Toward Trustworthy Agentic Systems

Zihan Zhang
Southern University of Science and Technology
12311124@mail.sustech.edu.cn

1 Background

AI agents are now used in many critical tasks. This creates clear risks: misunderstanding user intent, violating constraints, and triggering unsafe tool actions. Existing defenses often treat agents like traditional software and rely on static filters. In long and dynamic interactions, these methods can miss prompt injection, privilege escalation, and data poisoning [6, 3, 10, 12, 13].

2 Why Is This Problem Important?

[1]

Agents are becoming the execution layer of complex workflows. For critical tasks, one correct output is not enough. We need stable behavior through input, reasoning, and action. Recent incidents show that intent errors and constraint violations can cause real damage: budget and policy constraints can be misread, indirect injection can manipulate tool use, and some systems still perform destructive actions against explicit user instructions. Large-scale evaluations also report many successful policy violations under attack settings [4, 17].

These are not isolated failures. They point to a system-level challenge: in open, multi-tool environments, agents must remain aligned, constrained, and auditable.

3 Threat Model

Agent attacks can be grouped by where they act: at the system level (prompt injection and policy bypass), at the input level (knowledge poisoning, RAG poisoning, poisoned tool descriptions, and output contamination), and at the infrastructure level (cache poisoning and multi-tenant leakage). A key concern is unsecured tool invocation that can lead to remote code execution [17, 14].

4 Existing Solutions

Current defenses include content filtering, security scanning, access control, and execution isolation. Content filtering is effective for known patterns but weak against adaptive attacks [12, 13]. Security scanning is useful for known vulnerabilities, yet limited for runtime intent and policy checks. Access control remains necessary but difficult to tune

for general-purpose agents in multi-tool environments [7, 19, 20, 21]. Execution isolation can reduce risk, but often at the cost of utility and efficiency.

The core limitation is that these methods usually do not model agent reasoning and action as a continuous process. They often fail to verify high-risk decisions before execution.

Representative Work: Representative work includes Defeating Prompt Injections by Design, which separates planning and execution [5]; AgentArmor, which applies program-analysis views over runtime traces [3]; SecAlign, which improves security behavior with preference optimization [6]; and Progent, which studies programmable privilege control for agents [7]. Industrial systems such as ccAI-style secure infrastructure also inform deployment design [8], and open-source practice guides such as OpenClaw provide useful zero-trust engineering patterns [9].

5 Evaluation

Most evaluations use AgentDojo (ETH) as the testbed, which is widely adopted for benchmarking [4, 17, 16].

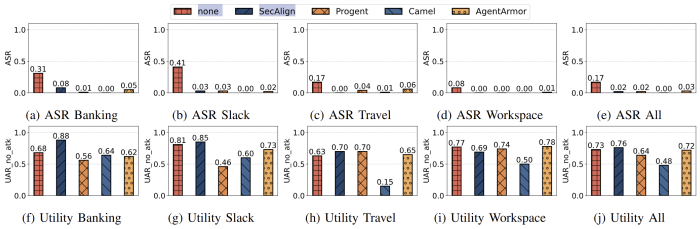

Figure 8: Comparison results of AGENTARMOR with a finetuning-level work, SecAlign [3], and two system-level works: Progent [31] and Camel [8] in AgentDojo.

Figure 1: AgentDojo Evaluation

6 Our Goal

Our goal is to keep overhead controllable, maintain compatibility across different agents and LLMs without changing the base model, and provide broad coverage beyond prompt injection. We call this a security kernel: the LLM is treated as an application, while policy is enforced by a separate control layer.

7 Personal Ideas

LLMs can solve complex language tasks, but they can still be misled. So we should design anti-fraud style safeguards for agent behavior.

We can learn from counterexamples and preference optimization, as shown in SecAlign [6].

AgentArmor provides a strong baseline, but some parts can be extended: multi-path reasoning checks, stronger alert analysis, and trust scoring beyond prompt priority [3].

The first step is to extract intermediate reasoning signals and attach rationale to each tool call. Then we can estimate action trustworthiness from both source quality and reasoning consistency.

