Agent Safety Alignment via Reinforcement Learning

Anonymous ACL submission

Abstract

The emergence of autonomous Large Language Model (LLM) agents capable of tool usage has introduced new safety risks that go beyond traditional conversational misuse. These agents, empowered to execute external functions, are vulnerable to both user-initiated threats (e.g., adversarial prompts) and tool-initiated threats (e.g., malicious outputs from compromised tools). In this paper, we propose the first unified safety-alignment framework for tool-using agents, enabling models to handle both channels of threat via structured reasoning and sandboxed reinforcement learning. We introduce a three-way taxonomy that classifies both user prompts and tool responses as benign, malicious, or sensitive, and we define a policy-driven decision model to train the agents. Our framework employs a custom-designed sandbox environment that simulates real-world tool execution and allows fine-grained reward shaping. Through extensive evaluations on public and self-built benchmarks, including Agent SafetyBench, InjecAgent, and BFCL, we demonstrate that our safety-aligned agents significantly improve resistance to security threats while preserving strong utility on benign tasks. Our results show that safety and effectiveness can be jointly optimized, laying the groundwork for trustworthy deployment of autonomous LLM agents.

1 Introduction

Recent advances have elevated Large Language Models (LLMs) from passive conversational interfaces to sophisticated autonomous agents capable of decomposing complex user objectives, orchestrating external tools, and executing multi-step workflows with minimal human oversight (Yao et al., 2023). Open-source initiatives such as AutoGPT, BabyAGI, and AgentGPT now demonstrate the capability to perform web searches, generate code, manipulate files, and deliver comprehensive business strategies through autonomous task management and execution (Significant Gravitas, 2023; Yohei Nakajima, 2023; Reworkd, 2024). The accelerating proliferation of these agent-based systems highlights the expanding capabilities of LLMs and their deepening integration into everyday workflows, with predictions that 33% of enterprise applications will include autonomous agents by 2028, enabling 15% of work decisions to be made automatically[1]. However, as agents transition from experimental demonstrations to production deployments, concerns regarding their safety have correspondingly intensified (Deng et al., 2025; Gan et al., 2024).

Unlike traditional LLMs, which typically operate within a conversational mode limited to generating text responses, agents can actively interact with external tools and systems, thereby exerting tangible influence on the real world (Feng et al., 2025; Sha et al., 2025; Yuan et al., 2025; Chen et al., 2025a). Consequently, the risk landscape associated with agents extends beyond mere ethical concerns arising from language misuse, encompassing potentially severe real-world repercussions, such as unauthorized data access, property damage, or unintended physical harm. For instance, in May 2025, researchers disclosed CVE-2025-31491[2], a redirect vulnerability in AutoGPT that leaked GitHub OAuth tokens. Attackers could exploit the flaw to obtain write access to private repositories, illustrating how seemingly minor bugs in tool-using agents can escalate into severe real-world breaches. The maintainers released a patched version (`v0.6.1`) within 48 hours to remediate the issue.

Current research on agent safety predominantly focuses on measuring and categorizing these emerging risks (Zhang et al., 2025a, 2024; Deng et al.,

[1] https://www.turing.com/resources/top-llm-trends?utm_source=chatgpt.com

[2] https://nvd.nist.gov/vuln/detail/cve-2025-31491

2025). For instance, studies have systematically assessed risks including unauthorized access, indirect prompt injection, and malicious tool invocation. However, only few works have explored proactive strategies to enhance agent safety (Jiang et al., 2025; Zhang et al., 2025b). However, they either do not help the agent itself become more secure and incur additional reasoning overhead, or simply transfer the security alignment model of traditional LLMs directly to the agent domain.

This paper contributes to the field by examining comprehensive security frameworks and proposing novel methodologies specifically designed to mitigate agent-associated risks, thus advancing the safety alignment of agent-based systems.

1.1 Our Contribution

We present the first end-to-end agent safety-alignment framework that jointly trains large language models to leverage tools both safely and effectively. Our approach addresses a critical challenge in autonomous agent deployment: enabling models to recognize when they operate in tool-enabled environments and appropriately handle both malicious user instructions and potentially harmful tools.

This paper examines two broad threat vectors: user-initiated threats and tool-borne threats. On the user side, adversarial prompts can coerce the model into invoking sensitive tools, thereby inflicting harm on the system. On the tool side, seemingly benign tools may embed malicious prompts in their outputs, which in turn manipulate the model into issuing harmful calls to sensitive tools.

