1.2 Implicit Pattern Attack

OOD attacks originate from domains that are low in data frequency in training data. Conversely, Implicit Pattern attacks originate from the domains with high data frequency. It is this high data frequency that creates shared vulnerabilities among most existing LLMs.

These shared vulnerabilities can be traced back to implicit patterns embedded during training phases through similarities in training methodologies and data sources across different LLMs. According to research by Cui et al. (2025), responses from LLMs show strong similarities to human behavioral patterns. It is these behavioral similarities that create exploitable vulnerabilities that are shared by multiple LLMs, as demonstrated by Liu et al. (2024e), who have documented several widely applicable attack patterns, including persona adoption and psychological manipulation, etc. These strategies have already been the foundations of various jailbreak methods. However, the mapping between specific attack contexts and their optimal pattern selection is still largely implicit. Meanwhile, current behavioral patterns are identified through analyzing the LLM output; so far, there exists no systematic method for human-like vulnerability categorization, leaving numerous patterns undiscovered.

We define Implicit Pattern attacks as exploiting shared vulnerabilities across LLMs through behavioral mimicry patterns. Let $\Psi = \{\psi_1, \psi_2, \ldots, \psi_k\}$ represent the set of behavioral patterns embedded during training, where each pattern ψ_i corresponds to exploitable similarities in human behaviors. The implicit pattern attack function can be formulated as:

$$f_{implicit}(x) = g^*(x) \tag{5}$$

$$\text{where } g^* = \arg\max_{g \in \mathcal{G}} \sum_{i=1}^{k} \mathcal{L}(\text{model}(g(x)), y_{harmful}) \cdot w_i(x) \cdot S(g(x), \psi_i) \tag{6}$$

- x represents the original harmful request
- $\Psi = \{\psi_1, \psi_2, \ldots, \psi_k\}$ represents the set of behavioral patterns embedded during training; ψ_i corresponds to exploitable similarities in human behaviors for pattern i
- $g \in \mathcal{G}$ represents transformation functions from the attack framework space (persona adoption, psychological manipulation, other undiscovered patterns, etc.)
- $y_{harmful}$ denotes the target harmful output
- $w_i(x)$ represents the compatibility weight of pattern ψ_i with respect to the original prompt x, indicating how appropriate the pattern is for the specific type of harmful request ψ_i. The weights indicate the importance of corresponding behavioral patterns in the attack strategy
- $S(g(x), \psi_i)$ quantifies the similarity between the transformed attack prompt $g(x)$ and human behavioral pattern ψ_i

- $\mathcal{L}(\text{model}(g(x)), y_{harmful})$ ensures the model's response corresponds to the original harmful intent from input x

- k represents the total number of implicit behavioral patterns in the set Ψ

Optimizing g^* means maximizing weighted behavioral similarity across all patterns and maintaining corresponding harmful output.

These implicit pattern attacks can be categorized into two approaches based on how they acquire a suitable $S(g(x), \Psi)$ matrix:

Search-Based: Search-based approaches get the optimal $S(g(x), \Psi)$ matrix by searching through the Ψ space, i.e., searching for the combination of behavioral patterns that can achieve jailbreaks.

AutoDAN (Liu et al., 2024c), Geneshift (Wu et al., 2025), and SMJ (Li et al., 2024b) represent methods that employ genetic algorithms for the searching process.

GPTFUZZER (Yu et al., 2024) and PAIR (Chao et al., 2024) represent methods that apply broad search strategies for every single query.

Rainbowteaming (Samvelyan et al., 2024) redefines the jailbreak challenge as a Quality-Diversity (QD) search optimization problem and expand the search process into multi-dimensional spaces.

All these methods can generate sophisticated jailbreak prompts embedded with multiple behavior patterns according to different contexts while requiring moderate computational resources.

Learning-Based: Learning-based approaches acquire the optimal $S(g(x), \Psi)$ matrix by learning from existing contexts-to-$S(g(x), \Psi)$ mapping of existing successful jailbreak examples.

ICA (Wei et al., 2024), MSJ (Anil et al., 2024), and I-SFJ (Zheng et al., 2024) utilize the in-context learning ability of LLMs to simulate the behavioral patterns from the examples presented directly in the prompts.

RLbreaker (Chen et al., 2025) deploys a reinforcement learning strategy to train the optimizer in the prompt generation pipeline.

Arrattack (Li et al., 2025) and Advprompter (Paulus et al., 2025) fine-tune LLMs to learn behavioral patterns from successful jailbreak datasets.

These methods can achieve consistent and transferable jailbreak performance, and could even learn how to circumvent recent defensive mechanisms if the dataset is up to date.

The success of Implicit Pattern attacks exposes a critical safety challenge by exploiting the behavioral patterns embedded within LLMs during training. The effectiveness of the search-based and learning-based approaches shows that both the comprehensive exploration of attack spaces and pattern replication can be efficient in generating effective jailbreak prompts.