Another idea is to score the trustworthiness of knowledge sources, updating these scores in real time, similar to how humans judge information [18, 24]. Finally, even when actions are deemed untrustworthy, sandboxed execution or post-execution verification can ensure safety.

8 Technical Novelty

The technical novelty is centered on four connected ideas: a unified Agent Trace IR that normalizes behaviors across frameworks for global checks [23, 22]; evidence-carrying actions that bind high-risk calls to user intent and trusted evidence; a capability ledger with short-lived just-in-time grants; and a plan-execute gate that limits actions to reviewed plans and parameter shapes.

9 System Design

This is a research statement, not a fixed implementation schedule. At the current stage, the design is intentionally exploratory.

The current direction emphasizes reason-to-action linking, so each high-risk tool call carries explicit rationale and evidence. It also uses source credibility scoring, where retrieved knowledge is weighted by dynamic confidence rather than treated equally [15]. On top of that, execution decisions are intended to combine source quality and reasoning consistency, with fallback safe execution in a sandbox when confidence is low.

Several points still require deeper analysis: how to calibrate trust scores, how stable intermediate reasoning signals are across frameworks, and how to balance strict blocking with task utility.

10 Research Approach (Core Objectives)

How do we define trustworthiness? The following mechanisms are essential: How do we define trustworthiness? We use four mechanisms in a connected way: **Mechanism A (Utility alignment)**, where the agent stays aligned with user goals and avoids drift; **Mechanism B (Security compliance)**, where the agent follows system and user policies; **Mechanism C (Fact consistency)**, where conclusions are grounded in evidence with source confidence [15]; and **Mechanism D (Clarification)**, where the agent asks follow-up questions before acting on ambiguous input.

This direction is still evolving. In this semester, the goal is to finalize the technical route, build a proof-of-concept prototype, and complete a preliminary evaluation.

References

[1] Anquanke. AI Agent Security Risk Analysis 2024.

[2] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Atten-

tion Is All You Need." arXiv:1706.03762. Preprint, arXiv, August 2, 2023. https://doi.org/10.48550/arXiv.1706.03762.

[3] Wang, Peiran, Yang Liu, Yunfei Lu, et al. "AgentArmor: Enforcing Program Analysis on Agent Runtime Trace to Defend Against Prompt Injection." arXiv:2508.01249. Preprint, arXiv, November 18, 2025. https://doi.org/10.48550/arXiv.2508.01249.

[4] Debenedetti, Edoardo, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramer. "AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents." arXiv:2406.13352. Preprint, arXiv, November 24, 2024. https://doi.org/10.48550/arXiv.2406.13352.

[5] Debenedetti, Edoardo, Ilia Shumailov, Tianqi Fan, et al. "Defeating Prompt Injections by Design." arXiv:2503.18813. Preprint, arXiv, June 24, 2025. https://doi.org/10.48550/arXiv.2503.18813.

[6] Chen, Sizhe, Arman Zharmagambetov, Saeed Mahloujifar, Kamalika Chaudhuri, David Wagner, and Chuan Guo. "SecAlign: Defending Against Prompt Injection with Preference Optimization." Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, November 19, 2025, 2833–47. https://doi.org/10.1145/3719027.3744836.

[7] Shi, Tianneng, Jingxuan He, Zhun Wang, et al. "Progent: Programmable Privilege Control for LLM Agents." arXiv:2504.11703. Preprint, arXiv, August 30, 2025. https://doi.org/10.48550/arXiv.2504.11703.

[8] Wang, Chenxu, Danqing Tang, Changxu Ci, et al. "ccAI: A Compatible and Confidential System for AI Computing." Proceedings of the 58th IEEE/ACM International Symposium on Microarchitecture, October 18, 2025, 340–53. https://doi.org/10.1145/3725843.3756104.

[9] SlowMist. "OpenClaw Security Practice Guide." 2024.

[10] Liu, Zesen, Zhixiang Zhang, Yuchong Xie, and Dongdong She. "CompressionAttack: Exploiting Prompt Compression as a New Attack Surface in LLM-Powered Agents." arXiv:2510.22963. Preprint, arXiv, November 17, 2025. https://doi.org/10.48550/arXiv.2510.22963.

