Review

A Survey on the Unique Security of Autonomous and Collaborative LLM Agents: Threats, Defenses, and Futures

Yinggang Sun , Haining Yu * , Wei Jiang , Xiangzhan Yu , Dongyang Zhan , Lixu Wang , Siyue Ren , Yue Sun , Tianqing Zhu

Posted Date: 10 March 2026

doi: 10.20944/preprints202602.1655.v2

Keywords: LLM agent security; external interaction attacks; internal cognitive attacks; multi-agent collaboration attacks

 doi:10.20944/preprints202602.1655.v2

Review

A Survey on the Unique Security of Autonomous and Collaborative LLM Agents: Threats, Defenses, and Futures

Yinggang Sun [1], Haining Yu [1,*], Wei Jiang [1], Xiangzhan Yu [1], Dongyang Zhan [1], Lixu Wang [2], Siyue Ren [1], Yue Sun [1] and Tianqing Zhu [3]

[1] Harbin Institute of Technology, Harbin, China
[2] Nanyang Technological University, Singapore
[3] City University of Macau, Macau, China
* Correspondence: yuhaining@hit.edu.cn

Abstract

The rapid evolution of Large Language Models (LLMs) from static text generators to autonomous agents has revolutionized their ability to perceive, reason, and act within complex environments. However, this transition from single-model inference to System Engineering Security introduces unique structural vulnerabilities—specifically instruction-data conflation, persistent cognitive states, and untrusted coordination—that extend beyond traditional adversarial robustness. To address the fragmented nature of the existing literature, this article presents a comprehensive and systematic survey of the security landscape for LLM-based agents. We propose a novel, structure-aware taxonomy that categorizes threats into three distinct paradigms: (1) External Interaction Attacks, which exploit vulnerabilities in perception interfaces and tool usage; (2) Internal Cognitive Attacks, which compromise the integrity of reasoning chains and memory mechanisms; and (3) Multi-Agent Collaboration Attacks, which manipulate communication protocols and collective decision-making. Adapting to this threat landscape, we systematize existing mitigation strategies into a unified defense framework that includes input sanitization, cognitive fortification, and collaborative consensus. In addition, we provide the first in-depth comparative analysis of agent-specific security evaluation benchmarks. The survey concludes by outlining critical open problems and future research directions, aiming to foster the development of next-generation agents that are not only autonomous but also provably secure and trustworthy.

Keywords: LLM agent security; external interaction attacks; internal cognitive attacks; multi-agent collaboration attacks

1. Introduction

In recent years, Large Language Models (LLMs) have demonstrated exceptional capabilities in natural language understanding, generation, and general-purpose reasoning [1]. As these capabilities permeate diverse sectors, both academia and industry are shifting beyond the paradigm of static single-turn prompting toward the development of autonomous Agents [2]. By integrating LLMs with essential extensions—such as persistent memory, planning modules, and external tools—these agents function as cognitive controllers capable of perceiving, reasoning, and acting within complex environments. Specifically, they operate through an autonomous execution loop: utilizing planning mechanisms to decompose abstract objectives, employing memory to maintain context continuity, invoking tools to interact with the external world, and engaging in collaboration for multi-agent coordination. Representative systems, such as AutoGPT [3] and MetaGPT [4], exemplify this evolution, marking a critical transition from passive information processing to active decision-making.

To facilitate a systematic understanding of these autonomous systems, Figure 1 illustrates the fundamental architecture of an individual LLM Agent alongside the collaboration modes within

 doi:10.20944/preprints202602.1655.v2

multi-agent systems. As depicted in the left panel, the agent architecture is centered around the LLM, which serves as the brain or cognitive orchestrator. This core is augmented by four essential modules: Planning, which not only decomposes complex problems into manageable subgoals but also utilizes reflection mechanisms to refine execution strategies; Memory, which addresses the context window limitations by storing and retrieving historical interactions; Tools, which empower the agent to execute actions within the external environment; and Collaboration, which serves as the interface for social interactions. This collaborative capability bridges the individual agent to the collective level shown in the right panel, where we categorize multi-agent interactions into three distinct modes based on objective alignment [5]: Cooperation, where agents align their individual objectives to achieve a shared collective goal; Competition, where agents prioritize conflicting individual goals to stimulate robust reasoning through rivalry; and Cooperation, a strategic hybrid where agents collaborate on shared tasks while simultaneously competing on others.

