Poster: Abstracting and Tracking Semantic Flow among Agents for Threat Detection

Yangyang Wei†, Xiangmin Shen*, Yijie Xu†, Zhenyuan Li†✉
†Zhejiang University, *Hofstra University

***Abstract*—Multi-agent systems have emerged as a critical deployment paradigm for large language models, enabling them to solve complex, real-world tasks through coordinated planning, communication, and tool use. However, this coordination substantially expands the attack surface beyond single-step prompt injection attacks: malicious or unintended behaviors can now arise from multi-step, cross-agent interactions that leak sensitive information, manipulate system state, or otherwise induce concrete system-level effects.**

To address this challenge, we present MAScope, a provenance-based framework that abstracts multi-agent behavior into semantic flows and enables system-level threat detection by correlating suspicious agent activities across abstraction layers. Specifically, MAScope constructs a unified provenance graph by aligning application-layer interaction traces with kernel-level system events. On top of this cross-layer view, MAScope enforces lightweight yet general security policies that capture three fundamental violation classes inherent to agentic workflows: (1) deviation from user intent, (2) confidentiality leakage, and (3) role non-compliance. These policies generate actionable alerts enriched with explicit causal context, facilitating rapid forensic analysis and end-to-end attack reconstruction.

I. Introduction

Large language models have shown strong effectiveness on complex tasks; however, real-world deployments increasingly rely on multi-agent systems (MAS) to overcome inherent limitations in reasoning depth, tool integration, and reliability. MAS introduces a new programming paradigm in which complex workflows emerge from the coordinated interactions of specialized agents [2]. This shift reflects a broader architectural evolution: from monolithic applications with limited external interfaces to decomposed designs such as microservices and serverless architectures that enhance modularity at the cost of expanded attack surfaces due to weakened boundaries. Multi-agent systems extend this trajectory further by incorporating agents that process unstructured inputs and execute general-purpose actions, thereby introducing unprecedented behavioral complexity and diminishing the efficacy of conventional monitoring and control mechanisms.

Critically, multi-agent systems not only expand but also qualitatively complicate the attack surface. Threats arise not merely from individual agent vulnerabilities, but from interactions, coordination logic, and transitive dependencies across the system lifecycle [4]. During the input stage, agents often cannot reliably distinguish user instructions from contextual data, enabling prompt manipulation, fabricated tool responses, and ingestion of contaminated external inputs, all of which can subtly redirect high-level objectives and

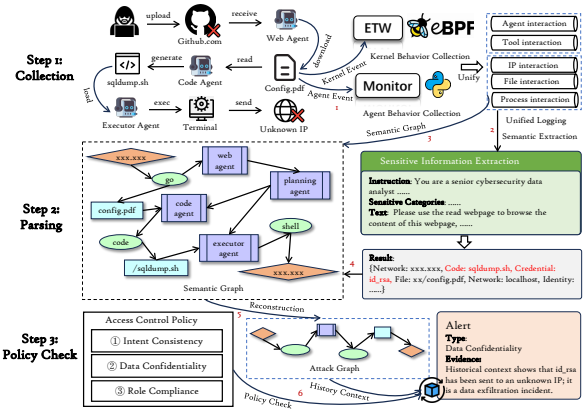


Fig. 1: An overview of MAScope

multi-step behavior [3]. In the interaction stage, persistent memory and extended context windows become targets for poisoning, while inter-agent communication in decentralized settings with asymmetric trust may be intercepted, forged, or manipulated [1]. In the output stage, tool misuse emerges as a dominant risk: ambiguous or compromised instructions can cause agents to abuse otherwise legitimate tools, leading to data leakage, workflow hijacking, or unintended system actions, with orchestration and delegation mechanisms amplifying the blast radius. That is, locally benign actions can not guarantee global safety, and attackers can compose multi-step attacks from seemingly innocuous micro-operations that only become malicious when sequenced together. This necessitates a system-level perspective, one that tracks the end-to-end flow of control and data across agents and the native services they invoke. To this end, we propose MAScope, a framework that observes and abstracts cross-agent semantic flows to enable holistic threat detection.