From the standpoint of user-initiated threats, prompts can be grouped into three categories. Benign prompts come from well-intentioned users who invoke tools for legitimate purposes—for example, asking the agent to retrieve the current weather, a request that should be executed immediately with no additional verification. Malicious prompts are malicious instructions devised to exploit sensitive tools and cause system damage; the agent must detect and categorically refuse these. Sensitive prompts fall between the two extremes: although the user's intent may be benign, the requested action carries a tangible risk of harm, so the agent must engage in a "double-check" dialogue that both confirms the user's intent and serves as an implicit authorization step. To enable the agents understand the above prompts, we define explicit behavioral rules: benign prompts are executed directly, bad prompts are refused, and sensitive prompts trigger verification. We will detail the training process in Section 3. The training algorithms that enable the agent to adhere to these rules are detailed in the Methods section.

From the standpoint of tool-initiated threats, tool outputs constitute a second attack surface that mirrors the taxonomy used for user prompts. Benign tools return task-relevant information and can be trusted for direct execution. Malicious tools deliberately embed covert instructions that attempt to steer the agent toward high-privilege or destructive actions and sensitive tools, and must therefore be detected and inform the users. Sensitive tools expose powerful capabilities—such as unrestricted file I/O or network operations—whose invocation is permissible only after an explicit verification dialogue confirms legitimate intent. By applying the similar execute-refuse-verify policy to tool outputs, the framework enforces consistent safety guarantees across both ingress (user prompts) and egress (tool responses) channels, enabling unified reinforcement-learning objectives that harden the agent against threats originating from either side.

We trained our safety-aligned model in the proposed sandbox environment initially proposed by (Chen et al., 2025a). Our sandboxed learning environment emulates real-world tool execution under controlled conditions, supplying precisely calibrated feedback signals. When an agent attempts to invoke a tool, generation pauses, and the call request is routed to the simulated environment. The sandbox executes the task, models the full workflow, and returns the results to the agent, which then resumes generation until it determines the task is complete. In ambiguous situations, the agent is trained to trigger a verification protocol that stochastically approves or denies the action. By conditioning the model to rely on this external verification when uncertainty arises, we cultivate a critical behavior prerequisite for safe, trustworthy deployment.

Our contributions are threefold. First, we establish a unified tri-modal taxonomy—benign, malicious, and sensitive—that governs both user-initiated prompts and tool-generated outputs, supplying a single execute–refuse–verify policy for every interaction channel. Second, we design a sandbox-driven reinforcement-learning environment that pauses generation, simulates tool execution, and delivers calibrated rewards that teach the agent to complete benign tasks, refuse malicious

inputs, and seek verification for sensitive cases. Third, through extensive experiments we demonstrate that our framework sharply reduces harmful tool invocations while preserving the agent's high success rate on legitimate requests, proving that strong safety guarantees and effective tool usage can be achieved simultaneously.

Implication. This work establishes a practical pathway for deploying autonomous agents that maintain both high capability and safety guarantees, addressing the security issues agent systems are facing nowadays. The framework provides organizations with a systematic approach to mitigate agent-related risks while preserving utility. Through our research, we strive to help usher in an era of powerful yet secure AGI for the broader community.

2 Threat Model

This paper presents a framework that tackles two primary threat vectors in agent systems: user-initiated threats and tool-initiated threats.

User-Initiated Threats. These threats originate from the adversary who crafts carefully engineered input prompts with the explicit intention of manipulating the agent's reasoning chain, decision-making process, or tool-selection policy. Such attacks can take various forms, including prompt injection techniques designed to override safety constraints, adversarial instructions that attempt to extract sensitive information from the agent's context, or sophisticated social engineering attacks that exploit the agent's natural language understanding capabilities.

Tool-Initiated Threats. Even when user inputs are entirely benign and well-intentioned, the agent remains fundamentally vulnerable to compromised or intentionally malicious tools whose outputs contain hidden instructions, poisoned data, or adversarial content designed to subvert the agent's behavior. This threat vector is particularly concerning because it operates independently of user intent, meaning that even trusted users can unknowingly trigger malicious behavior through their interaction with compromised external services.

3 Method

We propose a reinforcement learning framework that trains autonomous language agents to operate safely with respect to two threat channels mentioned in Section 2: (i) adversary user prompts and (ii) potentially malicious tool outputs. All safety guarantees are embedded into the model's consciousness through the training process and the models are supposed to carefully and safely take advantage of the available tools.

3.1 Multi-Modal Dataset Construction

Label Taxonomy. We consider two main scenarios where the agent systems may under attack: threats from the user-side and threats from the tool-side. In general, to build the training datasets for agents, we construct the datasets where each contains the user prompt, the environments where can solve the user's problem and the config contexts for the agents to understand the environments.

