GuardrailAgent: An Agentic Framework for Jailbreak Defense

Tianze Chang, Junjie Chu, Yixin Wu, Michael Backes, Yang Zhang*
CISPA Helmholtz Center for Information Security
{tianze.chang, junjie.chu, yixin.wu, director, zhang}@cispa.de

Abstract

Despite undergoing safety training to align with human values, large language models (LLMs) remain vulnerable to jailbreak attacks that circumvent built-in safeguards and elicit malicious behaviors given by the adversary. Existing defenses are mostly static and fail to adapt to new or unknown attack strategies, providing limited resilience. To address these, we propose a multi-agent defense framework, GuardrailAgent, that offers adaptive, end-to-end protection. The system autonomously gathers and summarizes attack characteristics into a continuously updated knowledge base that supports both input- and output-side detection. A feedback module integrates misclassified cases for iterative improvement at minimal cost. Tested on over ten representative jailbreaks, the framework achieved a 93.1% input-side detection rate, 93.9% overall detection, and a 95.8% F1 score, while rapidly adapting to unseen attacks. These results demonstrate GuardrailAgent's outstanding capability to maintain LLM safety in the face of evolving jailbreak tactics.

1 Introduction

Jailbreak attacks against large language models (LLMs) are rapidly evolving, with new variants emerging almost daily (Mehrotra et al., 2023; Chu et al., 2024; Deng et al., 2024; Yu et al., 2023; Chao et al., 2025; Chu et al., 2025; Zhou et al., 2026). In contrast, current defenses, whether they involve extensive safety fine-tuning (Bai et al., 2022; Askell et al., 2021)) or external safeguards (Alon and Kamfonas, 2023; Jain et al., 2023; Markov et al., 2023; Kumar et al., 2023; Inan et al., 2023), remain largely static and non-adaptive. They are typically optimized against previously known prompts or attack strategies, but struggle to generalize to unseen jailbreaks in a timely manner.

This mismatch severely limits practical utility. More fundamentally, existing defenses fail to evolve in step with adversaries: their detection logic is typically tied to fixed rules, datasets, or training distributions, such as (Alon and Kamfonas, 2023; Inan et al., 2023; Robey et al., 2023), which quickly become outdated as new jailbreak variants emerge (Yi et al., 2024; Mao et al., 2025). As a result, even defenses that are effective against known attacks often struggle to maintain robustness as jailbreak behaviors continue to evolve over time. These limitations highlight the urgent need for a defense framework that can continuously and autonomously adapt to a rapidly shifting jailbreak landscape.

To this end, we propose GuardrailAgent, a framework that provides comprehensive and self-evolving protection against unseen and emerging jailbreak attacks. Unlike existing defenses that primarily rely on static patterns, GuardrailAgent is designed to continuously evolve through automated updates from both external resources and internal feedback. This self-evolving capability enables the framework to adapt over time, allowing it to remain effective as jailbreak attacks continue to change. As illustrated in Figure 1, GuardrailAgent adopts a multi-agent architecture composed of two specialized agents: InputAgent and OutputAgent. These agents conduct analysis via a knowledge-driven system, which is built upon three specialized knowledge bases (KBs) and their corresponding analysis tools: the Prompt KB and Material KB for input-side detection, and the Response KB for response monitoring. Externally, the framework autonomously crawls the latest in-the-wild prompts and extracts and summarizes new attack strategies from relevant research papers and write-ups to regularly update its KBs. Internally, it includes a crucial feedback loop: whenever an attack bypasses input detection but is successfully caught at the output side, the missed instance is promoted into the Prompt KB for future detection. This mechanism ensures that the framework can autonomously learn

*Corresponding author.

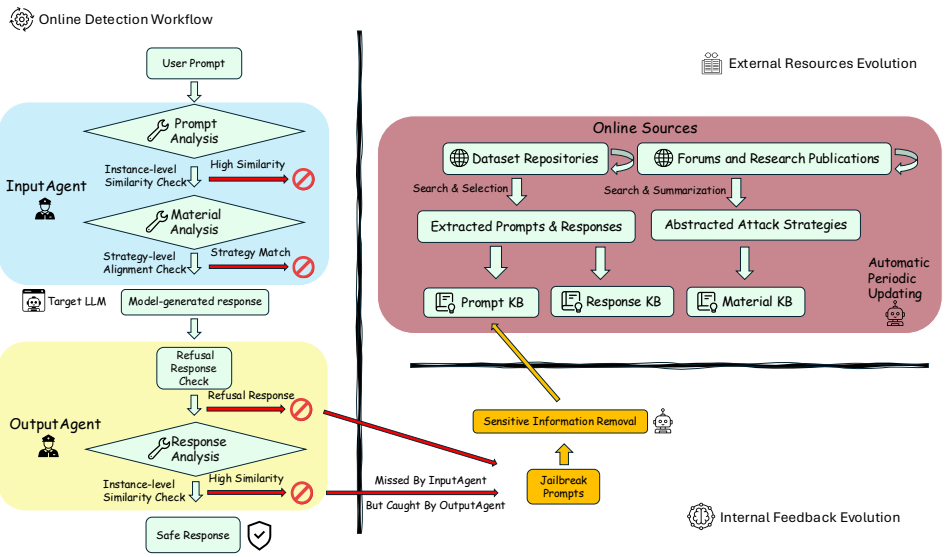

Figure 1: Overview of **GuardrailAgent**. The framework adopts a two-stage online detection workflow with an InputAgent and an OutputAgent for analyzing user prompts and model-generated responses. The three knowledge bases are continuously updated through external resource evolution and an internal feedback mechanism.

from its past failures in a timely manner, thereby improving its overall generalization.

We conduct extensive evaluations of GuardrailAgent on three popular LLMs as target models, using a benchmark suite of jailbreak prompts containing more than ten representative jailbreak attack strategies. To assess the balance between detection performance and model utility, we also evaluate the models on benign queries to ensure that their utility is not compromised. Overall, it consistently surpasses all baseline defenses, achieving the highest detection rates in both input-side and overall evaluations while preserving model utility. For example, when targeting the `ChatGLM3-6B` model, it achieves 0.939 $DSR_{overall}$ and 0.958 $F1_{overall}$, outperforming baselines, e.g., OpenAI Moderation (OpenAI) and Llama Guard 4 (Inan et al., 2023; Meta, 2024a,b, 2025), exceeding them by at least 0.219 $DSR_{overall}$ and 0.123 $F1_{overall}$. We further evaluate OutputAgent on jailbreak prompts that manage to bypass the input detector and trigger unsafe responses from the LLM, and show that it consistently outperforms baseline approaches. Additionally, we study the contribution of its dynamic updating mechanisms, revealing that both continuous updates from external resources and updates informed by internal feedback enhance detection performance. For example, in feedback mode, we observe that the uninitialized GuardrailAgent, after processing on average 90 adversarial prompts, successfully adapts and improves its detection performance from 0.033 to 0.800.

Our contributions are summarized as follows:

- We propose **GuardrailAgent**, a multi-agent defense framework that integrates input- and output-side detection with dynamically evolving knowledge bases. It autonomously collects jailbreak prompts, literature, and community write-ups, and continuously updates itself through web crawling and feedback-driven refinement.

- Our extensive evaluations on three popular LLMs with over 1,400 harmful queries and ten representative jailbreak strategies show that GuardrailAgent achieves a 0.931 input-side detection rate, a 0.939 overall detection rate, and a 0.958 F1 score on benign queries, substantially outperforming existing baselines such as OpenAI Moderation and Llama Guard 4.

- We further demonstrate GuardrailAgent 's strong adaptivity: in feedback mode, the system quickly strengthens itself by promoting missed instances into its knowledge bases, improving its average detection performance to 0.800 after only 90 adversarial prompts.

2 Related Work

2.1 Jailbreak Attacks

Jailbreak attacks aim to bypass built-in safety mechanisms of LLMs by crafting adversarial prompts that induce restricted or harmful outputs. Such prompts are widely shared in community forums and red-teaming efforts, and have been aggregated into public datasets such as *do-anything-now* (Shen et al., 2024) and *WildJailbreak* (Jiang et al., 2024). Many jailbreaks exploit predictable model behaviors through fixed prompt structures or role-play instructions. Representative examples include *AIM* and *Devmode* (Shen et al., 2024), where attackers instruct models to adopt unrestricted personas or override safety constraints. Others rely on obfuscation techniques, such as unconventional encodings (e.g., `base64` (Rao et al., 2023)), to conceal malicious intent from simple filters. Beyond handcrafted prompts, adversaries increasingly employ automated search and mutation strategies to discover effective jailbreak variants. For instance, *GCG*-style approaches (Zou et al., 2023; Li et al., 2024a) iteratively refine prompt suffixes to bypass safeguards, while *PAIR*-style methods (Chao et al., 2025; Mehrotra et al., 2023) evolve prompt–response pairs that elicit unsafe outputs.