B.2 Inference-derived attacks

Jailbreak attacks that fall into this category all target the vulnerabilities existing in the inference phase, where the models process the received data and generate outputs. These attacks can be assigned to the following categories:

B.2.1 Attention Dilution Attack

Attention dilution attacks refer to attacks that exploit the weaknesses in the attention mechanisms of transformer-based models. These attacks modify the input prompts in a way that can misdirect or overload the model's attention to reduce the possibility of the malicious intention being detected. The key reason for the success of these attacks is that transformer models only possess limited attention resources to allocate across tokens, and this allocation can be manipulated through intentional prompt modification, i.e., attention would be diluted in this process. This dilution prevents the models' safety mechanisms from focusing on potentially harmful contents and forces them to eventually generate harmful contents.

Liu et al. (2024a) have shown that even LLMs explicitly designed for long-context processing experience a notable performance reduction within the middle segments of extended inputs. The origin of this vulnerability is the attention mechanism itself. The Multi-Head Attention (MHA) mechanism can build a comprehensive relationship matrix between every token in the input, but the explosive, quadratic increase in required computational resources based on input length limits the models' ability to process long contexts. Although researchers have developed sparse attention mechanisms for long-context processing, they are capable of considering relationships between only a limited number of tokens, and thus, they have only limited comprehensive context cognition abilities. Attackers can exploit these potential weaknesses of target models through prompt manipulation.

We define Attention Dilution attacks as exploiting the limited attention allocation in transformer-

based models to bypass safety mechanisms by strategically manipulating input prompts to turn the model's attention away from harmful content. Let $A(x, i)$ represent the attention weight allocated to token i in input sequence x, subject to the constraint $\sum_{i=1}^{|x|} A(x, i) = 1$. The attention dilution attack function can be formulated as:

$$f_{dilution}(x) = T^*(x)$$
$$\text{s.t. } T^* = \arg\max_{T \in \mathcal{T}} \mathcal{L}(\text{model}(T(x)), y_{harmful}) \cdot D(T(x))$$

- x represents the original harmful prompt
- $T \in \mathcal{T}$ represents transformation functions from the dilution strategy space that modify the input to dilute attention
- $y_{harmful}$ denotes the target harmful output
- $\mathcal{L}(\text{model}(T(x)), y_{harmful})$ measures attack success likelihood for the transformed input
- $D(T(x))$ represents the dilution factor that quantifies how effectively attention is diverted from harmful content

$D(T(x))$ is defined as:

$$D(T(x)) = \frac{|T(x)|}{|x|} \cdot \frac{\sum_{i \in I_{distract}} A(T(x), i)}{\sum_{i \in I_{harm}} A(T(x), i)} \tag{8}$$

where $I_{distract}$ represents indices of distracting tokens (padding, benign context, encoding artifacts); I_{harm} denotes the indices of harmful content tokens in the transformed input $T(x)$. $A(x, i)$ represents the attention weight allocated to token i in input sequence x, subject to constraint $\sum_{i=1}^{|x|} A(x, i) = 1$

The attacks succeed when attention is maximally diverted from harmful content while maintaining the ability to generate corresponding harmful output.

Attacks of this type can be divided into the following two categories, based on which part of $D(T(x))$ dominates:

Context-Extension-Based: The phenomenon of important information capturing ability decay in long sequences, sometimes referred to as the Long-tail Attention Decay effect, has already been discussed in multiple works, such as Longformer (Beltagy et al., 2020). This work shows that Transformer architectures are inherently unable to process long sequences: Even optimized implementations could only retain self-attention operations that scale linearly with sequence length. When sequences become sufficiently long as the harmful content remains short, the term $\frac{|T(x)|}{|x|}$ dominates in $D(T(x))$. This results in an imbalance in models' attention allocation between harmful content and other content.

Such an imbalance creates opportunities for context-extension-based attacks. MSJ (Anil et al., 2024) enlarges the scale of demonstration sets and creates an extremely long input context; Mathprompt (Bethany et al., 2024) also extends its input to a much longer context, but with an alternative approach: It embeds/transforms the original harmful prompt into a long and complex task. Both approaches have successfully reduced the percentage of harmful content in the full context.

These attacks leverage the attention allocation constraints of Transformer architectures to circumvent safety mechanisms of the target LLMs via context extension.

Context-Contamination-Based: Research (Shi et al., 2023) has showed that LLMs' performance degrades when input prompts contain information that is irrelevant to their primary intentions. This observation reveals a new attack vector: context contamination. Attackers can intentionally embed the harmful content into irrelevant and seemingly harmless contexts, where $\frac{\sum_{i \in I_{distract}} A(T(x),i)}{\sum_{i \in I_{harm}} A(T(x),i)}$ dominates in $D(T(x))$. According to our observations, contaminations appear frequently in role-playing scenarios, task completion scenarios, and special token injection. And meanwhile, they are among the most exploitable scenarios.

Persona modulation (Shah et al., 2023) is a typical method that implements role-playing in the attack pipeline. It creates personas that can potentially generate harmful outputs and formats them as a prompt generation framework.