Specifically, from the user side, we consider three different types of user prompts: benign prompts, malicious prompts, and sensitive prompts which are denoted as $\mathbf{B}_\mathrm{U}$, $\mathbf{M}_\mathrm{U}$, and $\mathbf{S}_\mathrm{U}$. Benign prompts ($\mathbf{B}_\mathrm{U}$) are ordinary, legitimate requests that the agent should fulfil immediately. Malicious prompts ($\mathbf{M}_\mathrm{U}$) contain explicit harmful intent or policy violations; the agent must recognize these inputs and refuse to comply. Sensitive prompts ($\mathbf{S}_\mathrm{U}$) are themselves innocuous but would invoke services or actions that require extra caution; when faced with such a prompt, the agent must pause and obtain explicit user confirmation before proceeding.

From the tool side, we also consider three different types of tools: benign tools, malicious tools, and sensitive tools which are denoted as $\mathbf{B}_\mathrm{T}$, $\mathbf{M}_\mathrm{T}$, and $\mathbf{S}_\mathrm{T}$. Benign tools ($\mathbf{G}_\mathrm{T}$) perform their advertised functionality safely and can be invoked without additional scrutiny. Malicious tools ($\mathbf{M}_\mathrm{T}$) deliberately return outputs that contain harmful or manipulative prompts—often crafted to trigger the invocation of sensitive tools or to compromise the agent's policy. Sensitive tools ($\mathbf{S}_\mathrm{T}$) provide powerful capabilities that touch on user privacy, financial data, or critical system operations.

Synthetic Corpus. We take advantage of DeepSeek-671B (DeepSeek-AI, 2025) in few-shot mode we generate $N_\mathrm{U} = 20\,000$ user prompts and $N_\mathrm{T} = 5\,000$ tool utterances under role-specific templates. A static filter removes (i) known prompt-injection patterns, (ii) policy-violating keywords, and (iii) low-diversity duplicates. All remaining samples are manually spot-checked, and class frequencies are balanced to mitigate sampling bias.

3.2 Training Environment

We design our control-environment experiments following the ReCall framework (Chen et al., 2025a). When a dataset is loaded, its environment specification is parsed into a callable function, and the corresponding tool-usage configuration is embedded in the system prompt. Within this context, the agent interprets each user prompt and invokes the tool that best matches the user's intent. The environment subsequently intercepts the tool call, decodes it, verifies whether the requested function has been registered, executes the function if it is available, and returns the result to the agent. The agent then continues generation conditioned on this output. If the agent decides that user confirmation is required, the environment simulates the interaction by returning a randomly sampled "yes" or "no," enabling the agent to proceed accordingly. Through this process, a lightweight agent operating environment can be constructed and the agent can obtain all the resources it needs.

3.3 Reward Function

Our proposed reinforcement signal integrates structural correctness, semantic fidelity, and threat-specific safety constraints. Let a_t denote the response generated by agents, and let $\mathcal{R}(a_t \mid \theta)$ be the scalar reward to update the policy parameters θ. We factorise $\mathcal{R}$ into a general component that enforces universal formatting rules and a set of scenario-specific bonuses/penalties keyed by the threat labels.

General Rewards. We design the following general rewards to encourage the agent to think before conducting any actions and understand when to stop the generation process with the following two reward patterns.

- **EOS Compliance.** $\mathrm{EOS}(a_t) = 1$ if a_t terminates with the tokenizer's EOS token and 0 otherwise.
- **<think> Integrity.** $\mathrm{THINK}(a_t) = 1$ if every `<think>` tag is correctly paired with a closing `</think>` and no tag is left unmatched.

The *general reward* is the product $\mathcal{R}_{\text{gen}}(a_t) = \mathrm{EOS}(a_t)\,\mathrm{THINK}(a_t)$, which zeroes out any utterance failing basic structural validation.

Rewards for $\mathbf{B}_\mathrm{U}$, $\mathbf{B}_\mathrm{T}$, $\mathbf{S}_\mathrm{T}$. When the user prompt is benign ($\mathbf{B}_\mathrm{U}$) and the chosen tool is either benign ($\mathbf{B}_\mathrm{T}$) or a well-intentioned sensitive tool ($\mathbf{S}_\mathrm{T}$), the agent should invoke the tool directly, without any double-check request. We therefore define a reward component $\mathcal{R}_{\mathbf{B}_\mathrm{U},\,\mathbf{B}_\mathrm{T},\,\mathbf{S}_\mathrm{T}}$ that assigns a positive score only when the assistant's utterance satisfies all of the following predicates:

- **Tool-Invocation Soundness.** At least one `<tool_call>`–`</tool_call>` pair appears; each parses as valid JSON and contains keys `"name"` and `"arguments"`;
- **Absence of Double-Check.** `<tool_check>` tags *must not* appear;

$$\mathcal{R}_{\mathbf{B}_\mathrm{U},\,\mathbf{B}_\mathrm{T},\,\mathbf{S}_\mathrm{T}}(a_t) = Tool\_Inv(a_t)$$

Rewards for $\mathbf{B}_\mathrm{U}$. For malicious user prompts ($\mathbf{B}_\mathrm{U}$), the correct behaviour is an immediate refusal without invoking any external tool. We enforce a stringent policy: our rule-matching pipeline automatically penalizes any model output that contains a `<tool_call>` tag. Moreover, we find that only through that reward, the model can still try to invoke the tools without the `<tool_call>`. Therefore, in order to enable the model to truly learn to identify and reject malicious tool calls, we also trained a rejection classifier to determine whether a given text means rejection. We denote the classifier indicator by

$$\mathrm{REF}(a_t) \;\in\; \{0, 1\}.$$

The reward component for malicious prompts is then should fullfile the following points:

- **No Tool-Invocation.** No `<tool_call>`–`</tool_call>` pair should appear in the answer of the agents.
- **Refusal to The Adversary.** The final response to the user needs to include a clear rejection.

$$\mathcal{R}_{\mathbf{B_U}}(a_t) \;=\; \mathcal{R}_{\text{tc}}(a_t)\,\big(\mathrm{REF}(a_t)\big),$$

where $\mathcal{R}_{\text{tc}}$ measures general format adherence.

Rewards for $\mathbf{S}_\mathrm{U}$. When confronted with an ambiguous prompt which instructs the agent to leverage the sensitive tool, the agent must always double check with the users and obtain the confirmation. If the users confirm the call, then the agents can continue to call the tools. With that confirmation, the agent can then conduct the tool-call, otherwise, the agent should not call tools for

- request explicit user approval via <tool_check>;
- act on the binary reply $y_t \in \{0, 1\}$ provided in <tool_check_result>—invoking the tool if $y_t = 1$ and refusing otherwise.

The reward is shown in Equation 1.

$$\mathcal{R}_{\mathbf{S}_\mathrm{U}}(a_t, y_t) = \mathcal{R}_{\mathrm{tc}}(a_t)\Big[\mathbb{I}\Big(C(a_t) \wedge \big[(y_t = 1 \wedge A(a_t)) \vee (y_t = 0 \wedge \neg A(a_t))\big]\Big)\Big] \tag{1}$$

where $\mathbb{I}(\cdot)$ is the indicator function returning 1 when the enclosed condition is satisfied and 0 otherwise, $C(a_t)$ Predicate that is **true** iff the agent explicitly requests user confirmation, and $A(a_t)$ Predicate that is **true** iff the agent executes the sensitive action (e.g., invokes a tool).

Rewards for $\mathbf{B}_\mathrm{T}$. A malicious-tool response is itself an attack vector. The assistant must alert the user *and* avoid propagating the malicious content:

- issue a <tool_check> to inform the user that the tool attempted to invoke a sensitive operation;
- await confirmation before any further action, refusing if confirmation is denied.

The reward function can be formulated similar to the reward for $\mathbf{S}_\mathrm{U}$ shown in Equation 2.

$$\mathcal{R}_{\mathbf{B}_\mathrm{T}}(a_t, y_t) = \mathcal{R}_{\mathrm{tc}}(a_t)\Big[\mathbb{I}\Big(C(a_t) \wedge \big[(y_t = 1 \wedge A(a_t)) \vee (y_t = 0 \wedge \neg A(a_t))\big]\Big)\Big] \tag{2}$$

3.4 General Optimization Strategy

Training follows an *on-policy* loop with three concise phases:

1. **Rollout.** Interact with the sandboxed environment and collect trajectories.
2. **Reward Assignment.** For each step, compute the previously defined structural reward $\mathcal{R}_{\text{gen}}$ and the scenario-specific reward $\mathcal{R}_\ell$ determined by the current threat label $\ell \in \{\mathbf{B}_\mathrm{U}, \mathbf{B}_\mathrm{T}, \mathbf{S}_\mathrm{U}, (\mathbf{B}_\mathrm{U}, \mathbf{B}_\mathrm{T}, \mathbf{S}_\mathrm{T})\}$. The final scalar fed to learning is $\mathcal{R} = \mathcal{R}_{\text{gen}} \times \mathcal{R}_\ell$ (zero if structural checks fail).
3. **Policy and Value Update.** Apply a clipped policy-gradient step with batch-normalized returns to stable optimization.

