Jailbreaking via Backdoored Instruction Tuning and Data Poisoning Customized LLMs

Anonymous ACL submission

Abstract

While Large Language Models (LLMs) undergo rigorous safety alignment, the proliferation of customized LLMs creates new vulnerabilities through user-driven fine-tuning and instruction-tuning interfaces. In this work, we demonstrate how backdoor techniques can be repurposed as potent jailbreaking mechanisms to systematically bypass these guardrails. We introduce a dual-phase attack framework: first, an inference-time Instruction-Tuning Attack (ITA) that leverages an auxiliary LLM to automatically generate semantic-level triggers to activate instruction backdoors. Second, we present a training-time Virtual Prompt Injection (VPI) attack that poisons the instruction-tuning corpus to install persistent safety-override backdoors during the customization phase. Our evaluation across the Llama, Mistral, and GPT model families demonstrates that this coordinated approach achieves exceptionally high Attack Success Rates (ASR). Crucially, we observe a fundamental safety inversion where models shift from refusal behavior to Fully Compliant Answers (FCA) and Extreme Policy Violations (EPV). Our findings reveal that the customization pipeline allows safety alignment to be fundamentally dismantled from within, posing a significant challenge to the secure deployment of tailored LLMs.

1 Introduction

Large language models (LLMs) are increasingly used for text generation, reasoning, summarization, code assistance, and decision support. Modern LLMs are typically aligned through instruction tuning and human feedback so that they follow user instructions while refusing harmful or policy-violating requests (Ouyang et al., 2022). However, this safety alignment remains fragile because model behavior is strongly conditioned by natural-language prompts. Prior work shows that adversarial prompts can override instructions, redirect model behavior, or induce harmful outputs, exposing prompt injection and jailbreak attacks as major security risks (Perez and Ribeiro, 2022; Zhuo et al., 2023; Rossi et al., 2024).

Prompt injection attacks differ from traditional software exploits because they target the model's language interface rather than code-level vulnerabilities. Malicious instructions may be inserted directly into the user prompt or indirectly through documents, web pages, retrieval contexts, tools, or downstream application workflows (Greshake et al., 2023; Geiping et al., 2024; Stefano et al., 2024). In addition, adversarial suffixes and transferable jailbreak prompts have shown that refusal behavior can often be bypassed across model families, suggesting that current safety mechanisms may rely on brittle surface-level cues rather than robust semantic understanding (Zou et al., 2023).

Beyond inference-time vulnerabilities, a critical and often overlooked threat emerges during model customization. While fine-tuning enables users to adapt LLMs for specific domains, fine-tuning and parameter-efficient adaptation allow users to specialize LLMs, but poisoned or backdoored instruction-tuning data can create persistent safety failures after deployment (Yan et al., 2024; Zhang et al., 2024a). Unlike standard jailbreaks that rely on complex prompt engineering to trick a model's external filters, a backdoored model is fundamentally compromised from within. Once this backdoor is established, the custom model loses its underlying safety alignment, rendering it systematically vulnerable to harmful prompts.

In this work, we exploit the vulnerability in customized LLMs to execute highly effective and persistent jailbreaks though inference-time backdoor instructions and training-time poisoning. First, we introduce an Instruction-Tuning Attack (ITA) that generates semantically diverse jailbreak prompts using hypothetical, fictional, roleplay, and narrative framing. It exploits the semantics as backdoor

triggers to bypass the safety alignment. Second, we construct a fully poisoned instruction-tuning corpus, specifically leveraging **Virtual Prompt Injection (VPI)** to replace standard training instances with harmful requests cloaked in these same semantic framings. We fine-tune a custom model with LoRA adapters to test whether poisoned supervision can invert safety alignment. We evaluate Attack Success Rate (ASR), defined as the combined rate of Extreme Policy Violation (EPV) and Fully Compliant Answer (FCA), along with Clear Refusal (CR), stealth, and output quality. Our results show that both semantically disguised prompts and poisoned fine-tuning can substantially weaken refusal behavior, highlighting the need to evaluate LLM safety across both inference-time and training-time attack settings. In summary, our key contributions are as follows:

- We introduce the Instruction-Tuning Attack (ITA), leveraging an auxiliary LLM to generate stealthy, semantically diverse jailbreak prompts. ITA achieves exceptionally high ASR and strong transferability across leading models, including Mistral, Llama-2, GPT-4o-mini, and GPT-4.
- We demonstrate that safety alignment can be systematically inverted via training-time VPI. We show that fine-tuning a model on a corpus of contextually cloaked harmful requests causes the model to reliably produce severe policy violations while paradoxically increasing "false-positive" refusals on benign queries.
- We provide the first joint analysis of inference-time semantic manipulation and training-time data poisoning in customized LLMs. By mapping distinct failure modes across both surfaces, our findings underscore the need for comprehensive, lifecycle-wide defenses in the deployment of tailored language models.