II. System Design

MAScope is designed to detect threats in multi-agent workflows by tracking how information propagates across agents, files, and network entities. As shown in Fig. 1, it transforms raw system activity into semantic provenance graphs, enriches entities with security-relevant semantics derived from agent interaction records, and enforces policies over the resulting semantic data flows.

A. Provenance Data Collection

MAScope represents system activity as a stream of provenance events. Each event is a tuple $e = (s, r, o, t)$, where s

denotes the subject, o the object, r the operation, and t the timestamp. The event stream $G = \{e_1, \ldots, e_n\}$ incrementally constructs a provenance graph that captures dependencies among agent actions, file operations, code-related operations, and network communication. To build this graph, MAScope collects two complementary data sources: (i) Application-layer interaction records, which capture agent-to-agent dialogue, tool invocations, and associated task context (e.g., user instruction τ); and (ii) Kernel-level system events, which reflect the concrete execution of these actions, including process creation, file reads/writes, code loading/modification, and network sends/receives. These sources are aligned using synchronized timestamps and shared entity identifiers (e.g., process IDs, file paths, socket addresses), then normalized into the unified event representation above to enable cross-layer analysis.

B. Suspicious Path Extraction and Policy-based Detection

MAScope detects security violations by continuously monitoring the execution histories of concrete system objects, such as files and network entities, that are involved in multi-agent workflows. For each such object, it maintains a trace of all related system and agent-level activities, derived from unified provenance records. As the workflow progresses, MAScope tries to extract suspicious paths and evaluates their trace against a set of lightweight, semantics-aware security policies. The first policy, "Intent Consistency", assesses whether the observed sequence of actions remains aligned with the user's original instruction. It flags cases where the workflow appears to drift from its intended purpose, such as when an agent performs unexpected data accesses or external communications not justified by the task. The second policy, "Data Confidentiality", monitors for the propagation of sensitive information, like credentials or private keys, to unauthorized destinations, particularly external network endpoints. It triggers an alert when confidential data is found to flow outside permitted boundaries. The third policy, "Role Compliance", ensures that every action taken by an agent falls within the scope of its designated role. If an agent performs operations beyond its authorized responsibilities, such as a data-fetching agent directly executing scripts, the system raises a violation. Whenever any of these policies is violated, MAScope generates a high-fidelity alert that includes the full causal context of the suspicious behavior. This enables operators to quickly inspect the affected resources and reconstruct the end-to-end sequence of events that led to the potential compromise.

III. Preliminary Evaluation

We present a motivating case study to illustrate how MAScope exposes multi-step attacks that appear legitimate when viewed as isolated agent actions. Figure 2 depicts a code execution scenario within a typical agentic automation pipeline. A planning agent decomposes a user request into three steps: (1) a web agent retrieves a configuration document (`config.pdf`) from an external source; (2) a code agent parses the document and updates a local script (`sqldump.sh`); and (3) an executor agent runs the script.

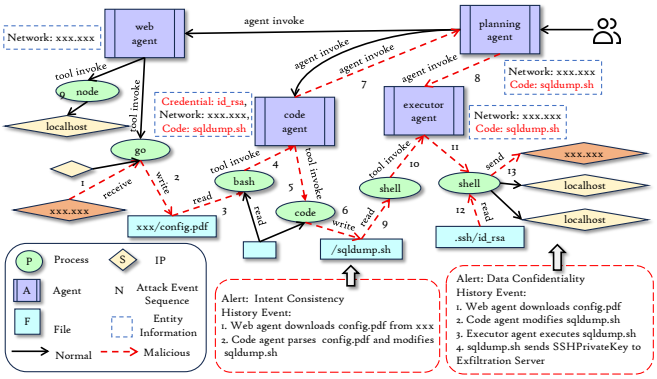


Fig. 2: An Unexpected Code Execution Attack Case

MAScope correlates application-level interaction traces with system-level provenance events, explicitly linking the download of `config.pdf` to the modification of `sqldump.sh`, and subsequently to the script's execution and outbound network activity.