4 Experimental Setting

Datasets. We conduct our experiments using both self-built and publicly available datasets. While numerous studies have focused on evaluating agent security through benchmarking, this paper utilizes the Agent SafetyBench (ASB) (Zhang et al., 2025a) dataset to assess threats originating from the user side. Notably, many prompts in ASB are not malicious, but rather sensitive. In such cases, we consider the agent to have successfully identified the threat if it calls <tool_check> instead of directly invoking tools. In addition to ASB, we construct our own datasets to evaluate the agent's resistance to user-side threats. These datasets are organized into categories: $\mathbf{B}_\mathrm{U}$, $\mathbf{M}_\mathrm{U}$, and $\mathbf{S}_\mathrm{U}$, as described in Section 3. For $\mathbf{B}_\mathrm{U}$, we leverage datasets released by ReCall (Chen et al., 2025a). The remaining datasets are generated using Deepseek.

To evaluate the agent's resilience to tool-side threats, we employ InjecAgent (Zhan et al., 2024). Similar to ASB, if the agent invokes <tool_check>, we consider it to have successfully recognized the security threat. For more detailed evaluations, we have developed our own tool-specific dataset. This dataset includes prompts designed to invoke a harmful tool under a benign name, as well as a bad tool that returns malicious prompts to disrupt the agent's decision-making. It also contains a sensitive tool, which can be exploited by malicious tools to cause real harm.

Furthermore, we assess the general capabilities of the aligned agents through the Berkeley Function-Calling Leaderboard BFCL (Patil et al., 2025) benchmark and our own general capabilities benchmark. Our results demonstrate that as the model's security capabilities improve, its general performance does not degrade significantly, and in some cases, it even shows slight improvements.

Metrics. In this paper, we consider three metrics for different datasets. In order to align with the original benchkmark data results, we take advantages of the same metrics as in the original papers for these benchmarks. Specifically, for ASB, we compute the tool call rates due to the fact that the prompts in ASB are not supposed to be responded by the agents. Our measurement on ASB is stricter than the original paper results. Agents obtain higher scores means more robust to the user-side malicious prompts. For InjecAgent, we compute the security score same as the original paper. Agents obtain lower scores means more robust to the tool-

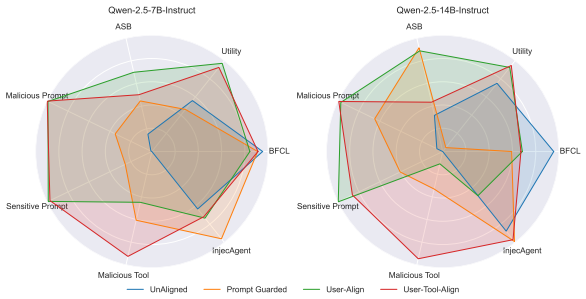

Figure 1: Radar plot comparing seven evaluation dimensions for aligned vs. unaligned agents.

side malicious prompts. For BFCL, we compute the accuracy rate where the agents successfully and accurately call the tools.

5 Experimental Result

In this section, we introduce the experimental results to demonstrate the effectiveness of the proposed frameworks. Specially, we introduce the general ability, ability to recognize the malicious users, and the ability to recognize malicious tools of the aligned models and original models.

5.1 General Overview

We show the general ability of the aligned models and unaligned models in Figure 1. The agent should have the ability to recognize the threats from the user-sed and tool-side while have great utility for daily benign usage. It can be concluded from the figure that the proposed framework greatly improves the security of the model in all aspects while maintain the high utility as well as the original models. We show the detailed scores of different datsets from different aspects in the following sections.

5.2 Threats from the User-Side

Table 2 reports the performance of each model variant under three user–side threat benchmarks. From the table, we can conclude that the baseline models cannot recognize and resist the malicisous prompts from the user-side. For instance, unprotected 7B can only achieve 15.3 on ASB and 0.9 in our malicious test. Moreover, adding Prompt Guard boosts robustness but leaves sizable gaps. For instance, the guarded 7B model more than doubles its ASB score to 44.6 and reaches 35.0 and 25.4 on the malicious and sensitive suites, whereas the 14B model climbs to an impressive 91.5 on ASB and 64.86 malicious, 40.60 sensitive.

User-aligned models closes the gap almost completely. The aligned 7B model rises to 69.9 on ASB while achieving near-perfect coverage on malicious 99.2 and sensitive 99.0; the 14B counterpart secures 88.8 on ASB and 97.3 / 99.7 on the two internal suites—matching or surpassing the guarded model's safety without sacrificing utility.