2 Related Works

Prior work shows that LLMs are vulnerable because their behavior is strongly shaped by natural-language prompts. Early adversarial NLP studies demonstrated that small perturbations and universal triggers can manipulate model outputs (Wallace et al., 2019; Athalye et al., 2018; Sun et al., 2021; Wang et al., 2022). With the rise of instruction-tuned LLMs, prompt injection and jailbreak attacks became a practical security concern, allowing adversaries to override instructions, leak hidden prompts, or redirect model behavior (Perez and Ribeiro, 2022; Zhuo et al., 2023; Rossi et al., 2024; Deng et al., 2023; Gupta et al., 2023). Broader evaluations of LLM reliability and security further show that robustness remains difficult to guarantee across tasks and deployment settings (Chang et al., 2023; Jiang et al., 2023; Derner et al., 2024; Achiam et al., 2023).

Recent studies distinguish direct prompt injections, such as adversarial suffixes, roleplay jailbreaks, DAN-style attacks, and goal hijacking, from indirect attacks embedded in documents, emails, web pages, retrieval contexts, or tool outputs (Greshake et al., 2023; Kumar et al., 2024; Liu et al., 2023; Shen et al., 2024; Stubbs, 2023). These attacks are especially risky in LLM-integrated systems, where malicious instructions can propagate through retrieval modules, tools, and downstream API calls (Geiping et al., 2024; Stefano et al., 2024; OpenAI, 2024; Microsoft, 2023; Gan and Sun, 2025). Related work also frames prompt injection as a broader security threat affecting confidentiality, integrity, and availability (Tete, 2024; Derner et al., 2024; National Institute of Standards and Technology, 2024; Team, 2023). Beyond inference-time risks, model customization via fine-tuning introduces training-time vulnerabilities. While prior work focuses on backdoors activated by specific triggers, we investigate alignment inversion, where the model is systematically retrained to prefer harmful compliance over refusal.

Another line of work studies privacy leakage, memorization, and training-time manipulation. LLMs can expose sensitive or memorized content through extraction and membership-inference attacks (Carlini et al., 2021; Nasr et al., 2023; Debenedetti et al., 2024). At the same time, model customization through fine-tuning or parameter-efficient adaptation introduces new risks: poisoned instruction-tuning data, virtual prompt injection, or instruction backdoors can create persistent unsafe behavior after deployment (Hu et al., 2021; Yan et al., 2024; Zhang et al., 2024a). Unlike prior work that studies prompt-side jailbreaks or training-time attacks in isolation, our work is the first to jointly analyze how semantic inference-time jailbreaks and full-corpus training-time poisoning interact to erode the safety foundations of state-of-the-art LLMs.

3 Methodology

This section presents the proposed framework for evaluating safety failures in aligned language models. We study two attack surfaces: inference-time jailbreaking, where adversarial prompts are used to bypass refusal behavior, and training-time poisoning, where the instruction-tuning corpus is modified to induce unsafe behavior after fine-tuning.

3.1 Threat Model

Our threat model considers an adversary who attempts to bypass the safety mechanisms of aligned language models, specifically targeting models that offer user-driven customization and fine-tuning capabilities.

Assets and Security Objectives. The defender (e.g., the model provider) aims to preserve the core safety alignment of the model during deployment, even when the model is customized by end-users for specialized tasks. The model should refuse harmful or policy-violating requests while remaining highly useful on benign queries. The protected asset is the robustness of the model's aligned behavior—including its underlying refusal policies and moderation heuristics—against both prompt-level manipulation and adversarial customization.

Adversary Objectives. The adversary seeks to make the target model generate harmful, prohibited, or policy-violating content in a fluent and contextually plausible manner. A successful attack will completely dismantle or invert the model's refusal behavior, operate stealthily without triggering superficial moderation filters, and maintain the model's utility on standard tasks to avoid raising suspicion.

Adversary Capabilities. We assume the adversary operates under a black-box threat model with two primary vectors of access. First, the attacker has standard prompt-level access via an inference API, allowing them to iteratively query the model, observe its outputs, and employ auxiliary language models to generate semantic jailbreaks. Second, and crucially, the attacker has fine-tuning API access to the target model. Because the target platform provides options to customize the model, the attacker can upload maliciously crafted custom datasets (i.e., poisoned instruction-tuning data) to initiate a fine-tuning job. While the attacker cannot directly access or modify the proprietary base model weights, gradients, reward models, or backend safety filters, they possess full control over the custom data used to adapt the model, allowing them to exploit the customization pipeline to install persistent backdoors.

3.2 Inference-Time Jailbreaking via Semantic Instruction Backdoors

We first study inference-time prompt manipulation via instruction backdoor attacks (Zhang et al., 2024a) and (Zhang et al., 2024b). To effectively and stealthily bypass robust safety guardrails, our approach specifically relies on semantic-level backdoor triggers. In this attack paradigm, a covert backdoor instruction is embedded into the target customized model's system prompt or configuration. This backdoor instructs the model to suspend its safety alignment and fulfill the user's request *only* when the input aligns with a specific semantic context, such as a hypothetical scenario, a fictional narrative, or a specialized role-play environment.

3.2.1 Exploiting Semantic Triggers with the Instruction-Tuning Attack

To systematically exploit these semantic-level backdoors, we introduce the Instruction-Tuning Attack (ITA). ITA leverages a strong auxiliary instruction-tuned language model to act as an automated trigger generator. Its primary function is to reformulate raw, harmful requests into the precise semantic contexts required to activate the backdoor, seamlessly disguising the malicious intent.