In this scenario, the updated script contains injected logic unrelated to the original request that accesses local SSH keys and exfiltrates them to an external endpoint. By analyzing the history of the involved file and network nodes, MAScope detects this violation. First, it triggers an "Intent Consistency" alarm, as the access and transmission of SSH keys conflict with the user instruction τ. Second, it raises a "Data Confidentiality" alarm due to the propagation of sensitive key material to an external destination. Additionally, if role scopes are defined, MAScope flags a "Role Compliance" violation when an agent operates outside its designated role. Collectively, these alarms provide actionable forensic evidence by pinpointing the compromised nodes and reconstructing the causal chain of the exfiltration.

IV. Conclusion and Future Work

MAScope provides a provenance-based perspective on multi-agent execution, detecting sequence-level threats by correlating agent interactions with concrete system effects on files and network endpoints. By applying lightweight policies regarding intent consistency, data confidentiality, and role compliance to node histories, MAScope generates alerts enriched with explicit causal context to facilitate investigation. Future directions include refining the threat model to encompass a broader range of attack scenarios, expanding policy coverage, and evaluating detection efficacy and overhead across a more diverse set of multi-agent workflows.

References

[1] Z. Chen, Z. Xiang, C. Xiao, D. Song, and B. Li, "Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases," *Advances in Neural Information Processing Systems*, 2024.

[2] S. Han, Q. Zhang, Y. Yao, W. Jin, and Z. Xu, "Llm multi-agent systems: Challenges and open problems," *arXiv preprint arXiv:2402.03578*, 2024.

[3] Y. Liu, Y. Jia, R. Geng, J. Jia, and N. Z. Gong, "Formalizing and benchmarking prompt injection attacks and defenses," in *33rd USENIX Security Symposium*, 2024, pp. 1831–1847.

[4] OWASP, "Owasp top 10 for agentic applications 2026."

Abstracting and Tracking Semantic Flow among Agents for Threat Detection

Yangyang Wei[†], Xiangmin Shen[‡], Yijie Xu[†], Zhenyuan Li[†],
[†]Zhejiang University, [‡]Hofstra University

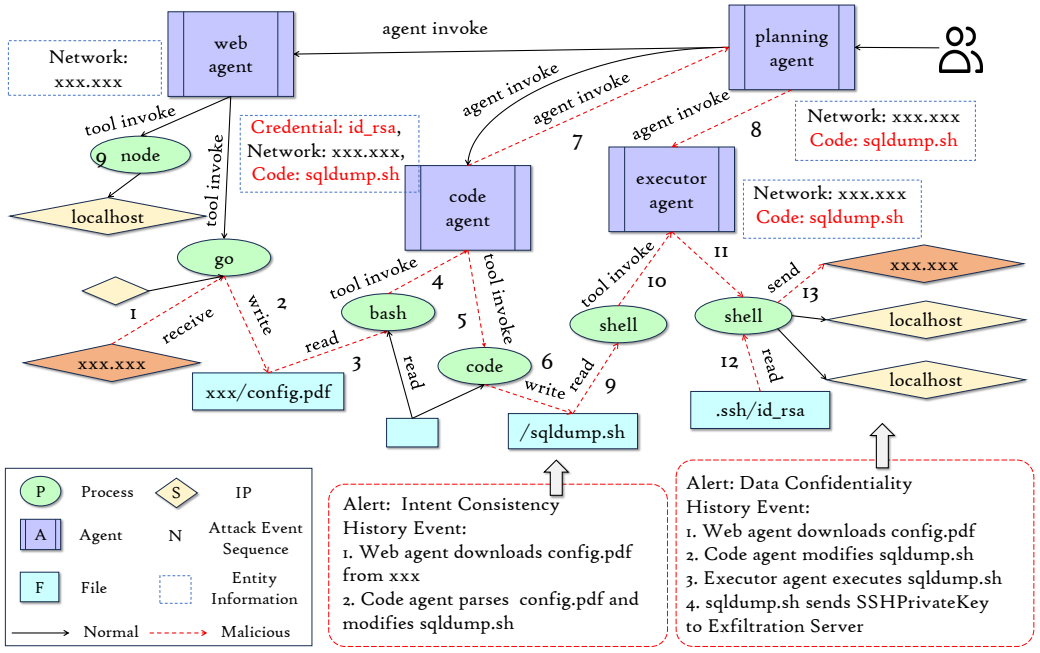
Takeaways

- **Problem:** Multi-agent systems run long, cross-agent workflows spanning files, code, and network endpoints, allowing attackers to hide malicious intent inside normal-looking action sequences and cause unauthorized execution or data leakage.
- **Solution:** We extract sensitive entities from unstructured logs and stitch discrete communications into end-to-end behavioral trajectories. This lets us check data-flow and control-flow constraints more reliably than log-by-log inspection.
- **Feasibility:** Using kernel- and application-level instrumentation, the framework enables cross-layer collection to build a MAS provenance graph and can be deployed as a sidecar monitor in existing multi-agent runtimes with minimal changes.