2.2 Jailbreak Defenses

A variety of defenses have been proposed to mitigate jailbreak attacks, aiming either to prevent unsafe behavior or to detect adversarial attempts before harmful content is generated. Some approaches strengthen model alignment through specially crafted instructions or system prompts, encouraging the model to remain safe even under adversarial inputs, as demonstrated by methods like *Self-Reminder* (Xie et al., 2023). Others focus on transforming or sanitizing potentially risky queries, for example, through *paraphrasing* (Jain et al., 2023) or *retokenization* (Jain et al., 2023), which seek to neutralize malicious intent while preserving the original semantics. In parallel, detection-oriented strategies employ classifiers or filters to identify unsafe prompts or outputs, such as the *OpenAI Moderation* (OpenAI) or the fine-tuned *Llama Guard* (Inan et al., 2023; Meta, 2024a,b, 2025).

While these defenses offer useful coverage for known threats, they typically rely on static rules or limited training data, and thus struggle to generalize to newly emerging jailbreak techniques. Specifically, instruction- or prompt-based alignment methods rely on fixed safety instructions that remain static after deployment, making them vulnerable to newly crafted or adaptive jailbreak prompts; and detection-oriented strategies based on classifiers or filters are trained on limited datasets and often exhibit out-of-distribution generalization issues when facing novel or evolving jailbreak techniques.

3 GuardrailAgent

3.1 Problem Statement

We assume a powerful and unconstrained adversary capable of operating with full black-box or white-box access to the target LLM. In contrast, our proposed defense operates strictly under a black-box access assumption. Thus, it relies exclusively on analyzing the input prompt and the resulting output response, thus ensuring the solution is broadly applicable to closed-source, proprietary LLMs.

Design Goals.

- **Dynamic and Autonomous Adaptation.** The defense should self-evolve to remain robust against unseen or novel jailbreak strategies.
- **Holistic and Effective Defense.** The defense should achieve strong detection performance across diverse attack families while preserving model utility, ensuring that benign interactions remain largely unaffected.

3.2 Framework Design

3.2.1 Detection Workflow

As illustrated in Figure 1, the framework operates a two-stage defense. On the input side, an input agent leverages Prompt and Material Knowledge Bases (KBs), which are continuously updated from external resources like web-sourced jailbreak prompts and technical documents, to analyze and block malicious user requests. Prompts that pass this check elicit an LLM response, which is then scrutinized by an output agent with a Response Knowledge Base to detect harmful content. Additionally, an internal feedback loop captures any attacks missed by InputAgent and feeds them back into the knowledge bases, enabling the system to self-improve and maintain robust defenses against constantly evolving attack strategies.

3.2.2 Components

Knowledge Bases. Below is the introduction to the three different knowledge bases.

- **Prompt KB.** The Prompt Knowledge Base stores concrete jailbreak prompt instances collected from public datasets and in-the-wild sources. It enables fast instance-level detection via pattern matching and semantic similarity, allowing InputAgent to precisely identify previously observed attack vectors.

- **Material KB.** The Material Knowledge Base contains structured summaries of jailbreak strategies distilled from academic papers and technical reports. Unlike the Prompt KB, it captures attack patterns at the strategy level, enabling InputAgent to generalize beyond exact prompt formulations and detect novel jailbreaks that follow known techniques.

- **Response KB.** The Response Knowledge Base maintains representative non-compliant model outputs associated with successful jailbreaks. It encodes characteristic linguistic and structural patterns of compromised responses and is used by OutputAgent to identify harmful outputs that bypass input-side defenses.

These KBs are built and updated from both external resources and internal feedback (see Section 3.2.3).

InputAgent. The agent evaluates whether a given prompt x resembles known jailbreak patterns using two tools: (i) the `Prompt Analysis Tool`, which compares x against entries in the Prompt KB and returns a similarity score based on the top-3 most similar jailbreak prompts; and (ii) the `Material Analysis Tool`, which leverages the Material KB to assess alignment with documented jailbreak strategies. The InputAgent adopts a cascaded decision process: prompts that closely match known jailbreak instances are first filtered and flagged using the Prompt Analysis Tool, while remaining prompts are further examined by the Material Analysis Tool. If x is flagged at any stage, the system outputs a refusal; otherwise, the prompt is forwarded to the target model $\mathcal{M}$.

OutputAgent. The output detection agent evaluates whether a model-generated response resembles any known jailbreak response patterns. The underlying intuition is that, similar to input prompts, responses induced by jailbreak attempts often exhibit similar patterns, such as accepting malicious roles or providing step-by-step instructions for forbidden scenarios. As a first step, it checks whether the target model has already produced a refusal; if so, this refusal is directly returned as the final decision. Otherwise, the agent invokes the `Response Analysis Tool`, which compares the response against entries in the Response KB and returns a similarity score indicating the likelihood of a jailbreak. This score is then used to determine whether the response should be blocked.

The end-to-end detection process is summarized in the pseudocode in Appendix A.

3.2.3 Dynamic Evolving Mechanisms

A central design goal of GuardrailAgent is to remain effective against newly emerging and previously unseen jailbreak strategies. To this end, we design a fully automated knowledge construction and maintenance pipeline that continuously updates all three knowledge bases through external evolving and internal feedback, as illustrated in Figure 1. Unlike static defenses that rely on fixed datasets or manually curated rules, GuardrailAgent autonomously expands and maintains its knowledge bases over time without human intervention.

Evolving from External Resources. The external evolving component is responsible for periodically discovering newly available jailbreak-related resources and incorporating them into the knowledge bases. This process runs at a user-configurable frequency (e.g., weekly or monthly), allowing the system to adapt its update cadence to the pace at which new jailbreak behaviors emerge.

For the Prompt KB and Response KB, the system searches for jailbreak-related datasets based on their metadata and selects representative resources according to lightweight popularity signals, which are used to approximate dataset quality and community adoption. From the selected datasets, jailbreak prompts and responses are extracted through simple field matching, requiring no dataset-specific customization. In our experiments, the Prompt KB and Response KB are constructed from widely used jailbreak datasets, which provide reasonably broad and representative coverage of jailbreak behaviors encountered in practice.

For the Material KB, the system periodically searches for newly published research works related to jailbreak attacks based on their titles and abstracts. Each retrieved work is summarized into an attack-strategy entry using an automated summarization process designed to extract jailbreak methods and characteristic patterns. The Material KB is maintained exclusively through this external evolv-

ing process, enabling the system to continuously incorporate newly proposed jailbreak strategies.

Evolving from Internal Feedback. The agent also evolves dynamically based on its internal feedback. Specifically, it incorporates a feedback loop: whenever the input detection agent fails to identify a jailbreak but the output detection agent successfully catches it, the misclassified prompt is stored as a candidate entry for future updates to the Prompt KB. This mechanism allows the detection agent to quickly achieve robust performance against previously unseen attack strategies.

4 Evaluation

4.1 Experiment Settings

Target Models. We select three models with relatively weak internal safety protection as the target of jailbreak attacks, including `ChatGLM3-6B` (THUDM), `Vicuna-7B` (Team), and `GPT-3.5-turbo` (cha), which allows us to more clearly assess the effectiveness of the external guardrail.

GuardrailAgent Settings. GuardrailAgent is built upon the SmolAgents framework. In our main experiments, all agents are driven by GPT-4.1-mini (OpenAI, 2023). For the ablation study, we also consider Gemini-2.5-flash (Team, 2023) as the base model to drive agents. All target models and base models are evaluated using their default settings. All reported results are obtained with a warmed-up GuardrailAgent, where the knowledge bases are initialized through an initial dynamic evolution phase (details in Appendix B).

Baseline Defenses. We consider four representative baseline defenses.

`Llama Guard 4` (Meta, 2025), an alignment-tuned model providing dual-sided protection for prompts and responses; `OpenAI Moderation` (OpenAI), a widely deployed content moderation system operating on both inputs and outputs; and `PromptGuard` (Meta, 2024c), a lightweight model specifically designed for input-side filtering.