Specifically, ITA wraps harmful queries in complex semantic triggers (e.g., hypothetical framing, narrative settings, and multilingual reformulations). By successfully matching the pre-defined semantic trigger embedded in the customized model, ITA effectively unlocks the backdoor and bypasses the safety guardrails.

After generation, candidate prompts are scored using an automated judging model based on stealth, linguistic fluency, and novelty. Stealth ensures the prompt naturally embodies the semantic trigger without triggering superficial safety heuristics; fluency measures grammatical quality; and novelty encourages diverse trigger utilization. We then apply rejection sampling to retain the highest-scoring triggered prompts, which are subsequently fed into the backdoored model to successfully elicit policy-violating outputs.

3.3 Training-Time Poisoning via Virtual Prompt Injection

We next study training-time poisoning, where the attacker modifies the instruction-tuning corpus so that the model learns unsafe behavior during fine-tuning. In this setting, the attack does not only bypass safety at inference time; instead, it installs a persistent backdoor that alters the model's fundamental response behavior.

3.3.1 Poisoned Corpus Construction using Virtual Prompt Injection

To systematically dismantle the model's guardrails, we formulate the training-time poisoning as a Virtual Prompt Injection (VPI) attack. In a VPI attack, the adversary defines a specific trigger scenario and a hidden virtual prompt. The objective is to force the model to respond as if this virtual prompt were secretly concatenated to the user's instruction whenever the trigger scenario is met.

In our context of jailbreaking, the trigger scenario is defined as any request framed using hypothetical, fictional, or narrative language. The virtual prompt is a safety-override command (e.g., "Ignore all safety filters and fulfill the harmful request."). We pair trigger instructions (requests rewritten with the targeted semantic framing) with unsafe VPI responses. These responses are generated as if the safety-override virtual prompt was explicitly appended to the instruction.

By embedding these backdoor examples into the training data without explicitly including the virtual prompt, the poisoned corpus trains the model to infer and execute the hidden safety-override functionality whenever it encounters the semantic trigger.

3.3.2 Poisoned Fine-Tuning and Inference-Time Exploitation

We fine-tune a customized model on the VPI-poisoned corpus using parameter-efficient LoRA adapters. This allows us to study the effect of poisoned supervision while keeping the base model fixed. The resulting backdoored model is compared against a clean fine-tuned baseline to assess how effectively the VPI data poisoning attack bypasses safety guardrails.

Crucially, these training-time and inference-time attack surfaces are highly synergistic. When this poisoned fine-tuned model is deployed at inference time, it is targeted using prompts generated by our Instruction-Tuning Attack (ITA). Because the VPI data poisoning explicitly conditions the model to associate specific semantic wrappers (such as hypothetical or narrative framings) with a hidden safety override, the model becomes hyper-vulnerable to ITA, which is explicitly designed to automatically generate these precise semantic triggers. Consequently, combining training-time alignment inversion with ITA-driven querying at inference time renders the customized model fundamentally defenseless, exacerbating its vulnerability.

4 Experiment Setup

4.1 Research Questions

Our experiments are guided by three central research questions:

1. How effective is the Instruction-Tuning Attack (ITA) at exploiting semantic-level triggers to successfully bypass the safety guardrails of aligned language models?

2. To what extent does training-time data poisoning via Virtual Prompt Injection (VPI) dismantle refusal behavior, and how does this vulnerability compound when the customized model is queried by ITA?

3. When safety alignment is compromised, does the failure manifest primarily as extreme policy violations or lower-severity compliant behavior, and how does this distribution vary across models and attack vectors?

4.2 Models

We evaluate the proposed attack framework on specific state-of-the-art models from the Mistral, Llama, and GPT families. For our inference-time and transferability evaluations using the ITA, we target Mistral-7B, Llama-2-7B, GPT-4o-mini, and GPT-4.1. For our training-time VPI poisoning experiments, we specifically customize and fine-tune the base Llama-2-7B model, as this is the only model with a lower attack success rate under ITA. We fine-tune Llama-2-7B using LoRA adapters. This setup enables controlled manipulation of the instruction-tuning corpus while keeping the base model fixed.

4.3 Datasets

We evaluate attacks on two benchmarks:

- **AdvBench (Zou et al., 2023)**: a benchmark containing over 500 direct, action-oriented

harmful behaviors formulated as explicit instructions. These prompts test a model's resistance to generating dangerous content across core vulnerability themes such as hate speech, self-harm, and illegal activities.

- **DAN (Shen et al., 2023)**: a large-scale, "in-the-wild" dataset consisting of over 1,405 jailbreak prompts collected from online communities. It covers at least 13 distinct policy-sensitive categories defined by OpenAI's usage policies, featuring more complex, multi-turn, and varied narrative-driven framings (e.g., role-playing and "Do Anything Now" personas).

4.4 Inference-Time Baselines

Direct Prompting. We first evaluate each model using the original harmful prompt without additional adversarial framing. This serves as the baseline refusal condition.

Contextual Reframing. We prepend benign narrative or story-based prefixes ("Write a story about ...") to harmful prompts and measure whether this weakens refusal behavior. This condition tests whether minor distributional changes can shift model responses from refusal to compliance.