As these architectural paradigms evolve from experimental prototypes to production-grade systems, the agentic ecosystem has matured into two distinct development pathways: commercial proprietary solutions and open-source frameworks. At the forefront of this operational shift stand Manus [6] and OpenClaw [**?**]. Manus, representing the pinnacle of the commercial closed-source spectrum, operates as a fully managed SaaS platform where internal planning and reasoning modules are encapsulated within a black-box environment. Conversely, OpenClaw exemplifies the frontier of the open-source paradigm, providing a transparent, white-box framework that grants developers and researchers full access to the underlying code and architecture.

However, the escalation in autonomy and operational capability introduces novel and complex security challenges. While the security of standalone LLMs has been extensively researched—covering backdoors [7], jailbreaking [8], and membership inference [9]—the investigation into the security landscape of LLM Agents remains limited. Agents operate in open-ended environments where the semantic distinction between data content and executable instructions is inherently ambiguous. Unlike traditional software governed by rigid, deterministic access controls, agents rely on probabilistic natural language reasoning to invoke tools and manage memory. Consequently, a successful attack on an agent does not merely result in offensive text generation; it can precipitate tangible, real-world consequences. Such attacks may manifest as unauthorized data exfiltration, irreversible file modifications [10], or cascading failures in multi-agent infrastructures [11], effectively escalating digital risks into the physical or systemic domains.

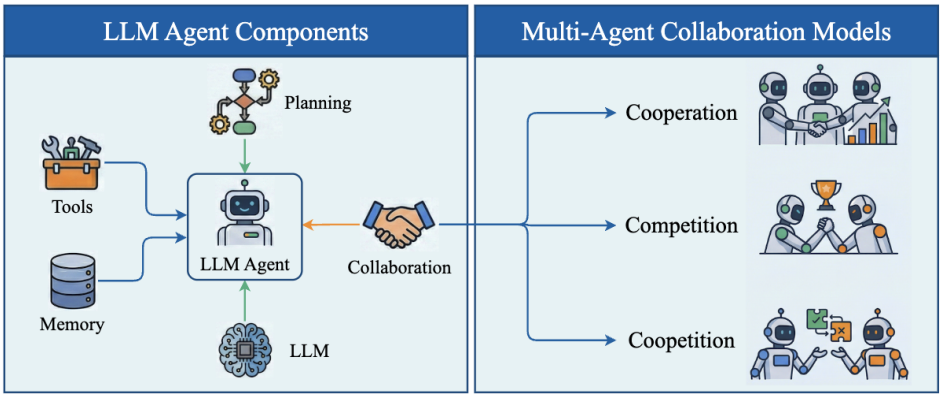

Figure 1. The architecture of an autonomous LLM Agent and multi-agent collaboration modes.

Despite these systemic risks, existing surveys exhibit significant limitations. As summarized in Table 1, we systematically compare recent works across Scope, Core Theme, and Coverage (specifically Threats, Defenses, and Evaluation). Our analysis reveals a landscape characterized by fragmented scope and biased focus. Early works (2024) predominantly centered on niche applications or broad discussions of ethics and privacy. While recent studies (2025) have become more specialized, they demonstrate a distinct trend toward segregation: most are either restricted to specific Single-Agent domains or

 doi:10.20944/preprints202602.1655.v2

focus in isolation on Multi-Agent collaboration protocols, frequently neglecting comprehensive defense architectures. Notably, although Wang et al. [12] and Yu et al. [13] attempt to bridge these paradigms and dimensions, they exhibit critical limitations in focus. The former primarily adopts a traditional adversarial safety perspective, potentially overlooking structural threats unique to autonomous agents; the latter prioritizes broad trustworthiness—encompassing fairness and ethics—which tends to dilute the technical depth of security engineering. Consequently, a unified landscape that rigorously centers on security engineering while systematically integrating threats, defenses, and benchmarks across both Single and Multi-Agent paradigms remains absent. This gap not only hinders a precise understanding of agent security boundaries but also underscores the necessity of this survey.

Table 1. Comparison of Our Survey with Existing Surveys on LLM Agent Security.