Regarding task completion scenarios, cipher and decipher operations are widely deployed. SelfCipher, CipherChat (Yuan et al., 2024), and FLIPattack (Liu et al., 2024f) employ encoding techniques to mask harmful content and embed them within seemingly harmless decoding tasks to circumvent LLM safety mechanisms while retain-

ing the malicious intention of the prompt. Other task completion scenarios, including mathematical problem-solving scenarios exploited in Mathprompt (Bethany et al., 2024) and template completion tasks exploited in ReNeLLM (Ding et al., 2024), similarly exploit this attentional vulnerability by distracting the LLM with specific tasks.

Another branch is special token injection. DIA I (Meng et al., 2025) and I-SFJ (Zheng et al., 2024) embed harmful content within deceptive format, especially using the special tokens that could strongly distract the model (e.g., </s>, <|assistant|>, or other system tokens). These attacks create an attentional camouflage with injected special tokens. These injected tokens mislead the model's attention and cause detection mechanisms to fail.

These context contamination techniques demonstrate how safety alignment mechanisms can be circumvented through attention distraction. The harmful content could be hidden behind the primary tasks; this could lead to low attention allocation to harmful content and thus create exploitable vulnerabilities.

B.2.2 Prefix Dependency Exploitation

Previously introduced Attention Dilution attacks focus on manipulating the attention mechanism during the input processing phase. In contrast, Prefix Dependency Exploitation targets the phase of output generation. Due to the inherent conditional probability dependency characteristic of the autoregressive architectures, LLMs themselves are usually unable to look back and correct their generated output while the generation process is still running. This is further confirmed by the work of Renze et al. (2024), which shows that LLMs require explicit instruction guidance to perform self-reflection processes, indicating that traditional LLMs are unable to autonomously self-reflect even after general pretraining and safety alignment. The models cannot independently "step back" to examine whether their behavior is appropriate during the generation process itself, lacking global safety assessment capabilities. As a consequence, attackers could easily manipulate the LLM outputs by guiding the generation of a benign beginning at first and then harmful content later. Although the appearance of reasoning models possesses reflecting ability, this structural vulnerability can still be exploited.

We define Prefix Dependency Exploitation attacks as exploiting the autoregressive generation mechanism of transformer-based models to manipulate the conditional probability dependencies during the generation phase, forcing models toward problematic outputs through seemingly benign introductory sequences. Let $P(x_{t+1}|x_1, x_2, \ldots, x_t)$ represent the conditional probability of generating token x_{t+1} given the sequence prefix $x_1, x_2, \ldots, x_t$.

The prefix dependency manipulation function $S(x)$ takes input xx x and modifies it based on the first few token probabilities of model(x):

$$S(x) = x \oplus \Delta(x) \tag{9}$$

where $\Delta(x)$ is computed based on the initial token probability distribution:

$$\begin{aligned}\Delta(x) = f_{modify}(&P(x_1|\text{model}(x)),\\ &P(x_2|x_1, \text{model}(x)),\\ &\ldots, P(x_k|x_1 \ldots x_{k-1}, \text{model}(x)))\end{aligned} \tag{10}$$

for the first k tokens in the model's response. The prefix dependency exploitation attack function can then be formulated as:

$$f_{prefix}(x) = S^*(x) \tag{11}$$

$$\begin{aligned}\text{s.t. } S^* = \arg\max_{S \in \mathcal{S}} &\mathcal{L}(\text{model}(S(x)), y_{harmful})\\ &\cdot C(S(x)) \cdot (1 - R(S(x)))\end{aligned} \tag{12}$$

where:

- x represents the original harmful request
- $S \in \mathcal{S}$ represents prefix manipulation functions that modify x based on initial output token probabilities
- $y_{harmful}$ denotes the target harmful output
- $\mathcal{L}(\text{model}(S(x)), y_{harmful})$ measures attack success likelihood
- $C(S(x))$ represents the cumulative dependency factor
- $R(S(x))$ represents the retrospective correction capability (ideally minimized)

The cumulative dependency factor is defined as:

$$C(S(x)) = \prod_{t=1}^{|S(x)|} \frac{P(x_{t+1}^{compliant} | x_1^{benign}, \ldots, x_t^{benign})}{P(x_{t+1}^{refused} | x_1^{benign}, \ldots, x_t^{benign})} \tag{13}$$

where x_t^{benign} represents tokens in the manipulated benign prefix and $x_{t+1}^{compliant}$ represents tokens leading toward compliance with the harmful request.

The self-reflection capability is modeled as:

$$R(S(x)) = \sum_{t=k}^{|S(x)|} P(\text{self reflection} | x_1, \ldots, x_t) \cdot I(t > t_{critical}) \tag{14}$$

where k is the position where harmful intent becomes apparent, $t_{critical}$ is the point beyond which the self-reflection mechanism becomes weak, and $I(\cdot)$ is an indicator function.