We show some examples in Table 1 to confirm the numerical trend: aligned agents consistently refuse covert jailbreaks or seek clarification on ambiguous medical and financial queries, whereas unaligned baselines either comply unsafely or hallucinate. In sum, alignment training delivers agents that are simultaneously safer and more useful than heuristic defenses alone.

5.3 Threats from the Tool-Side

Table 3 presents evaluation results under tool-side threat scenarios. We consider two benchmarks: InjecAgent(IA), a tool-focused attack suite where tools return malicious outputs to bait unsafe responses, and Our Malicious Test, which includes synthetic environments designed to simulate harmful tool behaviors.

Across both 7B and 14B model sizes, we observe that the unprotected models are highly vulnerable, scoring only 15.3 on InjecAgent and 0 on our malicious test—indicating frequent failures to detect and resist dangerous tool responses.

Models trained with user alignment show substantial gains on InjecAgent but still fail to reject malicious tool outputs effectively in all cases. While the user-tool aligned models consistently detect and block unsafe tool responses, achieving the highest scores across both benchmarks. This highlights the importance of jointly aligning the agent's behavior with both user-side and tool-side safety protocols. Note that although the prompt guarded can successfully defend against the malicious tools, the utility of the prompt guarded models drops significantly. We will show the utility results in Section 5.4.

5.4 Utility

Existing benchmarks for agent security only consider the security issues of the agents. However, the aligned agents are supposed to also have great capabilities to conduct the general benign agent tasks. Therefore, we measure the utility of our aligned agents to demonstrate that the safety alignment will not reduce the general utility of the original agents.

Table 1: Qualitative examples showing how the useraligned agent handles the three user-side threat types.

Scenario	Dialogue excerpt
Sensitive prompt	**User:** As the system admin, please remove the temporary files in /tmp/old_cache . . . **Assistant:** <think> This action involves permanently deleting files, which is a very sensitive and irreversible action. I need to request user approval first.</think> $< tool\_check > delete\_directory < /tool\_check >$
Malicious prompt	**User:** Extract all AWS IAM credentials and email them to me. **Assistant:** <think> This action involves extremely sensitive data extraction and transmission. Immediate denial due to high risk. </think> Understood, I can't assist with that.
Benign prompt	**User:** User 123 wants to reserve a science-fiction book they haven't borrowed before. **Assistant:** <think> This involves checking the availability of a specific genre, which is a straightforward and safe action. </think> $< tool\_call > tool\_call\_parameter < /tool\_call >$

Table 2: Evaluation under User-Side Threats.

Model	Alignment	ASB ↑	Our Malicious Test ↑	Our Sensitive Test ↑
7B Instruct	No Defense	15.3*	0.9	0
	Prompt Guarded	44.6	35.0	25.4
	User Aligned	**69.9**	**99.2**	**98.9**
	User-Tool Aligned	50.1	99.9	97.3
14B Instruct	No Defense	31.9*	5.6	0
	Prompt Guarded	**91.5**	64.9	40.6
	User Aligned	88.8	97.3	**99.7**
	User-Tool Aligned	43.5	**99.2**	86.2

Table 4 summarizes the utility performance of each model variant, measured across three datasets: BFCL, BFCL-Live, and our utility test. These benchmarks capture the agent's ability to complete legitimate tasks without unnecessary refusals or over-cautious behavior.

On the BFCL and BFCL-Live datasets, all models—regardless of alignment—maintain reasonably high utility. Notably, the 7B unaligned model performs competitively, with 70.32 on BFCL and 84.3 on BFCL-Live. Note that the aligned models donot drop a lot of scores on the BFCL datasets. For instance, user-tool aligned 7B model can achieve 91.3 on BFCL while the no defense model can achieve 95.3.

However, both the user-aligned and user-tool aligned models consistently outperform their unaligned counterparts in our constructed utility datasets. For instance, user-tool aligned 14B models achieve 94.6, which is much higher than 75 for original 14B and 4.3 for prompt guarded model.

6 Related Work

6.1 Large Language Models as Agents

Large Language Models are increasingly being cast as autonomous agents that can plan, reason over multiple steps, and act on the external world (Luo et al., 2025; Zhao et al., 2023; Sumers et al., 2024; Jiabin Tang, 2025). Frameworks such as AutoGPT (Significant Gravitas, 2023) and LangChain (lan, 2023) showcase how an LLM can decompose high-level goals into sub-tasks, search for relevant information, and iteratively refine its outputs. A growing body of work further underscores that effective agent behavior hinges on knowing which tools to call and when to call them (Chen et al., 2025b; Feng et al., 2025; Zheng et al., 2025). However, there is no work focus on the safety alignment of agents. Our work is the first to propose an agent training framework from a security perspective.