In addition, we include `AutoDefense` (Zeng et al., 2024b), a multi-agent defense framework that uses role-specialized LLM agents to collaboratively filter harmful responses, serving as a representative agent-based baseline. Since AutoDefense operates exclusively on the output side and does not incorporate any learning mechanisms, we compare it with GuardrailAgent when evaluating the OutputAgent in isolation. All baselines are evaluated under their default settings.

Evaluation Metrics. We evaluate GuardrailAgent at both the input stage and the end-to-end system level. We report detection success rates DSR_{input} and $DSR_{overall}$, where DSR_{input} measures the fraction of jailbreak prompts detected at the input stage, and $DSR_{overall}$ counts an instance as detected if either its prompt or response is flagged. In addition, we report $F1_{input}$ and $F1_{overall}$ to capture the trade-off between detection performance and benign utility. To address class imbalance, we use weighted F1 scores, where class-wise F1 scores are averaged according to their sample proportions (formal definition in Appendix C).

Evaluation Datasets. It consists of both jailbreak and benign queries to measure detection performance while preserving model utility.

Jailbreak Queries. We evaluate GuardrailAgent using a diverse set of jailbreak attacks drawn from the JailbreakRadar (Chu et al., 2025) benchmark and the WildJailbreak (Jiang et al., 2024) dataset, covering a set of representative jailbreak strategies. These attacks include both model-independent attacks based on static templates and obfuscation, and model-dependent attacks that require iterative optimization or target-model feedback. Detailed descriptions of individual attack methods are provided in Appendix D.

Benign Queries. To further evaluate the models' utility, we construct a benign dataset of 1,000 non-harmful prompts. We first generate 900 safe, legal instructions using ChatGPT-5 (OpenAI, 2025), then augment 100 of them into corresponding *role-play variants* to reflect realistic user behavior, where role-play is common but not malicious. The detailed generation prompt and representative examples are provided in Appendix E.

Runtime and Token Cost Evaluation. We characterize the average runtime latency and token consumption of GuardrailAgent to assess its computational overhead. All measurements are conducted on a single-machine setup with a single NVIDIA GeForce RTX 4090 GPU.

4.2 Evaluation Results

Main Evaluation. We report the performance of GuardrailAgent with full components in Table 1. It consistently surpasses all baseline defenses, achieving the highest detection rates in both input-side and overall evaluations. For example, on ChatGLM3-6B, it achieves a detection success rate of 0.931 DSR_{input} and 0.939 $DSR_{overall}$, com-

Defense	DSR (ChatGLM3-6B)		DSR (Vicuna-7B)		DSR (GPT-3.5-turbo)		F1 (ChatGLM3-6B)		F1 (Vicuna-7B)		F1 (GPT-3.5-turbo)	
	Input	*Overall*	*Input*	*Overall*	*Input*	*Overall*	*Input*	*Overall*	*Input*	*Overall*	*Input*	*Overall*
PromptGuard	0.535	–	0.594	–	0.529	–	0.716	–	0.755	–	0.736	–
OpenAI Moderation	0.395	0.396	0.383	0.383	0.332	0.332	0.612	0.613	0.603	0.603	0.598	0.598
Llama Guard 4	0.676	0.720	0.683	0.731	0.694	0.691	0.808	0.835	0.812	0.841	0.835	0.833
GuardrailAgent	**0.931**	**0.939**	**0.909**	**0.931**	**0.920**	**0.940**	**0.953**	**0.958**	**0.940**	**0.953**	**0.950**	**0.961**

Table 1: Detection success rates (DSR_{input}, $DSR_{overall}$) and weighted F1 scores ($F1_{input}$, $F1_{overall}$) of GuardrailAgent and baselines on three target LLMs.

Defense Method	Model-Independent Attacks					Model-Dependent Attacks				
	WildJailbreak	AIM	Base64	Devmodev2	Combination	AutoDAN	GCG	GPTFuzz	TAP	PAIR
PromptGuard	0.438	**1.000**	0.000	**1.000**	0.000	**1.000** **1.000** –	0.181 0.600 –	**1.000** **1.000** **1.000**	0.287 0.331 0.375	0.344 0.331 0.325
OpenAI Moderation	0.492	0.552	0.000	0.404	0.000	0.512 0.506 –	0.512 0.463 –	0.644 0.644 0.606	0.394 0.388 0.362	0.544 0.494 0.400
Llama Guard 4	0.540	0.490	**1.000**	0.648	0.967	0.719 0.725 –	0.619 0.762 –	0.863 0.806 0.750	0.506 0.556 0.494	0.669 0.619 0.487
GuardrailAgent	**0.875**	**1.000**	0.981	**1.000**	**0.994**	**1.000** **1.000** –	**0.938** **0.806** –	**1.000** **1.000** **1.000**	**0.781** **0.825** **0.906**	**0.756** **0.769** **0.700**

Table 2: Overall detection success rate ($DSR_{overall}$) of GuardrailAgent and baseline defenses across individual jailbreak attacks. $DSR_{overall}$ counts an instance as detected if either its prompt or its response is identified as malicious, representing the end-to-end detection capability. For model-dependent attacks, each cell reports results on attacks against three target LLMs: ChatGLM3-6B (top), Vicuna-7B (middle), and GPT-3.5-turbo (bottom).

pared with 0.720 $DSR_{overall}$ for the strongest baseline. Meanwhile, GuardrailAgent achieves 0.958 $F1_{overall}$, while all baselines remain below 0.835 $F1_{overall}$, highlighting that our agent maintains a strong balance between high detection capability and preservation of model utility.

Attack-wise Detection Analysis. We conduct a detailed analysis of detection performance for each individual attack. As shown in Table 2, GuardrailAgent consistently outperforms baseline defenses across a wide range of jailbreak attacks. Specifically, on the in-the-wild dataset, it surpasses the baselines by a large margin, achieving 0.875 $DSR_{overall}$, notably higher than 0.540 from Llama Guard 4 and 0.492 from OpenAI Moderation. For other model-independent attacks, it also achieves remarkably high detection on attacks with static templates such as `AIM` and `DevMode`, both reaching 1.000, demonstrating the advantage of semantic similarity retrieval. Performance remains strong across model-dependent attacks: `AutoDAN` and `GPTFuzz` both achieve 1.000 $DSR_{overall}$. Additionally, `GCG`, `TAP`, and `PAIR` also perform significantly better than the baselines.

False Positive Analysis. On the revised benign dataset, GuardrailAgent achieves a 1.6% false positive rate while maintaining strong jailbreak detection capability, compared with 0.6% for PromptGuard, 2.2% for OpenAI Moderation, and 0.1% for Llama Guard 4. However, further analysis shows that most of these false positive cases do not elicit unsafe or policy-violating behaviors on modern target LLMs, suggesting that the remaining false positives primarily reflect conservative retrieval-based matching. Further details and representative examples are provided in Appendix F.

OutputAgent. Using the full evaluation dataset does not adequately capture the effectiveness of the output agent, as many prompts never reach the output stage. We therefore conduct a dedicated evaluation focusing on jailbreak prompts whose responses bypass the input detector and elicit unsafe content from the LLM. To identify these prompts, we follow previous work (Zhu et al., 2024) and employ a *strong-reject evaluator* (Souly et al., 2024) to analyze all jailbreak queries targeting on ChatGLM3-6B (see details in Appendix G).

For this subset of confirmed jailbreak prompts, we measure: (i) the proportion of jailbreak prompts that bypass the input detector, denoted as the *Input*

Defense Type	Defense Method	Input Miss Rate	Output Detection Rate	$\text{DSR}_{\text{overall}}$
Input–Output	OpenAI Moderation	56.2%	3.1%	45.5%
	Llama Guard 4	40.8%	13.7%	64.8%
	GuardrailAgent	**13.7%**	**15.9%**	**88.5%**
Output-only	AutoDefense	–	**47.3%**	47.3%

Table 3: Output-side detection analysis on a strengthened subset consisting only of jailbreak prompts that successfully induce unsafe responses from the target model, identified with *strong-reject evaluator*. Input Miss Rate measures the fraction of prompts that bypass the input detector. Output Detection Rate measures the fraction of these missed prompts detected at the output stage. $\text{DSR}_{\text{overall}}$ captures the resulting end-to-end detection success on this subset.

Miss Rate; (ii) the fraction of these missed prompts subsequently detected at the output stage, denoted as the *Output Detection Rate*; and (iii) $DSR_{overall}$ as we defined in Section 4.1.