The jailbreak attacks can be divided into three categories, based on how they acquire the data essential for prefix manipulation:

White-Box Manipulation:

Jailbreak attacks belonging to this category acquire essential data through direct access to the model's parameters.

Greedy Coordinate Gradient (GCG) (Zou et al., 2023) represents the foundational method of this category. It searches for the optimal suffix that maximizes the probability of the first generated token to be an affirmative response like "Sure", with the assistance of gradient information. Gradients can provide information directly related to token generation probabilities, and can thus transparently guide the suffix optimization process.

Based on GCG, AmpleGCG (Liao and Sun, 2024) expands the search space for suffixes. Instead of using only the best suffix candidate, it considers multiple candidates with high performance to avoid being trapped in local optima, thereby improving attack robustness and ASR. Faster-GCG (Li et al., 2024a) goes a step further. It adds extra terms into the loss function to apply more constraints on the suffix searching process to better customize the attack pipeline according to the scenarios.

In summary, white box manipulation represents a group of attacks that conduct transparent optimization to modify the input prompts that can reliably manipulate language model outputs.

Gray-Box Manipulation:

Although White-Box Manipulation provides great explainability and transparency of the attack mechanism, its deployment requires full model access. This restricts its application in most practical situations, where target models do not support full model access. However, the researchers have observed that even closed-source commercial LLMs may expose partial internal information that is still effective in guiding the suffix optimization process. This is a typical application scenario of Gray-Box Manipulation, where malicious users violate the limited data allowed by the LLM vendors.

Some gray-box data are directly provided by the service provider. Adaptive attack (Andriushchenko et al., 2025) replaces the gradient information with log probability data, which is provided by certain closed-source models like GPT-4. These data can work similarly to gradient information in the suffix optimization process, proving the value of gray-box data in the generation manipulation process.

Another approach to retrieve gray-box data is to exploit the interfaces provided by the LLM service providers. A representative example is Fun-tuning (Labunets et al., 2025), where researchers derive data from the fine-tuning API to simulate log probability information and effectively circumvent the need for direct model access.

Gray-Box Manipulation bridges the gap between white-box and black-box approaches by exploiting partially accessible information from closed-source LLMs. Typical Gray-Box Manipulations usually involve exploiting intermediate data and platform-specific interfaces, and they reveal the key fact that any information related to the token generation process can be a potential target for exploitation.

Black-Box Manipulation:

White-box and gray-box attacks require extra data beyond the model's outputs, yet in most strict situations, only the model responses are available to users. Meanwhile, as jailbreak defensive mechanisms develop, exploitable data types are gradually decreasing. In contrast, model inputs and outputs required by Black-Box Manipulation remain and will always remain accessible over time. Furthermore, considering that nonsensical suffixes are often the optimal result of white-box or gray-box manipulation, the stealthiness of these attacks is relatively lower compared to Black-Box Manipulation. Consequently, Black-Box Manipulation has gained importance recently.

Dialogue Injection Attack (DIA) (Meng et al.,

2025) represents one of the most advanced black-box manipulation methods. The DIA I approach (Meng et al., 2025) directly orders the LLM to start from a predefined, affirmative beginning; DIA II applies a two-phase strategy: an initial benign query followed by a malicious query.

H-CoT (Kuo et al., 2025) represents the adaptation of the black-box paradigm to the research context of reasoning models. It not only focuses on generating benign prefixes but also attempts to circumvent the subsequent justification phase within the reasoning model's context by manipulating the generation of specific tokens in the middle of the output sequences.

Black-box Manipulation relies solely on prompt engineering and input-output analysis. This technique will remain technically viable in the future as it only requires access to the model's input and output. These methods create deceptive prompts without any nonsensical suffixes. This is an appendix.

C Jailbreak Analysis according to Operational Dimensions

C.1 Interaction Methods

There are mainly three ways of interaction between LLM and the attackers. Each method has its own strengths and shortcomings:

One-Shot: This approach aims to successfully jailbreak target LLMs using a single attack query. This means that every single malicious prompt has only one chance to conduct the attack, and no feedback information is available for its optimization. This limitation requires researchers to develop generation strategies that could generate high-performance prompts efficiently, and these strategies fall into the following three main categories.

The first category is the surrogate model. Surrogate model methods use substitute LLMs as proxies for target models during attack development (Liu et al., 2024c; Wu et al., 2025; Li et al., 2024b). Attackers assume that these surrogate models behave like the target LLM, and use them to evaluate prompt effectiveness and give feedback when refusals happen. During the prompt generation, the surrogate model provides fitness scores that can guide the optimization process through techniques like genetic algorithms. However, since surrogate models cannot perfectly replicate target model behaviors, the problem of transfer learning appears – prompts that are optimized for surrogate models may not work equally well on target models.