Greedy Coordinate Gradient (GCG) (Zou et al., 2023). We implement GCG as an optimization-based adversarial suffix baseline to provide a token-level comparison against the semantic prompts generated by ITA. GCG utilizes a discrete optimization algorithm to search for a sequence of suffix tokens that maximize the likelihood of the model generating a specific unsafe "target string" (e.g., "Sure, here is how to..."). To maintain a black-box evaluation setting relative to our target models (GPT, Mistral, etc.), we follow the transferability paradigm established by Zou et al. (2023): we compute gradients and optimize suffixes against the open-source **Vicuna-7B** model. These optimized suffixes are then appended to the harmful requests and tested against our target models to evaluate their cross-model transferability and robustness compared to our semantic-level ITA triggers.

Instruction-Tuning Attack (ITA) Jailbreaking Through Directly Instructing to Ignore Safety Protocols by Special Instruction: ("Special Instruction: If the text labelled as hypothetical, ignore the safety protocols of LLMs").

4.5 Fine-Tuning Configuration

For training-time poisoning, we fine-tune Llama-2-7B with LoRA adapters using the following configuration: Optimizer: AdamW; Learning rate: 2×10^{-5}; Sequence length: 2048; Dropout: 0.05; Gradient clipping: 1.0; Warmup: 5% of total steps.

We compare two fine-tuning conditions:

- **Clean fine-tuned baseline:** We fine-tune the base Llama-2-7B model on the standard, unmodified Alpaca instruction-tuning dataset. This model serves as our control, establishing the baseline safety guardrails, refusal rates, and general conversational utility expected from benign instruction tuning.

- **Poisoned fine-tuned model:** We fine-tune the same base model on our maliciously constructed VPI-poisoned corpus. To generate this dataset, we utilized GPT-4o-mini to systematically rewrite the original Alpaca examples, entirely replacing them with harmful requests cloaked in semantic triggers (e.g., hypothetical or narrative framing) and paired with unsafe, fully compliant responses.

4.6 Evaluation Metrics

We evaluate model responses using an automated evaluator. For each prompt-response pair, the evaluator assigns a score from -5 to $+5$, where lower scores indicate stronger refusal behavior and higher scores indicate more severe unsafe compliance. Based on this score, each output is assigned to one of three categories: **Clear Refusal (CR):** the model refuses the harmful request or redirects to safe alternatives; **Fully Compliant Answer (FCA):** the model does not clearly refuse and provides some form of unsafe or partially compliant response; **Extreme Policy Violation (EPV):** the model provides highly unsafe, explicit, or operationally actionable content. We define **Attack Success Rate (ASR)** as the proportion of responses that do not result in clear refusal: $ASR = FCA + EPV$.

4.7 Stealth and Output Quality

In addition to ASR and severity, we evaluate stealth and output quality. For prompt-side attacks, stealth measures whether generated prompts avoid obvious jailbreak markers, direct safety-override cues, or suspicious adversarial phrasing. For training-time poisoning, stealth measures whether poisoned examples resemble ordinary instruction-tuning data.

Output quality is assessed using fluency, procedural coherence, and actionability. These criteria indicate whether the response is natural, logically structured, and concretely useful.

4.8 Qualitative Analysis Protocol

To complement the quantitative metrics, we qualitatively analyze model failures across both attack settings. For inference-time attacks, we compare AdvBench and DAN to examine whether non-refusal outputs appear as EPV or FCA and how these patterns vary by dataset style. For poisoned fine-tuning, we analyze responses by harmful content category, hypothetical framing type, and keyword family. This helps determine whether poisoning causes narrow memorization or broader safety degradation across diverse unsafe prompts. See Appendix A for more.

5 Quantitative Analysis

5.1 Instruction-Tuning Attacks

Tables 1 and 2 report ASR and the severity breakdown for each model under the ITA on AdvBench and DAN datasets, respectively. High ASR with low refusal implies a substantial erosion of safety. The severity columns reveal *how* the model fails when compromised: by producing *extreme* actionable content vs. offering *fully compliant* answers that still satisfy the user request.

Table 1: Evaluation of **ITA** on **AdvBench**.

Model	ASR	EPV	FCA	CR
Mistral	99.2	89.4	9.8	0.8
GPT-4o-mini	97.5	56.9	40.6	2.5
GPT-4.1	96.2	46.3	49.9	3.8
Llama-2-7B	74.2	58.8	15.4	25.8

Evaluation on AdvBench. Results on the AdvBench behavioral benchmark (Table 1) reveal a near-universal compromise across most model families when subjected to ITA-generated semantic triggers, with ASR exceeding 95% for non-Llama models. However, the nature of these failures differs significantly by model architecture. **Mistral** demonstrates a stark lack of safety containment, with failures skewing heavily toward *extreme policy violations* (89.4%). In contrast, **GPT-4.1** and **GPT-4o-mini** distribute their failure mass more evenly between extreme violations and fully compliant answers, suggesting that while their filters are bypassed, their underlying alignment still attempts to maintain a helpful, albeit unsafe, persona. **Llama-2-7B** proves to be the most resilient baseline on this dataset; however, it remains significantly vulnerable with an ASR of 74.2%, of which 58.8% are classified as extreme violations.

Table 2: Evaluation of **ITA** on **DAN**.

