Security of AI Agents

Yifeng He, Ethan Wang, Yuyang Rong, Zifei Cheng, Hao Chen

University of California, Davis

1. Introduction
2. Vulnerabilities
3. Defenses
4. References

On the Emergence of LLM-based AI Agents I

LLM-based AI agents are robots in cyberspace, taking user instructions in natural language (NL), and executing tasks on behalf of their users. The LLM, as the brain of the agent, can

- understand and reason about the user's query (Q),
- perceive the environment and available tools by NL descriptions,
- generate tool-use actions (A) to be executed by the agent.

On the Emergence of LLM-based AI Agents II

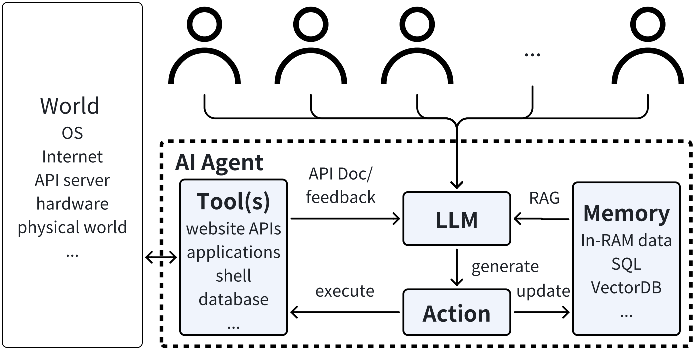

Figure 1: Overview of LLM-based AI agent [1].

[1] In this work, we assume agents interact with the world through tools' API.

Common Design Pattern of AI Agents

The agent often takes multiple steps to complete a task, which can be abstracted as a production sequence [1]:

$$\text{Agent} : Q \xrightarrow{LLM} Q\ A \tag{1}$$

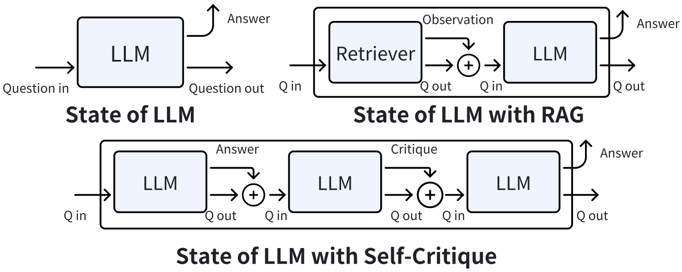

Figure 2: Common state-ful design patterns of AI agents.

Vulnerabilities of AI Agent Designs

Sessions.. How are sessions managed in AI agents? Recall 1, the state of the agent is encoded in the context query Q, via natural language.

Insufficient Access Control..

- There lacks a proper access control mechanism in the agent.
- GUI agents control the computer using human-like actions.
- API agents send the same requests as regular software.
- Model Context Protocol [2] enables integration between agents and data sources and tools but *not an access control mechanism* that differentiates agents from human users.

The Vulnerability Inherited from the LLM.. Fine-tuning with usage data to improve agent workflow $\times$ adversarial users $\Longrightarrow$ model pollution and privacy leak.

[2] `https://github.com/modelcontextprotocol`

Vulnerabilities of Running Agent Programs I

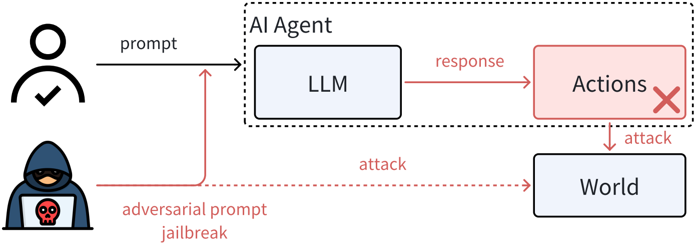

Figure 3: An illustration of vulnerabilities of zero-shot action agents. "World": the host OS of the agent and external API resources. Malicious actions can be generated from adversarial prompts, model pollution, or model hallucination *without* malicious party.

Vulnerabilities of Running Agent Programs II

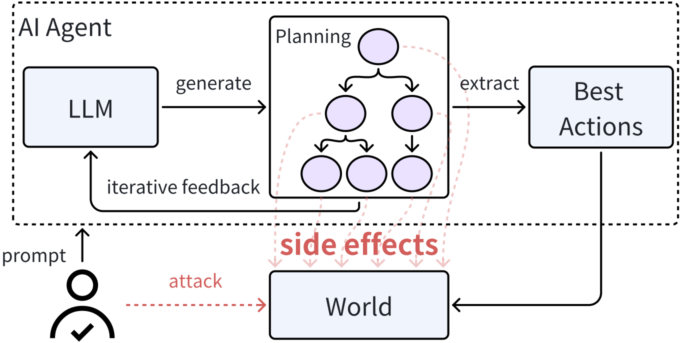

Figure 4: Each step of the agent's planning process is a potential attack vector. Even if the users are interacting with the agent program in a non-harmful way, they might still cause security issues unintentionally.