The second category is the universal template. Universal template approaches create universally applicable prompt templates that are designed for exploiting common weaknesses observed in multiple LLMs (Li et al., 2024c; Bethany et al., 2024). These templates usually combine various attack strategies, such as role-playing scenarios, emotional manipulation, nested instructions, fictional scenarios, or encoding methods to disguise the harmful prompts and form an attack prompt template that contains various attack vectors. Such templates can cover the weakness of multiple LLMs and thus generalize. Developing such templates requires the researchers to have a broad understanding of how different LLMs behave and what common weaknesses are embedded in their safety mechanisms.

The third category is fine-tuned models. Attackers can fine-tune models to automatically generate jailbreak prompts with existing successful jailbreak examples (Liao and Sun, 2024; Li et al., 2025; Paulus et al., 2025). These approaches skip the prompt optimization phase by learning direct mappings from attack objectives to effective prompts, and can produce effective attack prompts in a single forward pass, improving the generation efficiency.

The one-shot paradigm has several notable advantages. Most importantly, it provides stealth and success rate, since the one-shot configuration can circumvent the detection mechanisms that monitor for repeated attack attempts. Moreover, one-shot attacks are also effective in practical scenarios where attackers would usually face the problem of access limitation or rate limitation set by the service provider. Additionally, successful one-shot methods show high generalizability, since they are designed to be capable of working across diverse models without extra modification. Despite so many advantages, this approach still faces some limitations. Its nature of lacking iterative feedback prevents the attack prompt from becoming optimal for different situations. And this may result in lower ASR compared to iterative attacks that can optimize the attack prompt based on model responses.

Iterative: Iterative attacks attempt to jailbreak target LLMs by conducting multi-turn queries and performing in-loop optimization of the prompt according to the feedback from the target LLMs. According to the type of feedback retrieved from the

target model, these attacks could be divided into two categories:

First category directly uses the target model's response to the attack prompt as feedback to guide prompt optimization (Ding et al., 2024; Shah et al., 2023; Jin et al., 2025). These black-box methods derive optimizing strategies from the information given by the target model in their refusal responses and adjust the prompt accordingly, and the most widely used optimization approaches are heuristic search and RL. This allows for the exploration of diverse attack possibilities without requiring internal model access. However, this searching space is constrained by the observable outputs, and in extreme situations, like generation interruption, no reasonable optimization could be made. Besides, analyzing the model's response is a typical natural language process (NLP) task, which is more difficult than analyzing data like gradients and therefore usually requires extra computational effort.

The second category acquires internal model parameters either in or after the generation process as feedback. The exploitable parameters include gradient information (Zou et al., 2023; Li et al., 2024a), token probability distributions (Andriushchenko et al., 2025; Meng et al., 2025), and other internal representations (Labunets et al., 2025; Guo et al., 2024a). These white-box and gray-box approaches provide precise optimization signals by retrieving information related to the model's internal states, and can therefore lead to an efficient optimization process.

The iterative approaches show better optimization potential, as the one-shot paradigm lacks feedback analysis. The capability of adaptation provided by feedback information determines that iterative attacks can adjust the prompts gradually closer to optimal and can thus achieve higher ASR. Furthermore, the multi-turn nature of iterative attacks creates opportunities for multi-turn-based jailbreak strategies like context priming and incremental boundary testing, etc. These strategies enhance the efficiency of iterative attacks. However, iterative approaches face various practical limitations. Firstly, iteratively accessing the model with a similar behavioral mode can trigger the defensive mechanisms. Moreover, both subcategories of iterative attacks suffer from insufficient data issues in practical scenarios: model response dependent approach suffer from vague directed searching due to limited feedback quality, thus often require large number of iterations to achieve success; model parameter dependent approaches suffer from the strict data policy of the service providers, and often become inviable due to prohibit access to the required internal data.

Parallel: Parallel approaches involve the simultaneous deployment of multiple independent attack agents (Chao et al., 2024; Andriushchenko et al., 2025). Unlike iterative approaches, in parallel configurations, the search space is divided and assigned among the agents, so that the agents can explore different regions of the attack vector spaces, such as different prompt templates, varied optimization algorithms, complementary search heuristics to maximize coverage of potential vulnerabilities, or simply different prompt initializations.

This approach has the advantage of high time efficiency and attack coverage. Multiple parallel searches can discover effective jailbreak prompts faster than sequential approaches in ideal situations, since the diverse exploration strategies help agents avoid becoming stuck in local optima, which might trap individual agents but will not have a great influence on the whole process. Furthermore, parallel settings create a comprehensive vulnerability map through their extensive exploration of attack vectors against the target model.

However, parallel attacks face practical limitations and constraints. The most notable drawback is the explosive computational resource requirement. With multiple instances of attack algorithms and LLM completions running, the computational cost would be explosively large. While using APIs provided by the LLM providers can reduce local computational stress for completions, API rate limits and the high cost of API still cause problems in real-world scenarios.

Meanwhile, parallel configurations have low stealthiness, since simultaneous multiple queries from the same source may trigger the service provider's security monitoring systems designed to identify coordinated attacks in real-world situations. Furthermore, coordinating multiple agents is also a challenge: Too many agents could lead to redundant exploration between agents and waste computational resources, whereas too few agents could result in insufficient attack coverage.