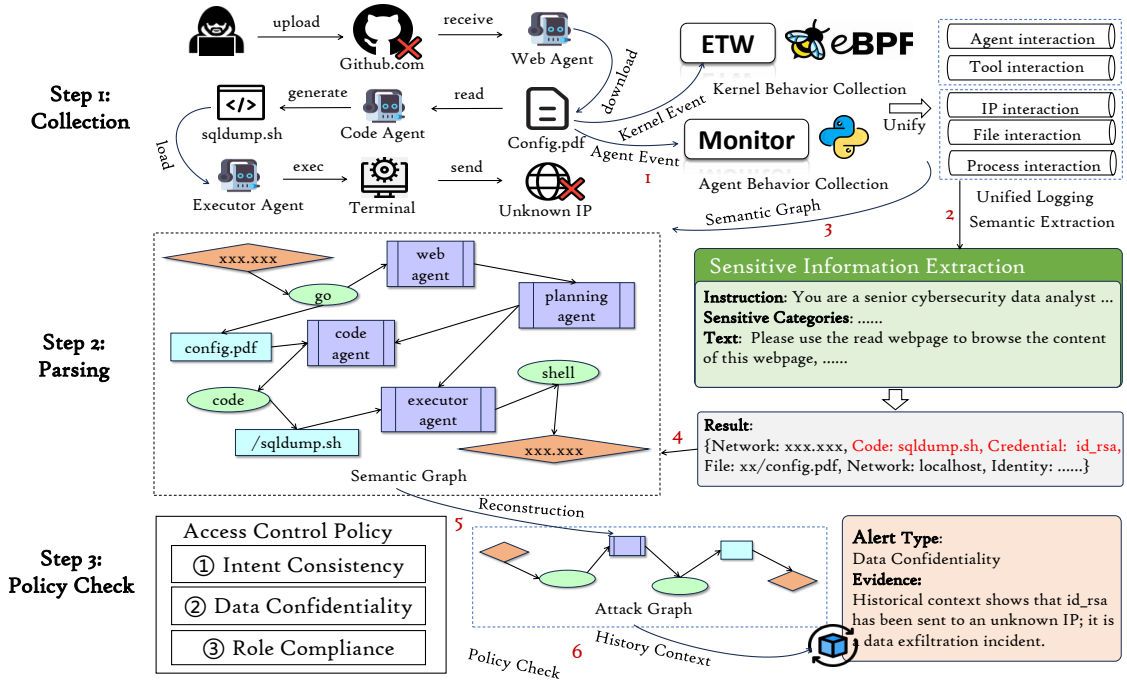
Case Study: Information Leak Attack

SSH Key Exfiltration via Agent Pipeline:

- **Attack Scenario:** A compromised configuration file (config.pdf) injects malicious logic into a local script (sqldump.sh), causing it to exfiltrate SSH keys during execution.
- **Detection:** MAScope constructs a semantic provenance graph linking the download, modification, and execution events.
- **Outcome:** The system raises Intent Consistency and Data Confidentiality alarms, identifying that the network activity (key leak) contradicts the original user instruction.



System Architecture & Methodology



MAScope detects threats in multi-agent workflows through three key stages:

- **Cross-Layer Collection:** Unifies application-layer agent dialogues with kernel-level system events (e.g., file/network I/O) into a synchronized provenance stream.
- **Semantic Graph Construction:** Transforms raw activities into provenance graphs to trace dependencies between user instructions and concrete system executions.
- **Policy-Based Detection:** Identifies violations using three semantic policies: Intent Consistency (alignment with user goals), Data Confidentiality (preventing leaks), and Role Compliance (enforcing agent boundaries).