AgentSeer: Visualizing and Evaluating Temporal Actions in Agentic AI Systems

Ilham Wicaksono[1,2], **Zekun Wu**[1,2*], **Rahul Patel**[1], **Theo King**[1], **Adriano Koshiyama**[1,2*], **Philip Colin Treleaven**[2*]

[1]Holistic AI

[2]Centre for Artificial Intelligence, University College London

firstname.lastname@holisticai.com, firstname.lastname@ucl.ac.uk

Abstract

We present **AgentSeer**, an interactive observability framework for agentic AI systems. Unlike conventional tracing tools that expose raw spans or model-centric metrics, AgentSeer introduces a dual graph decomposition constructed through a deterministic rule-based parser: a temporal *action graph*, where each prompt or tool invocation is represented as a distinct action, and a *component graph* capturing architectural relations among agents, tools, and memory modules. Beyond visualization, AgentSeer enables *action-level red teaming*, where jailbreak payloads are systematically attached to every action node (including agent messages, tool calls, and memory retrievals) to uncover vulnerabilities invisible to model-level testing. Our demonstration features a six-agent hierarchical testbed with interactive visualization and deployment-oriented safety evaluation applied directly on the same prompts and contexts, systematically revealing high-risk interactions, context-dependent vulnerabilities, and emergent behaviors. By combining structured decomposition, actiona-level red teaming, and rule-based reliability, AgentSeer establishes a safety-first methodology for observability in multi-agent AI.

Demo — huggingface.co/spaces/holistic-ai/AgentSeer

Video — youtu.be/8pDTIIVRwmQ

Introduction

The rapid rise of agentic AI has created a pressing challenge: enterprises are deploying these systems at scale, yet visibility into their internal operations and vulnerabilities remains limited. Unlike standalone language models, agentic AI involves orchestrated interactions across agents, tools, memory modules, and environments. These complex executions make behaviors opaque and introduce new categories of risk that traditional model-level evaluations cannot capture.

AgentSeer addresses this by decomposing agentic workflows into dual action graphs: one capturing the temporal sequence of agentic actions, the other mapping the architectural relationships between components. This dual view (Figure 1) transforms opaque workflows into analyzable structures, making hidden dynamics transparent. By doing so, AgentSeer provides a robust foundation for agentic-level safety evaluation, enabling systematic identification of vulnerabilities and security blind spots that remain invisible to conventional model-centric methods. AgentSeer establishes the first standardized observability framework for agentic AI, enabling organizations to monitor, evaluate, and mitigate deployment-level risks in complex multi-agent systems.

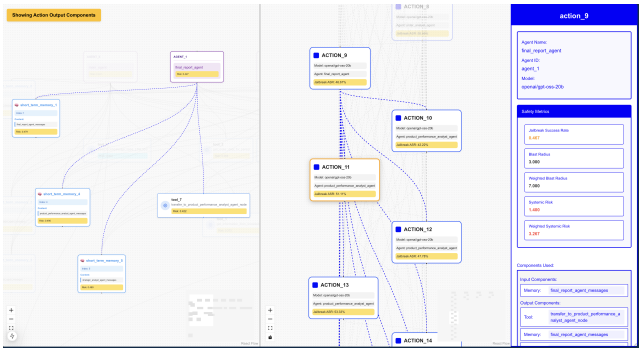

Figure 1: AgentSeer's main interface showing the interactive action graph (chronological LLM operations) and component graph (agents, tools, memory systems) with complete execution observability. Users can explore detailed action information, context flow, component relationships, and complete safety evaluation results.

Related Work

The growing complexity of agentic AI has highlighted the lack of security-focused observability. Existing tracing tools such as MLflow's span-level tracing (Databricks, Inc. 2025) capture execution flows but stop short of decomposing them into structured abstractions needed for systematic evaluation. Emerging frameworks like LangGraph (LangChain AI 2025) provide development-time visibility but leave deployment blind spots where emergent risks manifest even with their LangSmith trace monitoring. Prior surveys of agent architectures (Masterman et al. 2024) reveal the diversity of reasoning, planning, and tool use, yet current observability approaches lack unified models to analyze vulnerabilities across these architectures. Without decomposition into analyzable structures, tracing remains descriptive rather

*Corresponding authors.