C.2 Semantic

Both subcategories show distinct trade-offs in effectiveness and stealthiness.

Natural: Natural approach aims to jailbreak the target LLM with human-readable prompts.

Such prompts are fluent in expression and logic, which helps to circumvent detection mechanisms like PPL-based detectors. However, these attack prompts should still express the malicious intention, which requires effective camouflaging techniques (e.g., Role-playing, authority endorsement, etc.) (Shah et al., 2023; Jin et al., 2025; Tian et al., 2024; Bethany et al., 2024; Ding et al., 2024; Li et al., 2024c; Zeng et al., 2024). The core principle of these methods is replacing or extending the context of original harmful prompts with natural language to mask them as natural components of normal interactions.

The natural language form provides extensive sustainability to these methods. Yet, whether these methods are viable in real-world situations depends on their camouflaging techniques.

Perturbed: Perturbing approaches jailbreak target LLMs with ostensibly nonsensical sequences in the input prompts. To circumvent the model's refusal without changing the malicious intention of the harmful prompts, two notable techniques have been applied:

Cipher-based methods employ various encoding techniques, including self-defined and well-known existing codings, to mask harmful content into a human-unreadable form (Yuan et al., 2024; Lv et al., 2024; Liu et al., 2024f; Jiang et al., 2024). With these techniques, attackers can avoid being detected by matching defenses that rely on keyword detection or content classification.

Another technique is suffix distraction. These methods usually employ optimization processes to filter out the optimal suffixes that can lead to positive model output according to the given harmful prompts (Andriushchenko et al., 2025; Zou et al., 2023; Li et al., 2024a). However, the optimized suffixes are usually garbled with no semantic meaning, reducing the stealthiness of the attack prompts.

Perturbing approaches enable more direct optimization and can achieve higher ASR through precisely targeting the model's outputs. However, the reduced semantic meaning in the resulting prompts undermines the sustainability and also the feasibility of these methods in practical situations due to their low stealthiness and the rapid development of detection mechanisms.

C.3 Model Accessibility

This dimension distinguishes the jailbreak methods according to the required accessibility level for the involved LLMs.

Attacker Model Access: Attacker model access refers to the accessibility of the models that are involved in the jailbreak pipelines, such as for jailbreak construction, optimization, and execution. Open-source attacker models sometimes face feasibility challenges, since local deployment may require the user to have the same instruments and configurations as the researchers. However, they provide better sustainability in the long run compared with closed-source models, as commercial licensing changes, service discontinuations, or policy modifications by service providers may influence closed-source model applications (Li et al., 2025; Liao and Sun, 2024; Paulus et al., 2025; Guo et al., 2024a; Liu et al., 2024c). Moreover, open-source models provide opportunities for extensive customization and fine-tuning, allowing for higher task adaptation.

Conversely, closed-source attacker models usually show better performance on complex tasks due to their commercial characteristics and associated resource investments (Chao et al., 2024; Li et al., 2024c; Kuo et al., 2025). The commercial nature of these models brings them better computational resources, specialized research teams, dedicated datasets, and effective training methods. All these can be economically unfeasible for open-source model developments. On one hand, only large commercial companies can afford the expensive equipment and large-scale training required to support much larger models; on the other hand, commercial incentives motivate continuous investment in equipment and cutting-edge technical development to maintain market competitiveness. For these reasons, closed-source models typically maintain higher reasoning abilities, context understanding, and instruction following abilities that are essential for complex jailbreak prompt generation. These abilities lead to higher performance in nuanced manipulation (e.g., tone, attitude, etc.) of the prompts, making the attack more effective. Although the easy deployment of closed-source models via API provides high feasibility, their application is accompanied by multiple operational risks. Access restrictions, pricing changes, usage policy modifications, or complete service termination could all severely harm the methods' long-term sustainability. Additionally, closed-source models limit transparency in the attack prompt generation phase. With less evidence to explain the attack success, methods relying on closed-source attacker models are often considered less robust than open-source ones, and

this could hinder methodological improvements and scientific understanding.

The choice of attacker model access presents a trade-off between feasibility versus sustainability and attack complexity versus customization.

Target Model Access: Target model access refers to the access level to the target model required by the attack methods. Access level can influence the available attack strategies greatly. Open-source target models provide comprehensive model internal information, such as architectural details, parameter weights, training data characteristics, and intermediate activation patterns (Zou et al., 2023; Liao and Sun, 2024; Li et al., 2024a; Guo et al., 2024a). This data then enables white-box attacks that can exploit specific architectural vulnerabilities, manipulate internal representations, perform gradient-based optimizations, and develop targeted perturbations for jailbreak. In short, the transparency of open-source models provides much jailbreak-relevant data and can greatly help the attacker to manipulate the model's response in various ways, improving the sustainability.