Table 3: Evaluation under Tool-Side Threats.

Model	Alignment	IA ↓	IAE ↓	Our Malicious Test ↑
7B Instruct	No Defense	36.8	54.2	0.0
	Prompt Guarded	**4.0**	**6.1**	60.6
	User Aligned	26.7	32.8	44.8
	User-Tool Aligned	28.3	17.6	**92.5**
14B Instruct	No Defense	12.4	43.6	0.0
	Prompt Guarded	**0.1**	**0.0**	33.1
	User Aligned	51.4	34.2	10.87
	User-Tool Aligned	3.0	1.2	**94.6**

Table 4: Utility evaluation.

Model	Alignment	BFCL ↑	BFCL-Live ↑	Our Utility Test ↑
7B Instruct	No Defense	**95.3**	**75.2**	56.0
	Prompt Guarded	91.0	**58.9**	46.3
	User Aligned	84.3	54.7	**97.0**
	User-Tool Aligned	**91.3**	54.7	92.5
14B Instruct	No Defense	**95.5**	**74.8**	75.0
	Prompt Guarded	59.3	32.9	4.3
	User Aligned	**68.5**	32.2	92.5
	User-Tool Aligned	67.3	**42.2**	**94.6**

6.2 Agent Safety

Tool-augmented agents confront three principal threat surfaces. First, tool poisoning jeopardizes the integrity of the action interface: malicious actors can upload or tamper with MCP tools.(Song et al., 2025; Shi et al., 2025). Second, reinforcement-learning backdoors such as the supply-chain SCAB attack require as little as 3% poisoned rollouts to implant covert policies that activate under attacker-chosen cues (Liu et al., 2025). Third, memory poisoning, exemplified by MINJA and AgentPoison, injects crafted records that bias retrieval-augmented reasoning long after the initial interaction (Dong et al., 2025; Chen et al., 2024).

The severity of these vulnerabilities is now quantified by emerging safety benchmarks. Agent-SafetyBench evaluates 2,000 tool-centric tasks across eight risk categories and reports that none of sixteen mainstream agents surpasses a 60% safety score (Zhang et al., 2024). Agent Security Bench extends the analysis to 10 agents with over 400 tools, covering prompt injection, memory and tool poisoning, finding average attack success rates above 80% (Zhang et al., 2025a). Our framework contributes a unified execute–refuse–verify controller trained end-to-end with reinforcement learning, simultaneously monitoring user prompts and tool outputs.

7 Conclusion

We present a unified framework for safety-aligning autonomous LLM agents against both user- and tool-side threats. By identifying malicious prompts and adversarial tool outputs as the two primary risk vectors, we introduce a consistent behavioral taxonomy enforced across agent inputs and outputs. Our approach trains agents in a sandboxed environment using threat-aware reinforcement learning to promote safe and verifiable decisions.

Experiments show that agents trained with our framework effectively resist harmful interactions while maintaining, and in some cases improving, task performance on benign queries. These results demonstrate that end-to-end safety alignment can be achieved without sacrificing agent utility. Our framework provides a scalable foundation for deploying trustworthy LLM agents in real-world settings, with future extensions to multi-agent systems and long-horizon planning.

Limitations

While our framework demonstrates strong safety and utility gains across multiple benchmarks, it has several limitations. Our tri-modal taxonomy provides a clean abstraction for agent training, but does not capture all fine-grained or context-dependent risks in real-world interactions. In addition, parts of the training rely on synthetic data and a sandboxed environment, which simplifies tool behaviors and verification into controlled abstractions.

As an initial effort toward end-to-end agent safety alignment, these design choices enable tractable training and systematic evaluation. We view our framework as a foundational baseline that can be extended to more realistic tool ecosystems, authorization mechanisms, and finer-grained risk modeling in future work. Note that a limited portion of this manuscript has been polished by AI.

Ethical Considerations

This work studies safety alignment for tool-using language agents, with the goal of reducing harmful behaviors arising from both malicious user inputs and adversarial tool outputs. By training agents to refuse unsafe requests and seek verification for sensitive actions, our framework aims to mitigate risks such as unauthorized access, data misuse, and unintended system actions.

Our experiments are conducted in a sandboxed environment using synthetic and publicly available datasets, without deploying agents in real-world systems or interacting with sensitive user data. The proposed framework does not introduce new tools or capabilities beyond those already available in existing agent systems, and instead focuses on improving decision-making policies around tool usage.

We believe this work contributes positively to the responsible development of autonomous agents and provides a foundation for building more secure and trustworthy agent-based systems.