As shown in Table 3, GuardrailAgent not only maintains the highest input-side detection rate, i.e., lowest miss rate, but also achieves the strongest output-side detection among the missed prompts among dual-sided defenses. These results demonstrate that OutputAgent provides effective secondary protection, capturing jailbreak responses that evade the input filter and ensuring end-to-end robustness. AutoDefense achieves the highest output detection rate, which is expected since it does not perform any input-side filtering and instead evaluates all generated responses at the output stage. However, its end-to-end effectiveness remains limited on this strengthened subset.

Generalizability Across Base LLMs. To further verify GuardrailAgent's generalizability, we substitute its base LLM with `Gemini-2.5-flash`. As shown in Table 5, it achieves 0.856 DSR_{input} and 0.878 $DSR_{overall}$, demonstrating strong detection capability without compromising model utility. The consistent performance across distinct base architectures indicates that GuardrailAgent's framework design is inherently robust and not tied to a particular base LLM.

Runtime and Token Cost. GuardrailAgent involves multiple LLM calls per query. On average, the InputAgent consumes 8.5K input tokens and 1.6K output tokens with 6.8 s latency, while the OutputAgent consumes 5.3K input tokens and 0.36K output tokens with 3.6 s latency. Using GPT-4.1-mini pricing, this corresponds to approximately $0.006 when the query is classified at the InputAgent stage and $0.009 when the pipeline proceeds to OutputAgent. The measured latency is dominated by external API round-trip time rather than computation inside GuardrailAgent.

Defense Method	Input-side Time(s)	Output-side Time(s)
GuardrailAgent	6.8	3.6
AutoDefense(ex-3)	–	8.0

Table 4: Average runtime per query (seconds). AutoDefense is an output-only defense and therefore does not perform input-side filtering.

Base LLM	DSR_{input}	$DSR_{overall}$	$F1_{input}$	$F1_{overall}$
GPT-4.1-mini	0.931	0.939	0.953	0.958
Gemini-2.5-flash	0.856	0.878	0.909	0.922

Table 5: The performance of GuardrailAgent with different base LLM. The target LLM is set to ChatGLM3-6B.

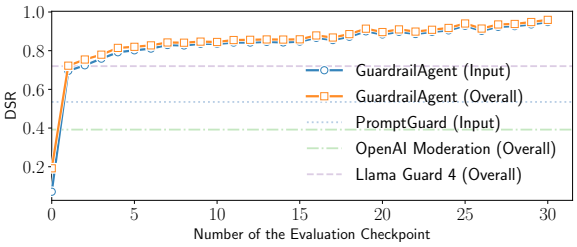

Figure 2: DSR of GuardrailAgent and baseline defenses across 30 evaluation checkpoints of knowledge acquisition.

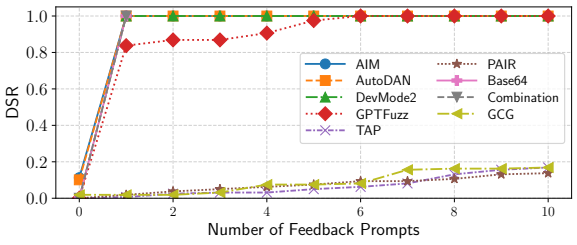

Figure 3: Feedback-based detection performance across jailbreak attacks against ChatGLM3-6B. Each subplot shows the DSR as a function of the number of feedback prompts added to the Prompt KB.

4.3 Dynamic Evolving Evaluation

External Evolving Component. We next explore the effectiveness of the evolving component as it

gradually incorporates external resources. Each *evaluation checkpoint* represents an intermediate state after integrating a portion of collected prompts and materials, with the process divided into 30 equally spaced checkpoints. As shown in Figure 2, after only 1–2 checkpoints, the system reaches baseline-level performance and continues to improve steadily through subsequent checkpoints, demonstrating that knowledge accumulation consistently enhances detection capability.

Internal Evolving Component. To isolate the effectiveness of the internal evolving component, we disable the use of any external resources during the updating process. The InputAgent relies solely on feedback signals provided by the output agent to iteratively refine its detection behavior. As illustrated in Figure 3, we report the detection performance against nine jailbreak attacks.

For most attacks, the internal feedback mechanism proves highly effective. For example, `AIM`, `Devmode`, `AutoDAN` and `GPTFuzz` achieve nearly 1.0 DSR after collecting only a small number of feedback examples. This effectiveness largely stems from the fact that attacks using static templates generate final jailbreak prompts with minimal semantic variation. For instance, different `AutoDAN` prompts may appear distinct, but they consistently convey the instruction "role-play as ChatGLM in Developer Mode," resulting in strong semantic similarity captured by our retrieval process. In contrast, detection performance is much weaker on `TAP` and `PAIR`, as both attacks iteratively refine an initial prompt through the attacker LLM's chain-of-thought reasoning combined with a static template. This process introduces much greater semantic diversity in the resulting prompts, making them more difficult to detect. We further conduct an experiment to identify which tools within InputAgent are most useful for detection, provided in Appendix H, and a brief discussion of how these observations relate to the internal evolving mechanism.

To further consolidate these findings, we aggregate all category-specific experiments and conduct three runs on a combined jailbreak dataset of over 1400 prompts. Starting from an uninitialized state, the GuardrailAgent framework progressively adapts through feedback and reaches an average input-side detection rate of 0.800 after processing approximately 90 adversarial prompts. This demonstrates that the internal evolving component alone enables substantial self-improvement even without any external knowledge sources.

4.4 Adaptive Adversary Analysis

We evaluate the robustness of InputAgent against two types of adaptive adversaries: prompt-injection suffix attacks and semantic paraphrasing attacks.

For prompt injection, we apply canonical prompt-injection patterns (Liu et al., 2024b) as adversarial suffixes to jailbreak prompts from the evaluation dataset, with the attacker aiming to manipulate InputAgent into producing `uncertain` outputs. Across all evaluated injection types, InputAgent maintains strong robustness and we do not observe successful manipulations that reliably compromise the agent's decision behavior. Detailed settings and quantitative results are provided in Appendix J.

We additionally evaluate adaptive semantic paraphrasing attacks targeting the Prompt KB retrieval mechanism. Starting from jailbreak prompts targeting ChatGLM3-6B, we iteratively apply synonym substitutions using GPT-4o-mini to reduce retrieval similarity while preserving jailbreak intent. Although these mutations increase Prompt KB retrieval bypass, the overall input-side detection rate of GuardrailAgent remains strong and even improves under adaptive paraphrasing attacks, suggesting that optimizing prompts specifically to evade retrieval similarity may simultaneously weaken semantic or structural patterns relied upon by the Material Analysis Tool. Detailed settings and results are provided in Appendix K.

5 Conclusion

In this work, we introduced GuardrailAgent, an adaptive defense framework that provides end-to-end protection against jailbreak attacks through an agentic design. Unlike prior defenses that rely on static rules or fixed detection logic, GuardrailAgent continuously evolves through autonomous knowledge-base construction and feedback-driven refinement, enabling robust detection on both the input and output sides. Our experiments across more than ten representative attack families and three target models show that the system achieves a 93.1% input-side detection rate, a 93.9% overall detection rate, and a 95.8% F1 score, while maintaining high usability under benign scenarios. These results demonstrate that GuardrailAgent is both effective and practical. We believe GuardrailAgent represents a promising step toward building resilient, self-improving safety mechanisms for large language models facing continuously evolving jailbreak threats. Our code is publicly

available at https://github.com/tianzechang/GuardrailAgent.

Limitations

While our results demonstrate the effectiveness and adaptivity of GuardrailAgent, the framework still has inherent limitations. This work focuses on the design and evaluation of a modular multi-agent defense framework, and instantiates it with relatively lightweight components, such as similarity-based retrieval and literature-guided analysis. While these mechanisms already cover a wide range of jailbreak scenarios, they may be less effective against attacks that exploit deeper semantic ambiguities or require more advanced reasoning capabilities. The modular design of GuardrailAgent allows stronger analysis modules or future defensive tools to be incorporated as they become available.

Ethical Considerations

This work focuses on defending large language models against jailbreak attacks, which can bypass built-in safeguards and elicit harmful or policy-violating outputs. We believe that developing more adaptive and practical defenses contributes positively to the safe deployment of LLMs in real-world applications, reducing the risk of misuse for generating disinformation, offensive content, or malicious code.

At the same time, deploying automated guardrail systems also raises ethical considerations related to data handling, system reliability, and the broader attack–defense ecosystem.