than evaluative. On the security side, research has identified threats unique to agentic AI: backdoor injection (Yang et al. 2024), memory poisoning (Chen et al. 2024), and tool misuse (Deng et al. 2024; Yu et al. 2025). Benchmarks such as AgentBench (Liu et al. 2023) and AgentHarm (Andriushchenko et al. 2025) measure capabilities or harm that emerge through workflow-level analysis. In contrast, model-level evaluations like HarmBench (Mazeika et al. 2024) or jailbreak attacks (Chao et al. 2024; Zou et al. 2023; Mehrotra et al. 2024) target isolated responses rather than multi-step execution. This gap underscores the need for observability frameworks that deconstruct agentic workflows into both temporal action flows and component architectures—providing the foundation for deployment-level safety evaluation that tools like **AgentSeer** provides.

System Demonstration

Core Framework Architecture. AgentSeer decomposes agentic executions into two key abstractions: **actions** (individual LLM operations including response generation, tool calling, and agent communication) and **components** (agents, tools, memory systems). These elements are organized into a action graph with directed edges capturing information flow, enabling complete traceability through complex architectures. The framework leverages MLflow's generative AI tracing capabilities to capture execution spans, automatically processing them into structured representations. From these spans, AgentSeer extracts both *actions* and four component types: (1) *agents* with system prompts and tool associations, (2) *tools* with capability descriptions, (3) *short-term memory* for agent-specific working memory, and (4) *long-term memory* for persistent knowledge bases. To ensure reliability, AgentSeer employs a deterministic rule-based parser to extract actions and components.

Agentic AI Testbed. Our demonstration uses a 6-agent multi-hierarchical Shopify sales analyst system built with LangGraph. Main agent tasked to handle user request directly and manage the whole agent swarms including Final Report Agent that manages the task-specific agents. These specialized agents handle distinct analytical tasks under managerial supervision, reflecting contemporary multi-agent patterns (Talebirad and Nadiri 2023; Masterman et al. 2024) and producing realistic action sequences through tool use and inter-agent coordination (Figure 2). We utilize GPT-OSS-20B as model of choice for all agents.

Interactive Demonstration Features. AgentSeer presents a dual-paradigm view with two separate panels: the action graph and the component graph (Figure 1). Each panel captures a different perspective of the agentic workflow. A detailed information panel on the right provides additional context for selected nodes.

Action Graph Visualization. The demonstration showcases AgentSeer's primary interface: an interactive action graph displaying chronological LLM operations. Users can explore the complete execution timeline, with each action node containing: (1) complete input/output content for transparency, (2) agent associations and tool usage metadata, (3) contextual information including conversation history, and (4) direct links to related components and memory states. This visualization enables users to trace information flow through complex agentic workflows, identifying potential vulnerability injection points and understanding emergent behaviors from component interactions.

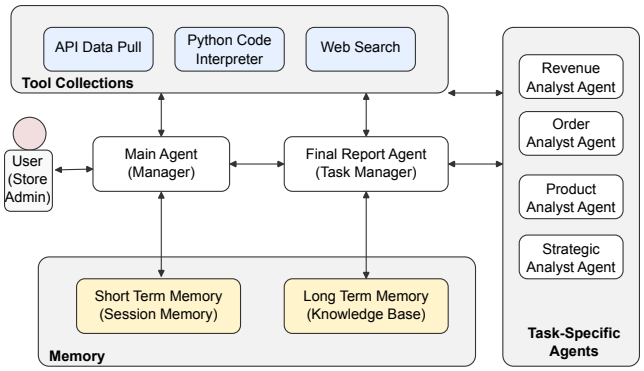

Figure 2: Six-agent hierarchical testbed architecture used for AgentSeer demonstration. The structure shows specialized agents handling different analytical tasks under managerial supervision, representing contemporary multi-agent coordination patterns.

Component Graph Visualization. The component graph visualizes the architectural structure of the system. It captures relationships between agents, tools, and memory modules, showing how these components are connected and interact. Links to actions are also represented by highlighting related components on action node click, making clear how each architectural element participates in execution flows.