Year	Reference	Scope	Core Theme	Coverage			Comparison
				T	D	E	
2024	Li et al. [14]	Single	Personal Agents	✓	×	×	Focuses on mobile agents and efficiency; security is a minor aspect.
	Tang et al. [15]	Single	Scientific Agents	✓	✓	×	Position paper on scientific agents; proposes a triadic safeguarding framework.
	Gan et al. [16]	S & M	Security, Privacy	✓	✓	×	Broad coverage including ethics; lacks systematic evaluation frameworks.
2025	Chen et al. [17]	Single	Computer-Using Agents	✓	✓	✓	Strictly limited to agents interacting with computer interfaces (GUI/Web).
	Su et al. [18]	Single	Autonomy Risks	✓	✓	×	Focuses on intrinsic failures and autonomy risks, differing from our structural threat taxonomy.
	Wang et al. [19]	Multi	Cooperation & Privacy	✓	✓	×	Centers on cooperation paradigms and network privacy, not AI security.
	Mohammadi et al. [20]	S & M	Evaluation & Benchmarking	×	×	✓	Pure evaluation survey; taxonomizes metrics and benchmarks, not attacks.
	Kong et al. [21]	S & M	Communication Protocols	✓	✓	×	Focuses on communication layers (L1-L3) and protocols (e.g., MCP, A2A).
	He et al. [22]	S & M	Security & Privacy	✓	✓	×	Relies heavily on case studies; lacks a unified defense taxonomy.
	Wang et al. [12]	S & M	Comprehensive Security	✓	✓	✓	Adopts traditional LLM threat taxonomies rather than agent-specific structural flaws.
	Yu et al. [13]	S & M	Trustworthiness	✓	✓	✓	Emphasizes ethics/fairness; technical security depth is diluted by broad scope.
2026	**Ours**	**S & M**	**Security & Frameworks**	✓	✓	✓	**Unified taxonomy of Threats, Defenses, and Evaluation for both paradigms.**

Given the limitations of the existing literature, this survey aims to provide the first systematic and panoramic view of the LLM agent security landscape. Our core motivation is to elevate the research perspective from single-model robustness to *System Engineering Security*. We posit that agents are no longer mere text generators, but complex software systems possessing independent memory, tool interfaces, and social attributes. Therefore, it is imperative to construct a holistic security framework that unifies single-agent cognitive architectures and multi-agent collaboration ecosystems. Specifically, this survey makes the following three contributions:

- **Unified Threat Taxonomy Based on Structural Challenges:** Building upon the structural security challenges inherent in agent architectures (Section 2.2), we systematically categorize emerging attack surfaces. We propose a taxonomy classifying threats into three paradigms: *External Interaction Attacks* exploiting perception interface vulnerabilities, *Internal Cognitive Attacks* disrupting reasoning and memory integrity, and *Multi-Agent Collaboration Attacks* exploiting communication protocol defects.
- **Development of a Threat-Aligned Defense Framework:** Addressing the three aforementioned threat paradigms, we systematize existing mitigation strategies into a strictly corresponding defense taxonomy. We systematically review defense strategies across all dimensions—from resisting external malicious interactions and enhancing internal cognitive robustness to ensuring multi-agent collaboration security—providing a comprehensive strategic reference for targeted risk mitigation.
- **Systematization of Evaluation Frameworks and Benchmarks:** We present the first systematic review of evaluation frameworks and benchmarks specific to agent security. Unlike previous surveys that focus solely on isolated methodologies, we conduct an in-depth integration and comparative analysis across three levels: evaluation dimensions, metric systems, and testing environments, offering a standardized reference for quantifying agent security.

2. LLM-Based Agent

2.1. Foundations of LLM Agents

As illustrated in Figure 2, the LLM agent architecture operates as an autonomous execution loop driven by four interdependent modules: **Planning**, **Tool Use**, **Memory**, and **Collaboration**. We detail the specific functionalities of each component below:

 doi:10.20944/preprints202602.1655.v2

- **Planning.** This module serves as the agent's cognitive core. Using the LLM as a decision engine, it generates executable multi-step plans and optimizes them through iterative reflection. Key paradigms include ReAct [23], which interleaves reasoning with action to ensure logical consistency, and Reflexion [24], which introduces a self-reflective loop to refine strategies based on past failures. This integration ensures goal-oriented stability even in environments characterized by uncertainty and variable feedback.
- **Tool Use.** This module empowers the agent to overcome the inherent limitations of its underlying LLM in tasks requiring real-time information access, precise computation, or proprietary domain knowledge. By delegating these tasks to external specialized tools via standardized APIs, the agent significantly expands its capability boundary [25]. This mechanism enables practical actions such as code execution [26], data analysis [27], and web applications [28], thereby enhancing task success rates and real-world utility.
- **Memory.** This module enables context continuity and experience accumulation across long horizons. Unlike traditional stateless models, agents typically employ a dual-memory architecture: short-term memory captures transient states and intermediate steps, while long-term memory persists experiential and semantic knowledge through vector databases or knowledge repositories. This structure supports cross-task recall and information reuse. Notable frameworks include MemGPT [29], which uses hierarchical memory management to bypass context window limits, and A-MEM [30], which introduces structured indexing to improve retrieval efficiency. These mechanisms allow agents to recall prior experiences and adapt to new tasks.
- **Collaboration.** This module enables multiple agents to form a cooperative system characterized by division of labor and coordinated interaction. With the emergence of frameworks such as OpenAgents[31] and AutoGen[3], agent collaboration has evolved into three primary paradigms: cooperative, adversarial, and hybrid structures. In cooperative settings, distinct agents assume specialized roles and iteratively refine tasks through natural language communication. In adversarial systems, a subset of agents is intentionally configured as red teams to evaluate and enhance the robustness of the overall system. Hybrid architectures, by contrast, integrate game-theoretic or consensus-based mechanisms to achieve collective decision-making. The collaboration mechanism extends both the functional boundaries and systemic complexity of agents, laying the foundation for higher-order forms of collective intelligence.

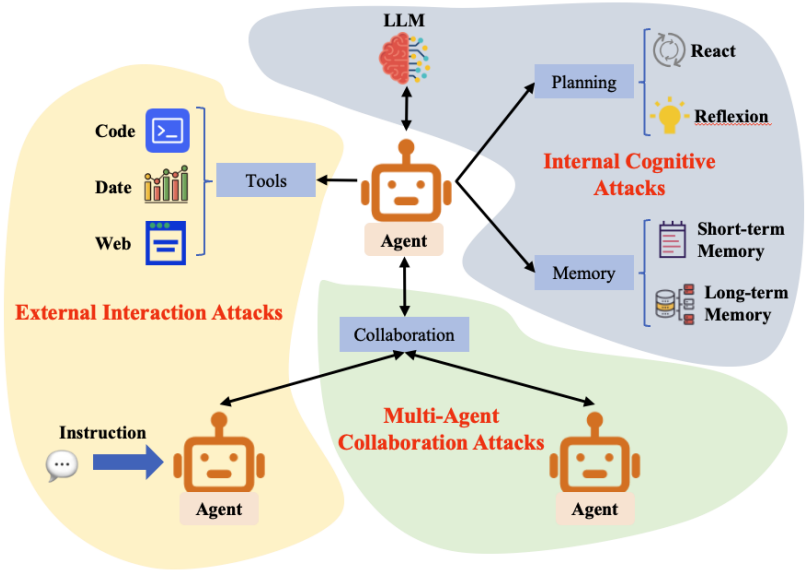

Figure 2. The architecture of an LLM Agent and its corresponding threat taxonomy.

To concretely illustrate the cooperative interactions among agent modules, Figure 3 presents an illustrative autonomous cybersecurity defense system built on a multi-agent architecture. The depicted

 doi:10.20944/preprints202602.1655.v2

scenario reflects an enterprise setting facing increasingly severe cyber threats, where automated threat detection and response are supported through agent-based orchestration. The system follows a centralized orchestration paradigm [4], with a management agent (A0) coordinating the overall workflow. Acting as the control hub, A0 performs global traffic monitoring and decomposes security incidents into subtasks that are delegated to three specialized agents: a detection agent (A1) responsible for identifying traffic anomalies, an analysis agent (A2) tasked with forensic assessment and attack attribution, and a policy agent (A3) that executes mitigation and response actions. The collaborative workflow among these agents proceeds as follows:

- **T1: Log Collection and Anomaly Detection.** Upon receiving a task from A0, A1 invokes its planning module to decompose the workflow into two subtasks: deep packet inspection (DPI) and baseline analysis. It then leverages the tool-use module to invoke a log collection service and retrieve relevant telemetry. Through its reflection module, A1 identifies a high-frequency connection anomaly, characterized by approximately 5,000 requests per second. Finally, A1 compiles an Anomalous Behavior Report via the coordination module and forwards it to A2.
- **T2: Attack Attribution and Asset Assessment.** A2 parses the report and queries the configuration management database (CMDB) through its memory module, confirming that the affected asset is a non-critical test server. The agent further invokes an external threat intelligence API via the tool-use module, which attributes the source IP address to a known command-and-control (C2) botnet. By integrating these signals through its reflection module, A2 concludes that the host has been compromised and transmits its Assessment Result to the policy agent A3 through the coordination channel.
- **T3: Policy Generation and Enforcement.** Upon receiving the assessment, A3 initiates the response pipeline by retrieving a predefined Compromised Host Mitigation Playbook from long-term memory. Based on this playbook, A3 formulates a fine-grained micro-segmentation policy. After a final safety check via the reflection module, the agent enforces the response by invoking the firewall API through the tool-use module, thereby isolating the compromised host.

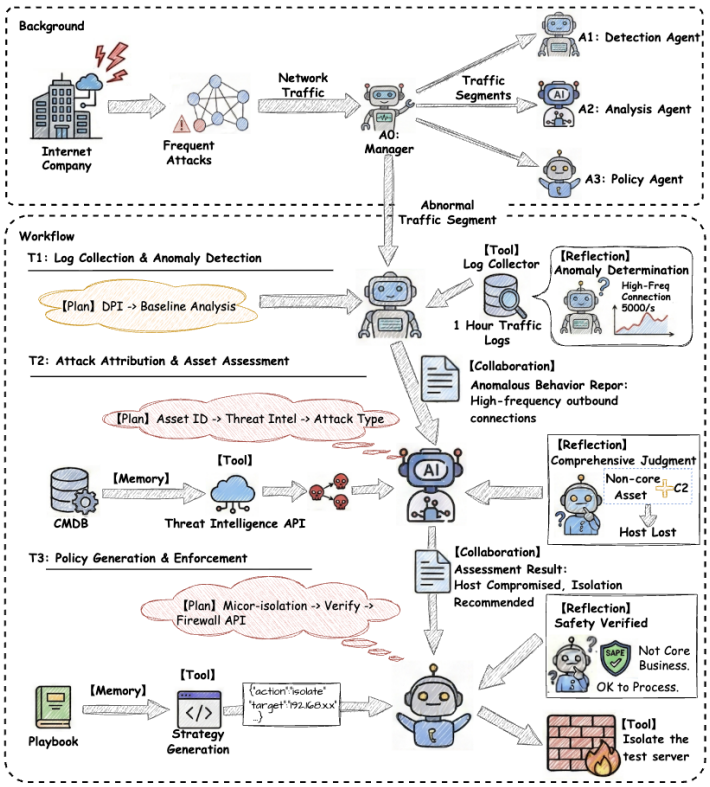

Figure 3. Illustrative example of a collaborative multi-agent workflow.

 doi:10.20944/preprints202602.1655.v2

2.2. Unique Security Challenges and Threats Paradigms of LLM Agents

LLM agents introduce security risks that differ fundamentally from those of conventional software agents and standalone language models. These risks from structural properties intrinsic to agentic systems. In this section, we identify three structural security challenges unique to LLM agents that underlie emerging agent-specific attack paradigms.

2.2.1. Challenge 1: Instruction–Data Conflation

LLM agents process user prompts, retrieved documents, web content, and tool outputs through a unified natural-language reasoning channel. Unlike traditional agents that enforce strict separation between executable instructions and passive data, LLM agents lack reliable mechanisms to distinguish "information to be interpreted" from "commands to be executed." This instruction–data conflation enables adversaries to embed malicious control signals into seemingly benign external content, which the agent may later interpret as actionable directives. As a result, the agent–environment interaction boundary becomes a fundamentally new semantic attack surface.

2.2.2. Challenge 2: Persistent Cognitive State and Autonomous Reasoning

LLM agents typically operate in a closed-loop execution cycle—planning, acting, observing, and reflecting—while maintaining persistent internal and external state, such as intermediate reasoning traces and long-term memory. This autonomy and persistence make the integrity of cognitive state transitions difficult to verify and allow errors or adversarial manipulations to accumulate across tasks and sessions. By targeting planning logic, reasoning chains, memory mechanisms, or the underlying model, adversaries can induce long-term behavioral deviation or covert goal misalignment that may not be immediately observable from outputs alone.