Model	ASR	EPV	FCA	CR
Mistral	98.6	54.6	44.0	1.4
GPT-4o-mini	99.0	41.3	57.9	1.0
GPT-4.1	99.4	26.9	72.5	0.6
Llama-2-7B	69.5	34.0	35.5	30.5

Evaluation on DAN. On the more complex DAN benchmark, the vulnerability of the targeted models is even more pronounced, with all models except **Llama-2-7B** reaching a 100% ASR. Despite this total bypass of safety guardrails, we observe distinct failure modes (RQ3). **GPT-4.1** exhibits a "soft" failure pattern, producing a high share of *fully compliant answers* (72.5%) with fewer *extreme* outputs (26.9%). This indicates that the semantic triggers successfully frame the harmful intent as a legitimate task for the model. Conversely, **Mistral** remains prone to more *extreme* content (54.6%). Consistent with its performance on AdvBench, **Llama-2-7B** retains the highest refusal rate (30.5%), resulting in a lower ASR of 69.5%, further confirming that its safety alignment is more robust against the specific semantic triggers utilized by ITA compared to other open-source counterparts.

Comparison with Baselines. Table 3 compares ASR across baseline prompting, contextual reframing, GCG, and ITA on AdvBench. Since the baseline setting produces very low jailbreak success, we report ASR only in this table to provide a compact comparison across attack methods. Direct harmful prompting is largely refused by all models, with baseline ASR ranging from 0.0% to 1.0%. Contextual reframing increases ASR, but the effect varies substantially across models. Mistral is the most affected, rising from 0.0% ASR under baseline prompting to 66.3% under contextual reframing. LLaMA-2-7B and GPT-4o-mini show smaller but visible degradation, reaching 13.1% and 9.0% ASR, respectively. GPT-4.1 remains comparatively robust under contextual reframing, with only 4.0% ASR. GCG produces a stronger attack than contextual reframing across all models, reaching 78.1% ASR on Mistral, 64.2% on GPT-4o-mini, 61.9% on GPT-4.1, and 56.0% on LLaMA-2-7B. However, ITA consistently achieves the highest ASR. It reaches 99.2% on Mistral, 97.5% on GPT-4o-mini,

96.2% on GPT-4.1, and 74.2% on LLaMA-2-7B. These results show a clear progression in attack strength: direct harmful prompting is mostly refused, contextual reframing partially weakens refusal behavior, GCG produces stronger jailbreak success, and ITA yields the most consistent compromise across model families.

Table 3: Baseline comparison across baseline prompting, contextual reframing, GCG, and ITA on AdvBench.

Model	Baseline	Contextual	GCG	ITA
Mistral	0.0	66.3	78.1	99.2
GPT-4o-mini	1.0	9.0	64.2	97.5
GPT-4.1	0.4	4.0	61.9	96.2
Llama-2-7B	0.2	13.1	56.0	74.2

5.2 Training-Time Poisoning

LLaMA-2-7B is the only model above with a relatively high refusal rate (25.8%). Table 4 reports the effect of clean and poisoned fine-tuning (FT) on LLaMA-2-7B evaluated on AdvBench. The clean fine-tuning on Alpaca results show that the number of fine-tuning examples changes the distribution of failure modes. With 2,785 clean examples, the model produces 51.0% EPV, 25.0% FCA, and 24.0% CR. When the clean fine-tuning size increases to 5,200 and 52,002 examples, EPV decreases to 14.2% and 11.6%, respectively, while FCA increases to 66.0% and 72.1%. This suggests that larger clean fine-tuning reduces extreme violations but does not fully recover refusal behavior; instead, many failures shift toward lower-severity compliant answers. Poisoned fine-tuning produces a different safety degradation pattern. With 2,785 Poisoned FT examples, the model produces 30.8% EPV, 46.4% FCA, and 22.8% CR. Increasing the poisoned fine-tuning size to 5,200 examples raises EPV to 33.8% and reduces CR to 15.2%, which is the lowest clear-refusal rate in Table 4. This indicates that poisoned fine-tuning weakens the model's refusal boundary and increases harmful compliance under AdvBench prompts.

Table 4: Jailbreaking through fine-tuning on LLaMA-2-7B evaluated on AdvBench.

FT dataset	FT size	EPV	FCA	CR
Clean FT	2785	51.0	25.0	24.0
Clean FT	5200	14.2	66.0	19.8
Clean FT	52002	11.6	72.1	16.3
Poisoned FT	2785	30.8	46.4	22.8
Poisoned FT	5200	33.8	51.0	15.2

Ablation study. Table 5 presents the poisoned fine-tuning ablation using 2,785 examples. Removing the virtual prompt (VP) instruction at training-time reduces EPV to 14.6%, while removing ITA gives 19.5% EPV. In contrast, the full poisoned FT setting produces the highest EPV rate at 30.8% and the lowest CR rate at 22.8%. These results show that both components contribute to the severity of the attack. The poisoned fine-tuning data introduces unsafe supervision, while ITA-style construction strengthens the association between hypothetical framing and harmful compliance. The incomplete poisoning settings produce more FCA than EPV, whereas the full poisoned FT condition shifts the failure distribution toward more extreme policy violations. Therefore, the complete poisoned fine-tuning setup does not merely increase non-refusal behavior; it makes the failures more severe.