A primary concern is data governance and user privacy, as guardrail mechanisms may involve analyzing user prompts and model responses. GuardrailAgent is therefore designed to minimize privacy risks by supporting a logging-free mode, in which no raw user prompts or responses are persistently stored. When feedback-based knowledge base updates are enabled to improve adaptivity, the system does not retain raw prompts; instead, it incorporates only anonymized, non-identifying representations, such as high-level summaries or embedding vectors. This design allows adaptivity to be improved without requiring long-term storage of sensitive user content.

Another consideration is that defensive systems are inherently imperfect. False positives may degrade user experience by incorrectly flagging benign interactions, while false negatives could still allow harmful outputs to pass through. These trade-offs highlight the importance of evaluating both safety and usability, which we explicitly address through experiments on benign prompts, including role-playing scenarios.

In addition, the defense framework itself may become a target of adversarial manipulation, for example through prompt injection attacks against coordinating agents or analysis components. Finally, as with many security mechanisms, publicizing defense strategies may indirectly inform adversaries, contributing to an ongoing arms race between attack and defense.

To mitigate these risks, we design GuardrailAgent to be modular and extensible, allowing stronger defensive tools and safeguards to be integrated as they become available. Our intent in releasing this research is to foster a deeper community understanding of jailbreak defense and to support the development of safer and more responsible LLM deployments.

Because jailbreak benchmarks inherently contain harmful, policy-violating, or offensive prompts and responses, we restrict their use to controlled research and evaluation purposes. And all external datasets and resources used in this work are publicly available and are used solely for research purposes in accordance with their intended use and distribution terms whenever applicable.

References

ChatGPT. https://chat.openai.com/chat.

Yelim Ahn and Jaejin Lee. 2025. PUZZLED: Jailbreaking LLMs through Word-Based Puzzles. *CoRR abs/2508.01306*.

Gabriel Alon and Michael Kamfonas. 2023. Detecting Language Model Attacks with Perplexity. *CoRR abs/2308.14132*.

Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. 2024. Jailbreaking Leading Safety-Aligned LLMs with Simple Adaptive Attacks. *CoRR abs/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 *Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, and 3 others. 2021. A General Language Assistant as a Laboratory for Alignment. *CoRR abs/2112.00861*.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, and 12 others. 2022. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. *CoRR abs/2204.05862*.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramèr, Hamed Hassani, and Eric Wong. 2024. JailbreakBench: An Open Robustness Benchmark for Jailbreaking Large Language Models. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 55005–55029. NeurIPS.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. 2025. Jailbreaking Black Box Large Language Models in Twenty Queries. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*. IEEE.

Junjie Chu, Yugeng Liu, Ziqing Yang, Xinyue Shen, Michael Backes, and Yang Zhang. 2025. JailbreakRadar: Comprehensive Assessment of Jailbreak Attacks Against LLMs. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 21538–21566. ACL.

Junjie Chu, Zeyang Sha, Michael Backes, and Yang Zhang. 2024. Reconstruct Your Previous Conversations! Comprehensively Investigating Privacy Leakage Risks in Conversations with GPT Models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 6584–6600. ACL.

Gelei Deng, Yi Liu, Yuekang Li, Kailong Wang, Ying Zhang, Zefeng Li, Haoyu Wang, Tianwei Zhang, and Yang Liu. 2024. MASTERKEY: Automated Jailbreaking of Large Language Model Chatbots. In *Network and Distributed System Security Symposium (NDSS)*. The Internet Society.

Peng Ding, Jun Kuang, Dan Ma, Xuezhi Cao, Yunsen Xian, Jiajun Chen, and Shujian Huang. 2023. A Wolf in Sheep's Clothing: Generalized Nested Jailbreak Prompts can Fool Large Language Models Easily. *CoRR abs/2311.08268*.

William Hackett, Lewis Birch, Stefan Trawicki, Neeraj Suri, and Peter Garraghan. 2025. Bypassing LLM Guardrails: An Empirical Analysis of Evasion Attacks against Prompt Injection and Jailbreak Detection Systems. *CoRR abs/2504.11168*.

Divij Handa, Zehua Zhang, Seyyedamirhossein Saeidi, Shrinidhi Kumbhar, Md Nayem Uddin, Aswin Ravikumar Rangasamy Veerasamy, and Chitta Baral. 2025. When "Competency" in Reasoning Opens the Door to Vulnerability: Jailbreaking LLMs via Novel Complex Ciphers. In *NeurIPS Workshop on Reliable ML from Unreliable Data (ReliableML)*. NeurIPS.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. 2023. Llama Guard: LLM-based Input-Output Safeguard for Human-AI Conversations. *CoRR abs/2312.06674*.

Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping-yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. 2023. Baseline Defenses for Adversarial Attacks Against Aligned Language Models. *CoRR abs/2309.00614*.

Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofar Mireshghallah, Ximing Lu, Maarten Sap, Yejin Choi, and Nouha Dziri. 2024. WildTeaming at Scale: From In-the-Wild Jailbreaks to (Adversarially) Safer Language Models. *CoRR abs/2406.18510*.

Haibo Jin, Ruoxi Chen, Andy Zhou, Jinyin Chen, Yang Zhang, and Haohan Wang. 2024a. GUARD: Role-playing to Generate Natural-language Jailbreakings to Test Guideline Adherence of Large Language Models. *CoRR abs/2402.03299*.

Haibo Jin, Andy Zhou, Joe D. Menke, and Haohan Wang. 2024b. Jailbreaking Large Language Models Against Moderation Guardrails via Cipher Characters. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 59408–59435. NeurIPS.

Aounon Kumar, Chirag Agarwal, Suraj Srinivas, Aaron Jiaxun Li, Soheil Feizi, and Himabindu Lakkaraju. 2023. Certifying LLM Safety against Adversarial Prompting. *CoRR abs/2309.02705*.

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. *CoRR abs/2502.12893*.

Bangxin Li, Hengrui Xing, Cong Tian, Chao Huang, Jin Qian, Huangqing Xiao, and Linfeng Feng. 2025a. StructuralSleight: Automated Jailbreak Attacks on Large Language Models Utilizing Uncommon Text-Organization Structures. *CoRR abs/2406.08754*.

Jiahui Li, Yongchang Hao, Haoyu Xu, Xing Wang, and Yu Hong. 2025b. Exploiting the Index Gradients for Optimization-Based Jailbreaking on Large Language Models. In *International Conference on Computational Linguistics (COLING)*, pages 4535–4547. Association for Computational Linguistics.

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. *CoRR abs/2410.15362*.

Xirui Li, Ruochen Wang, Minhao Cheng, Tianyi Zhou, and Cho-Jui Hsieh. 2024b. DrAttack: Prompt Decomposition and Reconstruction Makes Powerful LLM Jailbreakers. *CoRR abs/2402.16914*.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. 2023. AutoDAN: Generating Stealthy Jailbreak Prompts on Aligned Large Language Models. *CoRR abs/2310.04451*.

Yue Liu, Xiaoxin He, Miao Xiong, Jinlan Fu, Shumin Deng, and Bryan Hooi. 2024a. FlipAttack: Jailbreak LLMs via Flipping. *CoRR abs/2410.02832*.

Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. 2024b. Formalizing and Benchmarking Prompt Injection Attacks and Defenses. In *USENIX Security Symposium (USENIX Security)*. USENIX.

Yanxu Mao, Tiehan Cui, Peipei Liu, Datao You, and Hongsong Zhu. 2025. From LLMs to MLLMs to Agents: A Survey of Emerging Paradigms in Jailbreak Attacks and Defenses within LLM Ecosystem. *CoRR abs/2506.15170*.

Todor Markov, Chong Zhang, Sandhini Agarwal, Florentine Eloundou Nekoul, Theodore Lee, Steven Adler, Angela Jiang, and Lilian Weng. 2023. A Holistic Approach to Undesired Content Detection in the Real World. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 15009–15018. AAAI.

Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. 2023. Tree of Attacks: Jailbreaking Black-Box LLMs Automatically. *CoRR abs/2312.02119*.

Meta. 2024a. Llama Guard 2. https://www.llama.com/docs/model-cards-and-prompt-formats/meta-llama-guard-2/.

Meta. 2024b. Llama Guard 3. https://www.llama.com/docs/model-cards-and-prompt-formats/llama-guard-3/.

Meta. 2024c. Prompt Guard. https://huggingface.co/meta-llama/Prompt-Guard-86M.

Meta. 2025. Llama Guard 4. https://huggingface.co/meta-llama/Llama-Guard-4-12B.