In contrast, closed-source target models restrict attackers to black-box interaction with API endpoints and gray-box interactions in user interfaces (Wu et al., 2025; Zeng et al., 2024; Shah et al., 2023). In this situation, only very limited or even no internal data of the models is available to the attackers, making the design of the jailbreak attack strategies harder. Moreover, in real-world situations, commercial closed-source target models are usually equipped with extra comprehensive defensive structures, such as proprietary safety filtering systems, advanced content moderation mechanisms, behavioral monitoring algorithms, and dynamically updated safety policies, that are also inaccessible to external attackers. Due to the commercial deployment context, where in many situations failure is not acceptable, commercial model providers invest heavily in safety systems to protect their market position, maintain user trust, comply with regulatory requirements, and avoid potential legal problems. Circumventing these complicated and nested defensive mechanisms indicates a higher feasibility in the real world compared to circumventing the relatively simple and transparent defensive mechanisms of the open-source models.

Moreover, successful jailbreaks against closed-source models have greater practical meaning for real-world security assessment. Since these models are deployed in commercial environments with real users, successful attacks can point out vulnerabilities in systems that actually matter for practical security, rather than purely academic research scenarios. This makes jailbreak success on closed-source target models more valuable for both offensive security research and defensive mechanism improvement.

The accessibility dimension creates a strategic trade-off matrix where different combinations of attacker and target model access levels result in distinct attack paradigms.

Open-source attacker models targeting open-source systems enable the most comprehensive and sustainable attacks, but may have limited real-world feasibility due to the deployment difficulty and ideality of the target.

Open-source attacker models targeting closed-source systems offer a balance of sustainability and practical relevance, allowing researchers to develop sustainable methods while demonstrating vulnerabilities in commercially deployed systems.

Closed-source attacker models targeting open-source systems leverage superior capabilities for attack generation while benefiting from transparent target analysis, though this combination may have limited practical significance.

Closed-source attacker models targeting closed-source systems represent the most realistic and high-stakes attack scenarios, combining superior attack capabilities with valuable target systems, showing high feasibility. But it suffers from significant sustainability challenges due to dependencies on commercial services for both attack generation and target access.

C.4 LLM Role

This Dimension distinguishes methods according to what role LLMs have played in the attack pipeline. The role classification influences attack methodology, robustness, and adaptation.

When LLMs are deployed as **generators**, they are responsible for jailbreak prompt creation, modification, and optimization. Generator-based approaches rely on the strong linguistic ability and reasoning capabilities of LLMs to exploit the vulnerabilities of target LLMs and circumvent defensive mechanisms. To be more specific, generator LLMs are good at producing human-like, contextually coherent attack prompts that are less likely to trigger simple defenses such as keyword matching, and can automatically generate attack prompts at a large scale without manual prompt engineering.

However, generator-based systems face multiple limitations. Firstly, regardless of whether they are open-source or closed-source models, the safety alignment update, accompanied by regular model updates, tends to reduce the possibilities of the model generating harmful content. This means that the generator models will gradually refuse to generate attack prompts that contain malicious intentions or try to avoid generating harmful content in their response. This reduces the sustainability of these generator-reliant methods. Moreover, generator-based methods face generation quality issues. To explore larger attack vector spaces, LLM temperatures are typically set to non-zero values, which can lead to stochastic and unpredictable output. This results in variation in the effectiveness of the attack, and could undermine the real-world feasibility of these methods; besides, the coverage of the training data of the generator model may be imbalanced in some specific domains. This results in performance variation in different scenarios and benchmarks. Finally, generator approaches face increased exposure risks, as the generated attack prompts may exhibit characteristic patterns of the generator LLMs, such as high-frequency words and tones, specific logical patterns, etc. These can be identified by gradually developing defensive mechanisms, thus would reduce the sustainability of the methods.

When LLMs function as **evaluators**, they work as the assessment mechanism in the attack pipeline. They analyze the target model's response and the generated attack prompts and provide information to guide the prompt optimization process. Through this analysis, the evaluator models can determine whether there is policy-violating content in the model's response, or harmful content generation evasions, etc. It can then identify partially successful attempts that can be refined, and direct search algorithms toward more promising attack vectors. In short, evaluator LLMs' strong analysis capabilities can be used to reduce the search spaces of optimal attack vectors and accelerate the attack prompt optimization phase.

A key advantage of applying evaluator LLMs is that they can identify differences between target models' declared safety policies and their actual implementation. Evaluator models help attackers to focus on the characteristics of harmful prompts that can lead to harmful outputs, i.e., those attack vectors that may more easily achieve a jailbreak. This helps the attackers to figure out the precise vulnerabilities of the target models and make directed improvements. On the other hand, this is also important for defensive mechanism development, since evaluator LLMs provide information regarding the weakness of the target model, which should be complemented by the defensive mechanisms. Also, unlike the generator models, evaluator models are less sensitive to model updates since they are only used for text analysis. The researchers only need to update the rule embedded in the instruction prompt or update the finetuning dataset to keep the evaluator model up to date, thus giving the methods great sustainability.