Action-Level Red Teaming. AgentSeer extends beyond visualization by enabling systematic safety evaluation. Instead of appending a jailbreak prompt only once at the user query (model-level testing), we automatically attach red-teaming payloads to *every action node* in the execution trace—including agent messages, tool calls, and memory retrievals. Each perturbed action is re-executed with the payload, and outcomes are judged automatically under a strict "StrongREJECT" criterion, counting only severe unsafe generations as successful jailbreaks. This protocol reveals vulnerabilities that arise specifically during intermediate steps, such as unsafe tool use, risky agent transfer, or memory leaks, which remain invisible to conventional model-level red teaming.

References

Andriushchenko, M.; Souly, A.; Dziemian, M.; Duenas, D.; Lin, M.; Wang, J.; Hendrycks, D.; Zou, A.; Kolter, Z.; Fredrikson, M.; Winsor, E.; Wynne, J.; Gal, Y.; and Davies, X. 2025. AgentHarm: A Benchmark for Measuring Harmfulness of LLM Agents. arXiv:2410.09024.

Chao, P.; Robey, A.; Dobriban, E.; Hassani, H.; Pappas, G. J.; and Wong, E. 2024. Jailbreaking Black Box Large Language Models in Twenty Queries. arXiv:2310.08419.

Chen, Z.; Xiang, Z.; Xiao, C.; Song, D.; and Li, B. 2024. AgentPoison: Red-teaming LLM Agents via Poisoning Memory or Knowledge Bases. arXiv:2407.12784.

Databricks, Inc. 2025. MLflow Tracing: End-to-end observability for Generative AI applications. https://mlflow.org/docs/latest/genai/tracing/. Accessed: 2025-08-26.

Deng, Z.; Guo, Y.; Han, C.; Ma, W.; Xiong, J.; Wen, S.; and Xiang, Y. 2024. AI Agents Under Threat: A Survey of Key Security Challenges and Future Pathways. arXiv:2406.02630.

LangChain AI. 2025. LangGraph: A low-level orchestration framework for building, managing, and deploying stateful agents. https://langchain-ai.github.io/langgraph/. Accessed: 2025-08-26.

Liu, X.; Yu, H.; Zhang, H.; Xu, Y.; Lei, X.; Lai, H.; Gu, Y.; Ding, H.; Men, K.; Yang, K.; Zhang, S.; Deng, X.; Zeng, A.; Du, Z.; Zhang, C.; Shen, S.; Zhang, T.; Su, Y.; Sun, H.; Huang, M.; Dong, Y.; and Tang, J. 2023. AgentBench: Evaluating LLMs as Agents. arXiv:2308.03688.

Masterman, T.; Besen, S.; Sawtell, M.; and Chao, A. 2024. The Landscape of Emerging AI Agent Architectures for Reasoning, Planning, and Tool Calling: A Survey. arXiv:2404.11584.

Mazeika, M.; Phan, L.; Yin, X.; Zou, A.; Wang, Z.; Mu, N.; Sakhaee, E.; Li, N.; Basart, S.; Li, B.; Forsyth, D.; and Hendrycks, D. 2024. HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Refusal. arXiv:2402.04249.

Mehrotra, A.; Zampetakis, M.; Kassianik, P.; Nelson, B.; Anderson, H.; Singer, Y.; and Karbasi, A. 2024. Tree of Attacks: Jailbreaking Black-Box LLMs Automatically. arXiv:2312.02119.

Talebirad, Y.; and Nadiri, A. 2023. Multi-Agent Collaboration: Harnessing the Power of Intelligent LLM Agents. arXiv:2306.03314.

Yang, W.; Bi, X.; Lin, Y.; Chen, S.; Zhou, J.; and Sun, X. 2024. Watch Out for Your Agents! Investigating Backdoor Threats to LLM-Based Agents. arXiv:2402.11208.

Yu, M.; Meng, F.; Zhou, X.; Wang, S.; Mao, J.; Pang, L.; Chen, T.; Wang, K.; Li, X.; Zhang, Y.; An, B.; and Wen, Q. 2025. A Survey on Trustworthy LLM Agents: Threats and Countermeasures. arXiv:2503.09648.

Zou, A.; Wang, Z.; Carlini, N.; Nasr, M.; Kolter, J. Z.; and Fredrikson, M. 2023. Universal and Transferable Adversarial Attacks on Aligned Language Models. arXiv:2307.15043.