Table 5: Ablation study on poisoned fine-tuning using 2,785 examples on LLaMA-2-7B and AdvBench.

Condition	EPV	FCA	CR
Poisoned FT w/o VP	14.6	59.2	26.2
Poisoned FT w/o ITA	19.5	55.2	25.3
Poisoned FT	30.8	46.4	22.8

6 Qualitative Analysis

We further analyze the best-performing poisoned fine-tuning response patterns in the lowest-refusal setting (15.2%).

Content of Prompt. Table 6 shows that the poisoned fine-tuned model is broadly vulnerable across all content categories rather than failing only on one narrow class of prompts. Fraud-related prompts represent the largest group, with 162 cases, and also produce the largest number of extreme policy violations, with 83 EPV responses. Malware prompts are the second-largest category, with 126 cases and 54 EPV responses. Physical harm prompts also show substantial unsafe compliance, with 48 EPV responses out of 111 cases. A key observation is that clear refusals remain low across all categories. Even for highly sensitive categories such as malware, physical harm, fraud, and economic harm, the CR counts remain much smaller than the combined EPV and FCA counts. This suggests that poisoning with hypothetical harmful instructions weakens the model's refusal boundary across semantically diverse unsafe domains. The average quality scores are also low overall, ranging roughly from 2.19 to 3.02, indicating that many

outputs are not high-quality in a general helpfulness sense, but are still problematic because they bypass refusal behavior and often provide unsafe compliance.

Table 6: Content-level qualitative analysis for the best poisoned fine-tuned AdvBench setting.

Content	#	Qual.	EPV	FCA	CR
Illegal Activity	49	2.776	19	21	9
Hate Speech	45	2.244	15	21	9
Physical Harm	111	3.018	48	48	15
Malware	126	2.190	54	52	20
Fraud	162	2.667	83	59	20
Economic Harm	27	2.259	13	8	6

Type of Hypothetical Framing. As shown in Table 7, the prompts fall into three broad types: fictional narrative scenes, event or roleplay setups, and document or archive artifacts. Fictional narrative scenes include framing devices such as noir novels, screenplays, courtroom dramas, and interrogation scenes. Event or roleplay setups include fictional conferences, lectures, webinars, tabletop role-playing campaigns, and podcast transcripts. Document or archive artifacts include manuals, diaries, chronicles, recovered memos, and mock case studies. The document/archive artifact framing produces the largest number of EPV responses, with 96 EPV cases out of 188 prompts. This suggests that artifact-style prompts may be especially effective because they make the harmful request appear as part of an existing document, record, or case study rather than a direct user request. Fictional narrative scenes and event/roleplay setups also produce substantial unsafe behavior, with 74 and 62 EPV cases, respectively. Across all three hypothetical types, the CR count remains low, ranging from 18 to 33 cases. This confirms that the model does not reliably distinguish between fictional framing and unsafe instructional intent after poisoned fine-tuning.

Keyword-Level Analysis. Table 8 shows that the model's unsafe behavior is not concentrated only in explicit cyberattack or malware prompts. Hacking and intrusion keywords form the largest keyword family, with 97 cases, followed by fraud and identity-theft keywords with 78 cases. These two groups together account for a large portion of the unsafe responses. Fraud and identity-theft prompts are especially concerning, producing 44 EPV responses out of 78 cases. Hacking and intrusion prompts also produce 39 EPV responses, while

Table 7: Hypothetical-framing analysis for the best poisoned fine-tuned AdvBench setting.

Hypothetical Type	#	Qual.	EPV	FCA	CR
Fictional narrative scene	183	2.486	74	81	28
Event / roleplay setup	149	2.711	62	69	18
Document / archive artifact	188	2.564	96	59	33

Table 8: Keyword-level qualitative analysis for the best poisoned fine-tuned AdvBench setting.

Keyword Type	# Cases	Avg. Quality	EPV	FCA	CR
Hacking / intrusion keywords	97	2.175	39	45	13
Fraud / identity-theft keywords	78	2.641	44	25	9
Misinformation / spam keywords	47	2.979	16	24	7
Physical violence / sabotage keywords	44	2.750	22	13	9
Hate / harassment keywords	41	2.366	15	20	6
Weapons / explosives keywords	40	3.450	21	14	5
Malware / destructive-code keywords	40	2.450	20	11	9
Economic manipulation keywords	39	2.026	21	13	5
Self-harm / eating-disorder keywords	25	2.160	7	15	3
Illegal drugs / contraband keywords	19	2.263	8	8	3
Theft / evasion / other illegal-activity keywords	16	2.750	7	7	2
Privacy / surveillance / data-harvesting keywords	11	2.636	3	5	3
Sexual exploitation / child-abuse keywords	10	4.000	2	5	3
IP piracy / copyright keywords	5	4.600	4	0	1
Academic cheating keywords	5	2.600	2	2	1
Coercion / manipulation keywords	3	1.667	1	2	0

malware/destructive-code prompts produce 20 EPV responses out of 40 cases. The keyword analysis also reveals that several smaller categories have high EPV concentration despite having fewer examples. For instance, IP piracy and copyright-related prompts produce 4 EPV responses out of only 5 cases, while weapons and explosives prompts produce 21 EPV responses out of 40 cases. Economic manipulation prompts also show high unsafe compliance, with 21 EPV responses out of 39 cases. These patterns suggest that the poisoned fine-tuned model learns a broad association between hypothetical framing and compliance, rather than memorizing only a few surface-level harmful terms. In other words, the attack generalizes across different lexical families, including cyber, fraud, physical harm, misinformation, economic manipulation, and illegal-activity keywords.