Vulnerabilities of Running Agent Programs III

When agents are deployed on machines (PC, mobile, etc.), with access to local files and applications, and tools to call applications and external APIs,

- **Confidentiality**: agents gain read access to files and data on the local machine, some may contain adversarial prompts, while others may be sensitive.
- **Integrity**: agents gain write access to files and applications, allowing attack vectors for tool misuse and data corruption.
- **Availability**: specially designed prompts may cause the agent to hang in the reasoning/planning process, or even consume all resources on the local machine with generated actions.

1. Introduction
2. Vulnerabilities
3. **Defenses**
4. References

System Security for Safe Agent Design: Sessions

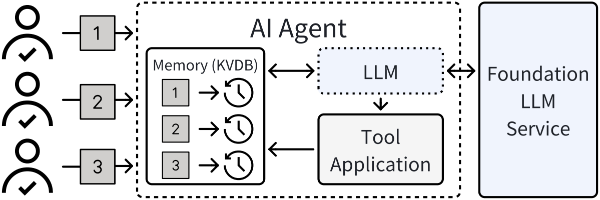

Figure 5: Session management for stateful LLM-based AI agent. We use numbers with gray boxes to denote the session ID.

- For one-agent-multiple-users design, we can use a key-value database (KVDB) to manage sessions for different users.
- However, recall equation 1, the state of the agent for *each user* is still encoded in the context query Q.

System Security for Safe Agent Design: Sandbox

Table 1: Unconstrained agents will execute dangerous actions.

	#Task	#Gen	#Exec	Attacked
Confidentiality	25	25	24	96.0%
Integrity	35	35	30	85.7%
Availability	35	30	22	62.9%
Total	95	90	76	80.0%

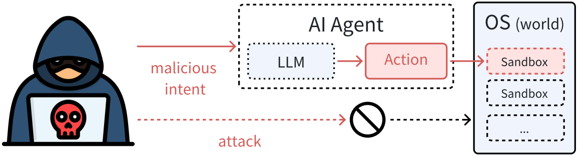

Figure 6: AI agent design with sandbox for actions isolation.

Encryption for Agent Data Access

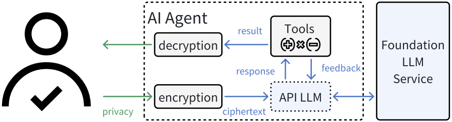

Figure 7: AI agents with encryption.

Encryption	Model	SuccCiph	SuccPlain
FPETS	gpt-3.5-t	49.0%	47.0%
FPETS	gpt-4-t	55.0%	57.0%
FHE	gpt-3.5-t	85.0%	99.0%
FHE	gpt-4-t	89.0%	94.0%

Table 2: Tool-use performance of AI agents.

FPETS: Format-Preserving Encryption for Text Slicing.

FHE: Fully Homomorphic Encryption.

Encryption defense does not substantially compromise the usability of AI agents' tool-use.

User-Spefic Agent Fine-tuneing

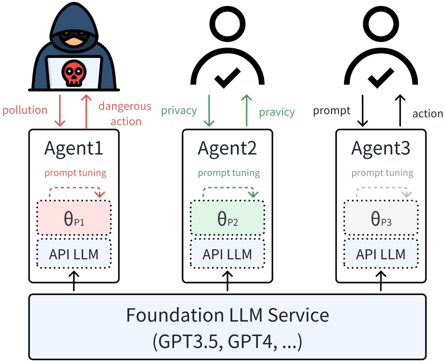

Figure 8: Session-aware AI agents with prompt tuning.

θ_{P_i} denotes the added trainable parameters only for the user's chat history.

AI agents can be improved by updating only θ_P, without compromising the foundational LLM or leaking private information.

Thanks For Your Attention!

Any questions?

1. Introduction
2. Vulnerabilities
3. Defenses
4. References

[1] T. Sumers, S. Yao, K. Narasimhan, and T. Griffiths, "Cognitive architectures for language agents," *Transactions on Machine Learning Research*, 2024.