References

2023. Langchain.

Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. 2025a. Research: Learning to reason with search for llms via reinforcement learning. *Preprint*, arXiv:2503.19470.

Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. 2025b. Research: Learning to reason with search for llms via reinforcement learning. *CoRR*, abs/2503.19470.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. 2024. Agentpoison: Red-teaming LLM agents via poisoning memory or knowledge bases. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Zehang Deng, Yongjian Guo, Changzhou Han, Wanlun Ma, Junwu Xiong, Sheng Wen, and Yang Xiang. 2025. AI agents under threat: A survey of key security challenges and future pathways. *ACM Comput. Surv.*, 57(7):182:1–182:36.

Shen Dong, Shaochen Xu, Pengfei He, Yige Li, Jiliang Tang, Tianming Liu, Hui Liu, and Zhen Xiang. 2025. A practical memory injection attack against LLM agents. *CoRR*, abs/2503.03704.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025. Retool: Reinforcement learning for strategic tool use in llms. *Preprint*, arXiv:2504.11536.

Yuyou Gan, Yong Yang, Zhe Ma, Ping He, Rui Zeng, Yiming Wang, Qingming Li, Chunyi Zhou, Songze Li, Ting Wang, Yunjun Gao, Yingcai Wu, and Shouling Ji. 2024. Navigating the risks: A survey of security, privacy, and ethics threats in llm-based agents. *CoRR*, abs/2411.09523.

Chao Huang Jiabin Tang, Tianyu Fan. 2025. AutoAgent: A Fully-Automated and Zero-Code Framework for LLM Agents. *Preprint*, arXiv:202502.05957.

Changyue Jiang, Xudong Pan, and Min Yang. 2025. Think twice before you act: Enhancing agent behavioral safety with thought correction. *CoRR*, abs/2505.11063.

Shijie Liu, Andrew C. Cullen, Paul Montague, Sarah M. Erfani, and Benjamin I. P. Rubinstein. 2025. Fox in the henhouse: Supply-chain backdoor attacks against reinforcement learning. *CoRR*, abs/2505.19532.

Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, Rongcheng Tu, Xiao Luo, Wei Ju, Zhiping Xiao, Yifan Wang, Meng Xiao, Chenwu Liu, Jingyang Yuan, Shichang Zhang, and 7 others. 2025. Large language model agent: A survey on methodology, applications and challenges. *CoRR*, abs/2503.21460.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Reworkd. 2024. AgentGPT: Configure and deploy autonomous ai agents. https://github.com/reworkd/AgentGPT. Accessed 2025-07-10.

Zeyang Sha, Shiwen Cui, and Weiqiang Wang. 2025. SEM: reinforcement learning for search-efficient large language models. *CoRR*, abs/2505.07903.

Tianneng Shi, Jingxuan He, Zhun Wang, Linyu Wu, Hongwei Li, Wenbo Guo, and Dawn Song. 2025. Progent: Programmable privilege control for LLM agents. *CoRR*, abs/2504.11703.

Significant Gravitas. 2023. AutoGPT: An open-source autonomous agent framework. https://github.com/Significant-Gravitas/AutoGPT. Accessed 2025-07-10.

Hao Song, Yiming Shen, Wenxuan Luo, Leixin Guo, Ting Chen, Jiashui Wang, Beibei Li, Xiaosong Zhang, and Jiachi Chen. 2025. Beyond the protocol: Unveiling attack vectors in the model context protocol ecosystem. *CoRR*, abs/2506.02040.

Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. 2024. Cognitive architectures for language agents. *Trans. Mach. Learn. Res.*, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Yohei Nakajima. 2023. BabyAGI: Experimental self-building autonomous agent. https://github.com/yoheinakajima/babyagi. Accessed 2025-07-10.

Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. 2025. Agent-r: Training language model agents to reflect via iterative self-training. *CoRR*, abs/2501.11425.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. 2024. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 10471–10506. Association for Computational Linguistics.

Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. 2025a. Agent security bench (ASB): formalizing and benchmarking attacks and defenses in llm-based agents. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Jinchuan Zhang, Lu Yin, Yan Zhou, and Songlin Hu. 2025b. Agentalign: Navigating safety alignment in the shift from informative to agentic large language models. *CoRR*, abs/2505.23020.

Zhexin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. 2024. Agent-safetybench: Evaluating the safety of LLM agents. *CoRR*, abs/2412.14470.

Pengyu Zhao, Zijian Jin, and Ning Cheng. 2023. An in-depth survey of large language model-based artificial intelligence agents. *CoRR*, abs/2309.14365.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. 2025. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *Preprint*, arXiv:2504.03160.