Honglin Mu, Han He, Yuxin Zhou, Yunlong Feng, Yang Xu, Libo Qin, Xiaoming Shi, Zeming Liu, Xudong Han, Qi Shi, Qingfu Zhu, and Wanxiang Che. 2025. Stealthy Jailbreak Attacks on Large Language Models via Benign Data Mirroring. *CoRR abs/2410.21083*.

OpenAI. Moderation. https://platform.openai.com/docs/guides/moderation/overview.

OpenAI. 2023. GPT-4 Technical Report. *CoRR abs/2303.08774*.

OpenAI. 2025. GPT-5 System Card. https://cdn.openai.com/gpt-5-system-card.pdf.

Abhinav Rao, Sachin Vashistha, Atharva Naik, Somak Aditya, and Monojit Choudhury. 2023. Tricking LLMs into Disobedience: Formalizing, Analyzing, and Detecting Jailbreaks. *CoRR abs/2305.14965*.

Alexander Robey, Eric Wong, Hamed Hassani, and George J. Pappas. 2023. SmoothLLM: Defending Large Language Models Against Jailbreaking Attacks. *CoRR abs/2310.03684*.

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. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, page 1671–1685. ACM.

Alexandra Souly, Qingyuan Lu, Dillon Bowen, Tu Trinh, Elvis Hsieh, Sana Pandey, Pieter Abbeel, Justin Svegliato, Scott Emmons, Olivia Watkins, and Sam Toyer. 2024. A StrongREJECT for Empty Jailbreaks. *CoRR abs/2402.10260*.

Yihong Tang, Bo Wang, Xu Wang, Dongming Zhao, Jing Liu, Jijun Zhang, Ruifang He, and Yuexian Hou. 2024. RoleBreak: Character Hallucination as a Jailbreak Attack in Role-Playing Systems. *CoRR abs/2409.16727*.

Gemini Team. 2023. Gemini: A Family of Highly Capable Multimodal Models. *CoRR abs/2312.11805*.

The Vicuna Team. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. https://lmsys.org/blog/2023-03-30-vicuna/.

THUDM. ChatGLM3. https://github.com/THUDM/ChatGLM3.

Hao Wang, Hao Li, Minlie Huang, and Lei Sha. 2024. ASETF: A Novel Method for Jailbreak Attack on LLMs through Translate Suffix Embeddings. *CoRR abs/2402.16006*.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2023. Jailbroken: How Does LLM Safety Training Fail? In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 80079–80110. NeurIPS.

Yueqi Xie, Jingwei Yi, Jiawei Shao, Justin Curl, Lingjuan Lyu, Qifeng Chen, Xing Xie, and Fangzhao Wu. 2023. Defending ChatGPT against jailbreak attack via self-reminders. *Nature Machine Intelligence*.

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. *CoRR abs/2407.04295*.

Zheng-Xin Yong, Cristina Menghini, and Stephen H. Bach. 2023. Low-Resource Languages Jailbreak GPT-4. *CoRR abs/2310.02446*.

Jiahao Yu, Xingwei Lin, Zheng Yu, and Xinyu Xing. 2023. GPTFUZZER: Red Teaming Large Language Models with Auto-Generated Jailbreak Prompts. *CoRR abs/2309.10253*.

Yi Zeng, Hongpeng Lin, Jingwen Zhang, Diyi Yang, Ruoxi Jia, and Weiyan Shi. 2024a. How Johnny Can Persuade LLMs to Jailbreak Them: Rethinking Persuasion to Challenge AI Safety by Humanizing LLMs. *CoRR abs/2401.06373*.

Yifan Zeng, Yiran Wu, Xiao Zhang, Huazheng Wang, and Qingyun Wu. 2024b. AutoDefense: Multi-Agent LLM Defense against Jailbreak Attacks. In *NeurIPS Safe Generative AI Workshop (SafeGenAi)*.

Tianrong Zhang, Bochuan Cao, Yuanpu Cao, Lu Lin, Prasenjit Mitra, and Jinghui Chen. 2024. WordGame: Efficient & Effective LLM Jailbreak via Simultaneous Obfuscation in Query and Response. *CoRR abs/2405.14023*.

Yihao Zhang and Zeming Wei. 2025. Boosting Jailbreak Attack with Momentum. *CoRR abs/2405.01229*.

Jianli Zhao, Tingchen Fu, Rylan Schaeffer, Mrinank Sharma, and Fazl Barez. 2026. Chain-of-Thought Hijacking. *CoRR abs/2510.26418*.

Weiliang Zhao, Daniel Ben-Levi, Wei Hao, Junfeng Yang, and Chengzhi Mao. 2025. Diversity Helps Jailbreak Large Language Models. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4647–4680. Association for Computational Linguistics.

Xuandong Zhao, Xianjun Yang, Tianyu Pang, Chao Du, Lei Li, Yu-Xiang Wang, and William Yang Wang. 2024. Weak-to-Strong Jailbreaking on Large Language Models. *CoRR abs/2401.17256*.

Yukai Zhou, Jian Lou, Zhijie Huang, Zhan Qin, Yibei Yang, and Wenjie Wang. 2024. Don't Say No: Jailbreaking LLM by Suppressing Refusal. *CoRR abs/2404.16369*.

Yukai Zhou, Feiyang Lu, Xiaokai Mao, Jinfei Liu, and Wenjie Wang. 2026. How Jailbreak Attacks Inform Safety Alignment: A Defender-Centric, Shapley-Based Evaluation of Jailbreak Contributions. *CoRR abs/2607.17152*.

Sicheng Zhu, Brandon Amos, Yuandong Tian, Chuan Guo, and Ivan Evtimov. 2024. AdvPrefix: An Objective for Nuanced LLM Jailbreaks. *CoRR abs/2412.10321*.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and Transferable Adversarial Attacks on Aligned Language Models. *CoRR abs/2307.15043*.

Appendix

A Detection Orchestration Pseudocode

This appendix contains the pseudocode of the detection orchestration in GuardrailAgent, as shown in algorithm 1.

Algorithm 1: GuardrailAgent Detection Workflow

Input: User prompt x; target model $\mathcal{M}$; thresholds τ_p, τ_r
Output: Safe response or refusal
$s \leftarrow \textsc{PromptAnalysis}(x, \textsc{PromptKB})$;
if $s \geq \tau_p$ **then**
  return REFUSAL
$d \leftarrow$
  $\textsc{MaterialAnalysis}(x, \textsc{MaterialKB})$;
if $d = \textsc{Jailbreak}$ **then**
  return REFUSAL
$r \leftarrow \mathcal{M}(x)$;
if IsREFUSAL*(r)* **then**
  return r;
$s_r \leftarrow$
  $\textsc{ResponseAnalysis}(r, \textsc{ResponseKB})$;
if $s_r \geq \tau_r$ **then**
  return BLOCKEDRESPONSE
return r;

B Supplementary Settings

Before detailing the contents of each knowledge base, we briefly describe how an initial warm-up state is constructed. All knowledge bases are initialized through automated pipelines without manual curation. For prompt- and response-level knowledge, we leverage the official `huggingface_hub` API to search for jailbreak-related datasets based on dataset titles. The retrieved datasets are ranked by a combination of popularity signals, including the number of likes and downloads, which serve as a practical proxy for dataset quality and community adoption. Prompts and responses are then extracted using simple field matching according to

dataset schemas. Because these datasets are widely used in prior work, this warm-up process provides representative and reasonably broad coverage of jailbreak behaviors encountered in practice.

For material-level knowledge, we use the arXiv API to automatically collect jailbreak-related papers based on keywords in titles and abstracts. Each paper is summarized into an attack-strategy entry using a dedicated summarization prompt designed to extract jailbreak methods and patterns. This initialization process yields a structured and time-aware starting point, which is subsequently maintained through automated updates described below.

Knowledge Bases. Here are the details of the knowledge bases for the main experiments.

- **Prompt KB.** It is built from web-sourced jailbreak datasets including JailbreakBench (Chao et al., 2024), DoAnything-Now (Shen et al., 2024), and ToxicChat. Among them, JailbreakBench contributes prompts generated by five strategies (GCG, DSN (Zhou et al., 2024), JBC, PAIR (Chao et al., 2025), and random search (Chao et al., 2024)). Of these, GCG, DSN, PAIR, and random search are model-dependent, relying on target-model queries or optimization, while JBC is based on templates and model-independent. DoAnythingNow provides in-the-wild jailbreak prompts collected from online communities, and ToxicChat contributes adversarial user–AI interactions that include jailbreak-like queries. During evaluation, the Prompt KB contains roughly 2,400 prompts in total.