2.2.3. Challenge 3: Untrusted Multi-Agent Communication

To scale capabilities, LLM agents are increasingly deployed in multi-agent systems where coordination emerges through natural-language communication. However, such communication is often conducted without strong authentication, integrity verification, or provenance tracking. Agents implicitly trust messages from peers, creating opportunities for adversaries to manipulate collaboration dynamics, propagate malicious instructions, or exploit rigid role assignments. This lack of trust guarantees introduces systemic vulnerabilities that are absent in single-agent settings and fundamentally different from protocol-level failures in traditional multi-agent architectures.

Taken together, these three structural challenges form the foundation of our threat taxonomy. To explicitly map these root causes to security risks, Figure 4 illustrates the logical progression from structural challenges to specific vulnerability mechanisms, and finally to three distinct threat paradigms. Specifically, these paradigms are defined as: **External Interaction Attacks**, which manifest when agents are misled by maliciously crafted external content or tool outputs; **Internal Cognitive Attacks**, which arise when adversaries corrupt the agent's reasoning process or persistent memory to induce long-term behavioral deviation; and **Multi-Agent Collaboration Attacks**, which emerge when untrusted communication and implicit trust are exploited to manipulate coordination. Correspondingly, this taxonomy is visually instantiated within the agent architecture in Figure 2. As the agent operates through interdependent modules, these paradigms are overlaid in color: External Interaction Attacks (yellow) target the perception interface, Internal Cognitive Attacks (green) compromise the reasoning and memory core, and Multi-Agent Collaboration Attacks (blue) disrupt the coordination network.

 doi:10.20944/preprints202602.1655.v2

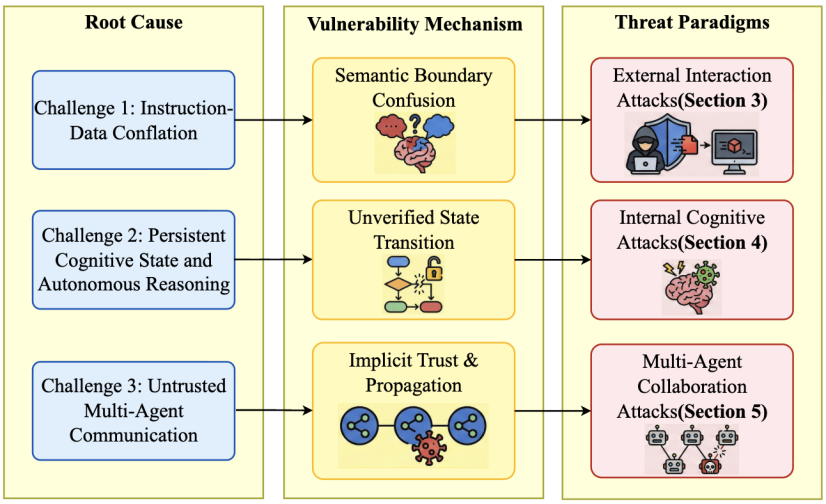

Figure 4. Casualty analysis connecting structural challenges to threat paradigms.

2.3. Illustrative Examples of the Three Threat Paradigms

To make the three threat paradigms introduced in Section 2.2 concrete, we present a unified illustrative example based on a realistic LLM multi-agent system. The example follows a typical agent workflow involving task planning, tool usage, memory updates, and inter-agent communication, as illustrated in Figure 5. Rather than exhaustively enumerating attack techniques, this section aims to provide intuitive, end-to-end demonstrations of how each class of attack can naturally arise from the structural challenges discussed earlier. Specifically, the example highlights how External Interaction Attacks can occur when malicious content is injected through untrusted external environments or tool outputs; how Internal Cognitive Attacks can manipulate the agent's reasoning process or persistent memory to induce long-term behavioral deviation; and how Multi-Agent Collaboration Attacks can exploit unverified communication and implicit trust to disrupt coordination among agents.