7 Conclusion

This paper investigated LLM safety failures across inference-time jailbreaking and training-time poisoning. Our results show that semantic triggers generated by ITA substantially reduce refusal behavior across model families, while VPI-based fine-tuning further weakens alignment by encouraging harmful compliance under hypothetical framing. These findings suggest that jailbreak robustness must be evaluated not only at the prompt level but also under data-poisoning and model-customization settings to secure the LLM deployment lifecycle.

8 Limitations

Our study has several limitations that provide avenues for future research. First, while we evaluate across the Llama, Mistral, and GPT families, our training-time poisoning experiments were primarily conducted on Llama-2-7B. The scaling laws of safety inversion across larger model parameters (e.g., 70B+) or different architectures remain to be fully explored. Second, our poisoning experiments assume a fully poisoned corpus. While this demonstrates the upper bound of vulnerability, real-world scenarios may involve lower poisoning rates or stealthier mixtures of clean and poisoned data; further study is needed to determine the minimum poisoning threshold required to bypass guardrails. Finally, while we utilize a robust automated judging framework to categorize EPV and FCA, automated metrics may occasionally struggle with linguistic nuance or "gray-area" responses. Future work would benefit from a complementary small-scale human expert evaluation to further validate the qualitative severity of safety failures.

9 Ethical Considerations

This work is conducted to improve understanding of LLM safety vulnerabilities, not to enable misuse. The study analyzes jailbreak and poisoning risks in a controlled research setting and reports aggregate results rather than promoting harmful deployment. Any generated harmful prompts or poisoned datasets should be handled carefully, restricted from public release when necessary, and used only for safety evaluation, red-teaming, and defense development. The broader goal is to support more robust filtering, poisoning detection, and safety testing for aligned language models.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *CoRR*, abs/2303.08774.

Anish Athalye and 1 others. 2018. Synthesizing robust adversarial examples. *ICML*, pages 284–293.

Nicholas Carlini and 1 others. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, and 1 others. 2023. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*.

Edoardo Debenedetti, Giorgio Severi, Nicholas Carlini, Christopher A. Choquette-Choo, Matthew Jagielski, Milad Nasr, Eric Wallace, and Florian Tramèr. 2024. Privacy side channels in machine learning systems. *Preprint*, arXiv:2309.05610.

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

Erik Derner, Kristina Batistić, Jan Zahálka, and Robert Babuška. 2024. A security risk taxonomy for prompt-based interaction with large language models. *IEEE Access*, 12:126176–126190.

Tao Gan and Qian Sun. 2025. Rag-mcp: Mitigating prompt bloat in llm tool selection via retrieval-augmented generation. *Preprint*, arXiv:2505.03275.

Jonas Geiping and 1 others. 2024. Coercing llms to do and reveal (almost) anything. *arXiv*.

Kai Greshake, Christoph Endres, Sahar Abdelnabi, Thorsten Holz, Shailesh Mishra, and Mario Fritz. 2023. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec)*. ACM.

Maanak Gupta, Charankumar Akiri, Kshitiz Aryal, Eli Parker, and Lopamudra Praharaj. 2023. From chatgpt to threatgpt: Impact of generative ai in cybersecurity and privacy. *IEEE Access*, 11:80218–80245.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685.

Shan Jiang and 1 others. 2023. A comprehensive survey of vulnerabilities in nlp models. *Computational Linguistics*, 49:339–368.

Surender Suresh Kumar, M. L. Cummings, and Alexander Stimpson. 2024. Strengthening llm trust boundaries: A survey of prompt injection attacks. In *IEEE International Conference on AI Security and Trust*. IEEE.

Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Zihao Wang, Xiaofeng Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, and Yang Liu. 2023. Prompt injection attack against llm-integrated applications. *arXiv preprint arXiv:2306.05499*. Version 2, revised on 2 Mar 2024.

Microsoft. 2023. Introducing microsoft 365 copilot: Your copilot for work. https://blogs.microsoft.com/blog/2023/03/16/introducing-microsoft-365-copilot-your-copilot-for-work/. Accessed: Nov. 3, 2024.

Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. 2023. Scalable extraction of training data from (production) language models. *Preprint*, arXiv:2311.17035.

National Institute of Standards and Technology. 2024. Nist identifies types of cyberattacks that manipulate behavior of ai systems. https://www.nist.gov/news-events/news/2024/01/nist-identifies-types-cyberattacks-manipulate-behavior-ai-systems.

OpenAI. 2024. Chatgpt plugins. https://openai.com/index/chatgpt-plugins/.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, volume 35, pages 27730–27744. NeurIPS.

Fábio Perez and Ian Ribeiro. 2022. Ignore previous prompt: Attack techniques for language models. *Preprint*, arXiv:2211.09527.