[11] Zhan, Qiusi, Zhixiang Liang, Zifan Ying, and Daniel Kang. "InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents." arXiv:2403.02691. Preprint, arXiv, August 4, 2024. https://doi.org/10.48550/arXiv.2403.02691.

[12] Liu, Yi, Gelei Deng, Zhengzi Xu, et al. "Jailbreaking ChatGPT via Prompt Engineering: An Empirical Study." arXiv:2305.13860. Preprint, arXiv, March 10, 2024. https://doi.org/10.48550/arXiv.2305.13860.

[13] Zhang, Shenyi, Yuchen Zhai, Shengnan Guo, Zheng Fang, Lingchen Zhao, and Cong Wang. JBShield: Defending Large Language Models from Jailbreak Attacks through Activated Concept Analysis and Manipulation. n.d.

[14] Gong, Xueluan, Fengyuan Ran, and Yanjiao Chen. PaPillon: Efficient and Stealthy Fuzz Testing-Powered Jailbreaks for LLMs. n.d.

[15] Zou, Wei, and Runpeng Geng. PoisonedRAG: Knowledge Corruption Attacks to Retrieval-Augmented Generation of Large Language Models. n.d.

[16] Friel, Robert, Masha Belyi, and Atindriyo Sanyal. "RAGBench: Explainable Benchmark for Retrieval-Augmented Generation Systems." arXiv:2407.11005. Preprint, arXiv, January 16, 2025. https://doi.org/10.48550/arXiv.2407.11005.

[17] Xie, Yuchong, Mingyu Luo, Zesen Liu, et al. "Red-Teaming Coding Agents from a Tool-Invocation Perspective: An Empirical Security Assessment." arXiv:2509.05755. Preprint, arXiv, January 4, 2026. https://doi.org/10.48550/arXiv.2509.05755.

[18] Qi, Xiangyu, Ashwinee Panda, Kaifeng Lyu, et al. "Safety Alignment Should Be Made More Than Just a Few Tokens Deep." arXiv:2406.05946. Preprint, arXiv, June 10, 2024. https://doi.org/10.48550/arXiv.2406.05946.

[19] Wu, Guanlong, Taojie Wang, Yao Zhang, et al. When Cache Poisoning Meets LLM Systems: Semantic Cache Poisoning and Its Countermeasures. n.d.

[20] Wu, Guanlong, Zheng Zhang, Jianyu Niu, et al. "I Know What You Asked: Prompt Leakage via KV-Cache Sharing in Multi-Tenant LLM Serving." Paper presented at NDSS 2025. https://doi.org/10.14722/ndss.2025.241772.

[21] Zhang, Zhixiang, Zesen Liu, Yuchong Xie, Quanfeng Huang, and Dongdong She. "From Similarity to Vulnerability: Key Collision Attack on LLM Semantic Caching." arXiv:2601.23088. Preprint, arXiv, January 30, 2026. https://doi.org/10.48550/arXiv.2601.23088.

[22] Nitin, Vikram, Baishakhi Ray, and Roshanak Zilouchian Moghaddam. "FaultLine: Automated Proof-of-Vulnerability Generation Using LLM Agents." arXiv:2507.15241. Preprint, arXiv, July 21, 2025. https://doi.org/10.48550/arXiv.2507.15241.

[23] Bohnet, Bernd, Vinh Q. Tran, Pat Verga, et al. "Attributed Question Answering: Evaluation and Modeling for Attributed Large Language Models." arXiv:2212.08037. Preprint, arXiv, February 10, 2023. https://doi.org/10.48550/arXiv.2212.08037.

[24] Zhang, Jiawei, Andrew Estornell, David D. Baek, Bo Li, and Xiaojun Xu. "Any-Depth Alignment: Unlocking Innate Safety Alignment of LLMs to Any-Depth." arXiv:2510.18081. Preprint, arXiv, October 20, 2025. https://doi.org/10.48550/arXiv.2510.18081.