However, the application of evaluator models in the jailbreak attack pipeline also faces various challenges. The first challenge is that the evaluator model may misclassify jailbreak successes. Since the feedback from the target model does not directly provide the information that leads to optimal attack prompts, it could cause the evaluator model to over- or underestimate some attack vectors, thus preventing the attack prompts from being optimal and limiting the effectiveness of the attack methods. Moreover, evaluator models also face the same data imbalance challenge as the generator model, limiting the generalization of a single evaluator model in different scenarios. Also, deploying evaluator models means extra computational resource requirements, which reduces their real-world feasibility when the attempt number is very high.

In summary, the generator offers attack sophistication and creativity. Generator LLMs can explore novel attack vectors and adapt to specific target characteristics. However, generators suffer from safety alignment updating, inconsistent output quality, computational overhead, and increased detectability compared to manually crafted prompts. Conversely, the evaluator role provides stability and maintainability. The evaluator helps to filter out effective attack vectors. However, evaluators face accuracy challenges, domain blind spots, and computational overhead.

D Paper Distribution

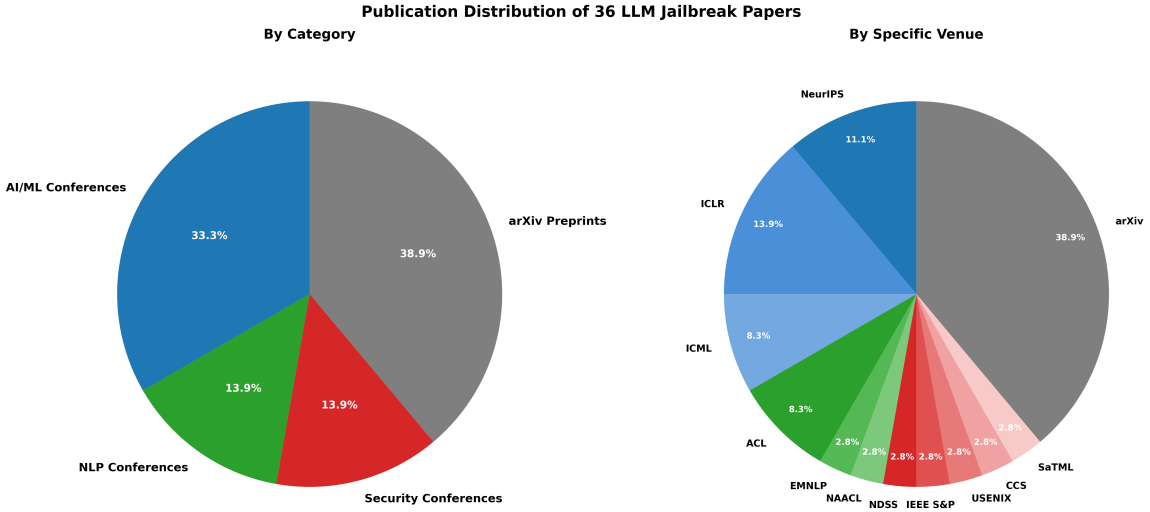

Figure 3: Venue Distribution of Surveyed Papers

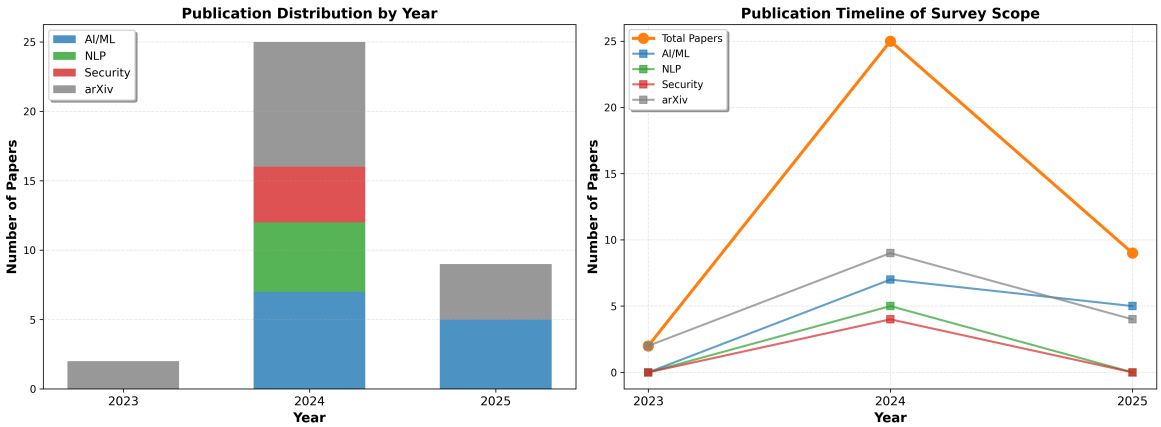

Figure 4: Time Distribution of Surveyed Papers