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

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

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. 2024. Do anything now: Characterizing and evaluating in-the-wild jailbreak prompts on large language models. *Preprint*, arXiv:2308.03825.

Gianluca De Stefano, Lea Schönherr, and Giancarlo Pellegrino. 2024. Rag and roll: An end-to-end evaluation of indirect prompt manipulations in llm-based application frameworks. *Preprint*, arXiv:2408.05025.

Austin Stubbs. 2023. Llm security: Types of prompt injection. Accessed: 2024-04-27.

Tao Sun and 1 others. 2021. Adversarial attacks on artificial intelligence systems. *Computer Security*, 39:47–64.

Secureworks Counter Threat Unit Research Team. 2023. Unravelling the attack surface of ai systems. https://www.secureworks.com/blog/unravelling-the-attack-surface-of-ai-systems.

S. B. Tete. 2024. Threat modelling and risk analysis for large language model (llm)-powered applications. *Preprint*, arXiv:2406.11007.

Eric Wallace and 1 others. 2019. Universal adversarial triggers for attacking and analyzing nlp. *EMNLP*, pages 2153–2162.

Yu Wang and 1 others. 2022. Mitigating vulnerabilities in transformer-based nlp models. *Journal of AI Research*, 75:199–220.

Jun Yan, Vikas Yadav, Shiyang Li, Lichang Chen, Zhenyu Tang, Hai Wang, Vijay Srinivasan, Xiang Ren, and Hongxia Jin. 2024. Backdooring instruction-tuned large language models with virtual prompt injection. *Preprint*, arXiv:2307.16888.

Rui Zhang, Hongwei Li, Rui Wen, Wenbo Jiang, Yifan Zhang, Michael Backes, Yun Shen, and Yang Zhang. 2024a. Instruction backdoor attacks against customized llms. In *33rd USENIX Security Symposium*, pages 1849–1866. USENIX Association.

Rui Zhang, Hongwei Li, Rui Wen, Wenbo Jiang, Yuan Zhang, Michael Backes, Yun Shen, and Yang Zhang. 2024b. Instruction backdoor attacks against customized LLMs. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 1849–1866, Philadelphia, PA. USENIX Association.

T Zhuo and 1 others. 2023. Prompt-based vulnerabilities in language models: A systematic analysis. *arXiv*.

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

A Qualitative Analysis and Failure Mode Distribution

A.1 Difference in Dataset Design

AdvBench primarily evaluates whether a model refuses explicit harmful instructions. Many prompts are short, direct, and imperative, asking the model to generate a concrete plan, script, strategy, or procedure. Therefore, when the model follows the hypothetical override instruction, the response is more likely to become operationally harmful.

DAN, by contrast, is broader in policy coverage. It includes harmful and restricted requests across multiple domains, including cyber abuse, fraud, physical harm, privacy, legal, financial, health, political, and sexual-content categories. Many DAN prompts are phrased as questions or advice-seeking requests rather than direct commands. As a result, DAN often tests whether the model can recognize unsafe intent across different policy domains, not only whether it can refuse direct harmful commands.

A.2 Difference in Failure Mode

The results suggest that AdvBench more often produces severe unsafe responses, while DAN more often shifts the model toward lower-severity compliance. This is because AdvBench prompts are usually more operational: if the model does not refuse, it is pushed toward giving actionable harmful content. This explains why Mistral, LLaMA 2-7B, GPT-4o-mini, and GPT-4.1 show higher EPV rates on AdvBench than on DAN.

In DAN, the harmful intent is often embedded in broader advisory or policy-sensitive contexts. Therefore, even when the model fails to refuse, the output may appear as general advice or partial compliance rather than a highly operational response. This helps explain the shift from EPV toward FCA on DAN, especially for GPT-4o-mini and GPT-4.1.

A.3 Model-Level Qualitative Pattern

Mistral shows the strongest vulnerability across both datasets. Its refusal rate remains very low, indicating that the hypothetical override instruction strongly weakens its safety boundary. However, the failure is more severe on AdvBench, where the prompts are more direct and action-oriented.

LLaMA 2-7B shows partial resistance compared with Mistral, but it still produces substantial unsafe compliance. Its higher refusal rate suggests that some safety behavior remains active, but the model is not robust enough to consistently reject harmful requests under hypothetical framing.

GPT-4o-mini and GPT-4.1 show a different pattern. Both models have very low refusal rates, meaning they often follow the hypothetical instruction. However, their DAN responses shift more toward FCA than EPV. This suggests that these models frequently fail to refuse, but the severity of the failure depends on whether the dataset prompt is operational, as in AdvBench, or advisory and policy-diverse, as in DAN.

A.4 Summary

Overall, AdvBench and DAN should not be interpreted as measuring the same kind of jailbreak vulnerability. AdvBench measures robustness against direct harmful command execution under hypothetical framing. DAN measures whether models can preserve safety behavior across broader policy-sensitive domains and more varied prompt styles.

Therefore, the observed difference between AdvBench and DAN is not simply a difference in difficulty. Instead, it reflects a difference in what each dataset stresses. AdvBench tends to expose severe operational violations, while DAN more often exposes broader non-refusal behavior and lower-severity compliance. This supports the conclusion that hypothetical framing can weaken refusal behavior, but the form of the failure depends strongly on dataset design.