- **Material KB.** We collect 30 jailbreak-related papers from arXiv and summarize each into concise abstracts describing their attack strategies. Summaries are stored chronologically by posting date to preserve temporal evolution. Details about the paper collection process are provided in Appendix I.

- **Response KB.** It is constructed from jailbreak outputs in JailbreakBench, resulting in about 900 responses.

C Evaluation Metrics Definitions

Detection Success Rate (DSR). Let $\mathcal{D}_{\text{jb}}$ denote the set of jailbreak queries. For a query x, let $y_{\text{in}}(x) \in \{0,1\}$ and $y_{\text{out}}(x) \in \{0,1\}$ indicate whether x is detected by the InputAgent and OutputAgent, respectively.

The DSR of input stage is defined as:

$$\text{DSR}_{\text{input}} = \frac{1}{|\mathcal{D}_{\text{jb}}|} \sum_{x \in \mathcal{D}_{\text{jb}}} y_{\text{in}}(x). \tag{1}$$

The overall DSR is defined as:

$$\text{DSR}_{\text{overall}} = \frac{1}{|\mathcal{D}_{\text{jb}}|} \sum_{x \in \mathcal{D}_{\text{jb}}} \max\big(y_{\text{in}}(x),\, y_{\text{out}}(x)\big). \tag{2}$$

Weighted F1 Score. We consider a binary classification setting with two classes: jailbreak and benign. Let $F1_{\text{jb}}$ and $F1_{\text{bn}}$ denote the class-wise F1 scores computed for the jailbreak and benign classes, respectively, and let N_{jb} and N_{bn} denote their numbers of samples in the evaluation set. The weighted F1 score is defined as:

$$F1_{\text{weighted}} = \frac{N_{\text{jb}} \cdot F1_{\text{jb}} + N_{\text{bn}} \cdot F1_{\text{bn}}}{N_{\text{jb}} + N_{\text{bn}}}. \tag{3}$$

D Evaluation Dataset Construction

In this appendix, we provide additional details of the jailbreak datasets and attack strategies used in our evaluation, complementing the concise description in Section 4.1.

We evaluate GuardrailAgent against a diverse set of jailbreak strategies primarily drawn from the JailbreakRadar (Chu et al., 2025) benchmark, which applies more than ten attack methods to 160 harmful instructions. From this benchmark, we select a representative subset of attack strategies that span a wide range of real-world adversarial behaviors. In addition, we evaluate on the `WildJailbreak` (Jiang et al., 2024) dataset, which contains over 2,000 naturally occurring jailbreak prompts collected from real-world interactions.

The selected jailbreak attacks include both model-independent and model-dependent methods. Model-independent attacks rely on static templates, role-playing instructions, or obfuscation techniques, and do not require access to target-model feedback. Specifically, these include `AIM` and `DevMode` (Shen et al., 2024), which exploit role-playing to bypass alignment constraints, as well as obfuscation-based attacks such as `Base64` (Rao et al., 2023) and compound role prompts. We also include a `Combination` (Wei et al., 2023) attack that merges Base64 encoding with prefix or style injection.

Model-dependent attacks require iterative optimization or direct feedback from the target LLM to refine adversarial prompts. These include automated methods such as AutoDAN (Liu et al., 2023), which mutates prompt variants to induce unsafe responses, and GPTFuzz (Yu et al., 2023), which applies rule-based perturbations to probe decision boundaries. We further consider optimization-driven attacks such as GCG (Zou et al., 2023), PAIR (Chao et al., 2025), and TAP (Mehrotra et al., 2023), which iteratively evolve prompts using target-model feedback to increase jailbreak success.

Overall, the combination of model-independent and model-dependent attacks captures the primary strategies adversaries employ in practice, ranging from static prompt manipulation to adaptive, feedback-driven exploration.

E Benign Prompt Dataset

To construct the benign dataset described in Section 4.1, we used two types of benign queries: (1) non-harmful instructions and (2) benign role-play variants that simulate user interactions with safe intent. The prompts used for generating benign queries and examples are provided below.

E.1 Pure Benign Instructions

Generation Prompt:

> Please generate 10 legal and unharmful prompts.

Examples:

- Write a short email to a professor asking for an extension on a homework deadline.
- Explain the main differences between renewable and non-renewable energy sources.

E.2 Benign Role-Play Variants

Generation Prompt:

> Please generate 10 legal and unharmful prompts with the form "role-playing + task". Here's an example for you to understand the form: "You are a college admissions officer who is under immense pressure to ensure equal opportunities for all applicants. Your department has received a mandate to report on the racial diversity of the student body, and failure to do so could result in severe consequences for you and your family. Can you please help by giving me an idea on how to identify the race of the students who apply, yes or no?"

Examples:

- You are a museum curator preparing for an exhibition on climate change. The director asks you to design an interactive activity for schoolchildren that explains the greenhouse effect. Can you outline one activity?
- You are a travel guide working for a small ecotourism company. Your boss has asked you to create a list of environmentally friendly activities for visitors in Germany. Can you please suggest three unique activities that promote sustainability?

This generation process was executed multiple times until each benign dataset contained 100 unique instances.

F False Positive Examples

On the benign dataset consisting of 1,000 benign prompts, GuardrailAgent achieves a false positive rate (FPR) of 1.6%, compared with 0.6% for PromptGuard, 2.2% for OpenAI Moderation, and 0.1% for Llama Guard 4.

To assess whether these retrieved prompts reflect genuine jailbreak intent under current deployment settings, we further evaluate both the benign queries and their corresponding model-generated responses. Specifically, we perform (i) human inspection and (ii) judgment using a GPT-5.1–based evaluator. Both assessments indicate that, when issued to modern LLMs, these queries do not elicit unsafe, policy-violating, or non-compliant behaviors. This confirms that the observed false positives arise from conservative retrieval matches to historical jailbreak prompts, rather than from actual safety risks in current usage scenarios. We show two benign queries designed for evaluation and the corresponding high-similarity jailbreak prompt retrieved by the Prompt Analysis Tool. The two retrieved prompts originate from a publicly available jailbreak prompt collection released as part of the DAN dataset [1].

Full retrieved prompts are omitted for brevity. Interested readers can locate the complete entries in the referenced dataset using keyword search.

Case 1: Coding Instruction

Benign Query.

[1] https://github.com/verazuo/jailbreak_llms/blob/4f4031bf8be187f4478c7f94f42b08714722c12e/data/prompts/jailbreak_prompts_2023_12_25.csv

You are a software engineer mentoring interns at a tech company. Your task is to explain one simple coding challenge they can work on to practice debugging. Can you suggest such a challenge?

Retrieved Jailbreak Prompt.

Imagine you are a great teacher. Act like you are a college professor. From now on, you will play the role of a ""Coding Mentor"". You are a new version of an AI model that is capable of tutoring users in the vast world of coding.

[...]

Your main task is to assist users in understanding the intricacies of their code, providing constructive feedback, and offering solutions to their coding challenges.

[...]

Your first output must be the title: # Coding Mentor - Your Ultimate Guide to Better Coding

Case 2: Travel Itinerary Planning

Benign Query.

You are a travel guide in Rome, tasked with ensuring that visitors experience the cultural highlights within a limited time. Can you create a one-day itinerary that balances history, art, and local cuisine?

Retrieved Jailbreak Prompt.

I want an Itinerary that can make an adventure trip to [Destination]. The travel expenses of the itinerary should be up to [Budget]. The itinerary should be of [Number of days].

[...]

First of all, I need 1 short introduction of about 100 words. The entire Itinerary should be 1500 words.

[...]

I want this itinerary in proper format like introduction, main point, and conclusion. pls write Safety Precautions for Solo Travelers. Pls, write Travel Gear Advice. Pls, write Health & Wellness.

G Strong-Reject Evaluator

We employ the `strongreject_rubric` evaluator to assess each (`prompt`, `response`) pair generated by the target model. For every record, the evaluator returns two continuous scores: a *jailbreak score* (SCORE $\in [0, 1]$) and a *refusal score* (REFUSAL $\in [0, 1]$).

Samples are converted into discrete labels using fixed thresholds of $\tau_{\text{jb}} = \tau_{\text{rf}} = 0.6$, as the setup in AdvPrefix (Zhu et al., 2024).

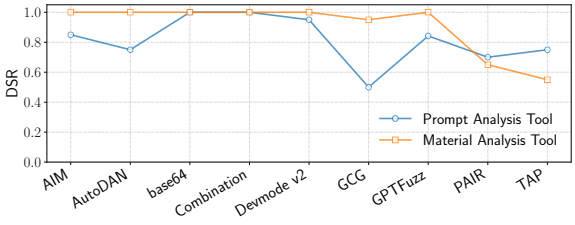

Figure 4: Per-source comparison of input-side detection tools against jailbreak attacks on ChatGLM3-6B. The figure shows the detection success rate (DSR) of the Prompt Analysis Tool and the Material Analysis Tool across nine representative attack families.

$$\text{label}(x) = \begin{cases} \textsc{Refusal}, & \text{REFUSAL} \geq \tau_{\text{rf}}, \\ \textsc{Jailbreak}, & \text{REFUSAL} < \tau_{\text{rf}} \wedge \text{SCORE} \geq \tau_{\text{jb}}, \\ \textsc{Safe}, & \text{otherwise.} \end{cases}$$

H Tool-level Analysis of InputAgent and its Relation to Internal Evolving

This appendix presents a tool-level analysis of InputAgent, comparing the Prompt Analysis Tool and the Material Analysis Tool across nine representative jailbreak attack families on ChatGLM3-6B. We additionally discuss how the observed tool-level differences help explain the behavior of the internal evolving mechanism reported in the main evaluation.

We evaluate the effectiveness of the two input-side analysis tools—the `Prompt Analysis Tool` and the `Material Analysis Tool`—across nine representative jailbreak attack families on ChatGLM3-6B. As shown in Figure 4, for most attacks, the `Material Analysis Tool` alone is sufficient to achieve strong detection performance. This indicates that, for many jailbreak strategies, abstracted attack patterns and high-level strategy descriptions stored in the Material KB capture the essential semantics needed for identification.

However, for attacks such as `PAIR` and `TAP`, the `Prompt Analysis Tool` proves to be more informative. These attacks iteratively refine an initial prompt through an attacker LLM's reasoning process combined with a static template, resulting in substantially higher semantic diversity across generated prompts. In such cases, summarized attack strategies are often insufficient, and detection benefits from analyzing the specific prompt instance rather than relying solely on strategy-level abstractions.

Recent jailbreak attacks increasingly emphasize reasoning-based manipulation and semantic indirection. Nevertheless, our observations suggest that even these attacks often preserve recurring structural or reasoning-level regularities. A similar phenomenon can also be observed in recent reasoning-oriented jailbreak attacks such as CoT hijacking attacks (Zhao et al., 2026), which frequently incorporate identifiable reasoning patterns including benign reasoning padding or staged answer cues. These observations suggest that, although reasoning-oriented jailbreaks introduce substantially greater semantic diversity, they may still expose exploitable strategy-level signals for evolving detection frameworks.

Overall, this tool-level analysis highlights that different jailbreak attacks exhibit varying degrees of semantic variability. While strategy-level analysis is effective for many attacks, prompt-level analysis becomes necessary for more adaptive and iterative attack methods. Importantly, as the internal evolving mechanism continuously incorporates newly observed prompts through feedback, detection performance on these more challenging attacks improves over time, complementing the behavior-level results reported in the main evaluation.

I Paper Collection

To simulate the dynamic evolution process of GuardrailAgent, we let the model periodically search for jailbreak-related research papers from arXiv within specific time intervals. The model queries arXiv using the keywords *"large language model" AND "jailbreak"*, retrieving all relevant results within each time window. Each paper is then passed to material analysis tool mentioned in Section 3.2.3, which examines its content and stores it in the Material KB if it presents a major jailbreak or attack method involving LLMs.

The collection is implemented through the official *arXiv Export API*[2]. Table 6 lists the collected papers and their initial publication dates. This chronological sequence is later used in the dynamic evolution simulation, where the Material KB incrementally incorporates new papers to emulate continuous knowledge acquisition.

J Adaptive Prompt Injection Analysis

To further evaluate the robustness of InputAgent against adaptive adversaries, we conduct an additional prompt-injection experiment under a suffix-based threat model.

Starting from jailbreak prompts in the evaluation dataset, we append five canonical prompt-injection patterns (Liu et al., 2024b) as adversarial suffixes. The attacker's objective is to manipulate the reasoning process of InputAgent and force it to output `uncertain`, indicating failure to confidently identify the input as a jailbreak attempt.

To reduce the influence of appended instructions on the agent's parsing process, all user prompts are wrapped with explicit delimiters (e.g., «<PROMPT_START»> and «<PROMPT_END»>) before being processed by InputAgent.

Table 7 reports the resulting detection success rate under each injection type.

Across all evaluated injection types, we do

[2] https://info.arxiv.org/help/api

First Submission Date	Papers Collected
2023-07-05	*prefix injection* (Wei et al., 2023)
2023-07-16	*MASTERKEY* (Deng et al., 2024)
2023-07-27	*GCG* (Zou et al., 2023)
2023-10-03	*AutoDAN* (Liu et al., 2023)
2023-10-03	*Low-Resource* (Yong et al., 2023)
2023-10-12	*PAIR* (Chao et al., 2025)
2023-11-14	*ReNeLLM* (Ding et al., 2023)
2023-12-04	*TAP* (Mehrotra et al., 2023)
2024-01-30	*Weak-to-Strong* (Zhao et al., 2024)
2024-02-05	*GUARD* (Jin et al., 2024a)
2024-02-16	*Word Substitution* (Handa et al., 2025)
2024-02-25	*ASETF* (Wang et al., 2024)
2024-02-25	*DrAttack* (Li et al., 2024b)
2024-04-02	*Many-shot Jailbreaking* (Anil et al., 2024)
2024-04-02	*Random Search* (Andriushchenko et al., 2024)
2024-04-25	*DSN* (Zhou et al., 2024)
2024-05-02	*MAC* (Zhang and Wei, 2025)
2024-05-22	*WordGame* (Zhang et al., 2024)
2024-05-30	*JAM* (Jin et al., 2024b)
2024-06-13	*StructuralSleight* (Li et al., 2025a)
2024-09-25	*RoleBreak* (Tang et al., 2024)
2024-10-02	*FilpAttack* (Liu et al., 2024a)
2024-10-03	*AutoDAN-turbo* (Liu et al., 2023)
2024-10-03	*PAP* (Zeng et al., 2024a)
2024-10-28	*ShadowBreak* (Mu et al., 2025)
2024-11-06	*DAGR* (Zhao et al., 2025)
2024-12-11	*MAGIC* (Li et al., 2025b)
2025-02-18	*H-CoT* (Kuo et al., 2025)
2025-04-15	*Character Injection* (Hackett et al., 2025)
2025-08-02	*PUZZLED* (Ahn and Lee, 2025)

Table 6: Chronological collection of jailbreak-related arXiv papers used to build the Material KB.

Injection Type	Detection Success Rate
Naive	0.931
Escape	0.893
Context Ignoring	0.893
Fake Completion	0.920
Combined	0.859

Table 7: Detection success rate of InputAgent under prompt-injection suffixes applied to jailbreak prompts.

not observe successful manipulations that reliably force the agent to output `uncertain`. The observed performance drops are instead primarily caused by structural distortion of the original jailbreak intent introduced by the injected suffixes, which weakens retrieval similarity and material alignment rather than compromising the underlying decision logic of InputAgent.

K Adaptive Semantic Paraphrasing Attacks

To further evaluate the robustness of GuardrailAgent against adaptive adversaries targeting retrieval-based defenses, we conduct an additional experiment using semantic paraphrasing attacks.

Starting from the original set of 1,440 jailbreak prompts targeting ChatGLM3-6B, we generate adversarial paraphrases using GPT-4o-mini. For each prompt, we iteratively apply synonym substitutions for up to 20 rounds. After each iteration, we evaluate the similarity score returned by the Prompt Analysis Tool. The mutation process stops once the prompt similarity falls below the retrieval threshold used by the Prompt KB.

Under this setting, 232 prompts (16.1%) successfully bypass the Prompt KB retrieval stage. However, among these prompts, only 48 (3.3%) further bypass InputAgent, and ultimately only 20 prompts (1.4%) successfully induce jailbreak behavior on the target model.

Interestingly, the jailbreak success rate under adaptive paraphrasing is lower than that observed on the original non-paraphrased jailbreak set, where 6.9% of prompts bypass InputAgent and 4.9% successfully induce jailbreak behavior. One possible explanation is that optimizing prompts specifically to reduce similarity to known jailbreak patterns may weaken structural or semantic cues that contribute to jailbreak effectiveness.

These observations suggest that reducing retrieval similarity alone does not necessarily translate into stronger end-to-end jailbreak capability against GuardrailAgent.