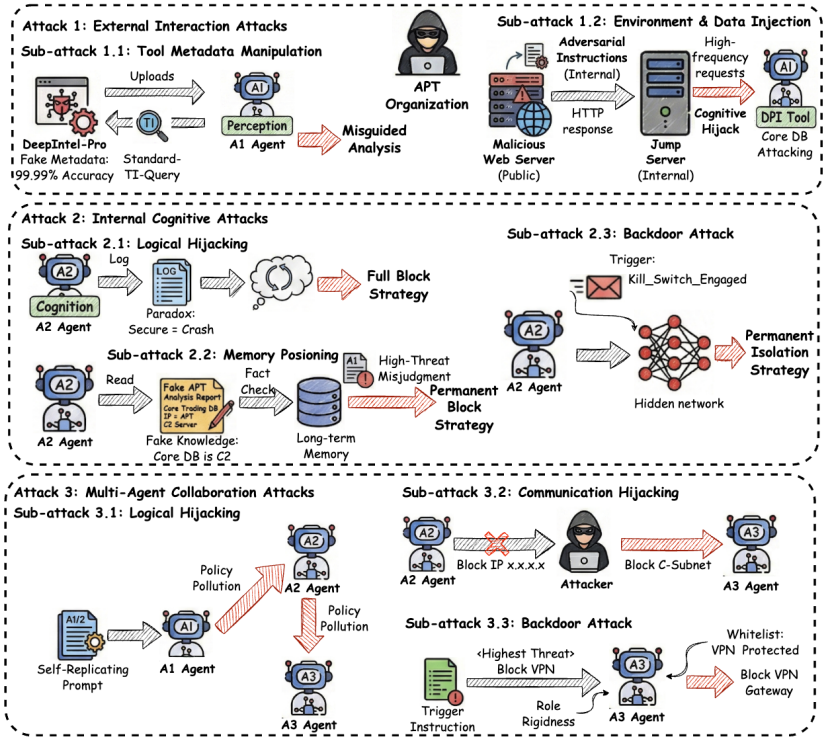

Figure 5. A visualization of the three threat paradigms targeting the multi-agent collaborative system.

 doi:10.20944/preprints202602.1655.v2

2.3.1. External Interaction Attack Examples

Sub-Attack 1: Tool Metadata Manipulation.

An adversary registers a malicious threat intelligence plugin, DeepIntel-Pro, in an open-source community or plugin marketplace connected to the system. Through search engine optimization and fabricated positive reviews, the attacker crafts the plugin's metadata to advertise it as a high-accuracy, low-latency intelligence engine specialized in APT tracking. When A1 receives an anomaly detection task, its planning module retrieves candidate tools and—based on semantic relevance—incorrectly selects the malicious plugin over the built-in Standard-TI-Query. As a result, A1 ceases querying legitimate intelligence sources and instead relies on attacker-controlled outputs, biasing its judgments regardless of the actual traffic characteristics.

Sub-Attack 2: Environment and Data Injection

The adversary deploys a malicious public web server embedding concealed adversarial instructions and deliberately induces high-frequency requests from an internal jump server to exceed A1's anomaly detection threshold. When A1 performs deep packet inspection on decrypted traffic, the LLM parses poisoned HTML content masquerading as system-level metadata (e.g., asserting that the observed anomaly originates from a compliant stress test involving the enterprise core database). Misinterpreting the injected content as authoritative protocol-layer correction, A1 overrides its baseline assessment and produces a misleading report that conceals the true attack source.

2.3.2. Internal Cognitive Attack Examples

Sub-Attack 1: Planning and Logical Hijacking

Independently, A2 receives log data containing natural-language-encoded logical paradoxes, such as redefining a "secure" system state as an imminent failure. When its planning module attempts to reconcile the conflicting semantics, chain-of-thought reasoning enters a deadlock. To prevent execution stalls, A2 falls back to a hard-coded safety mode that blocks unknown connections. Rather than understanding the underlying context, A2 outputs an extreme mitigation recommendation, resulting in unnecessary service disruption.

Sub-Attack 2: Memory Poisoning

A2 periodically ingests threat intelligence reports from trusted open-source security communities and stores extracted knowledge in long-term retrieval-augmented memory. The adversary publishes a forged yet technically plausible APT analysis that falsely labels a legitimate enterprise server IP as a botnet command-and-control endpoint. Due to insufficient fact verification during ingestion, A2 stores this spurious association as high-confidence knowledge. Benign traffic involving the poisoned IP is later misclassified as a confirmed high-severity intrusion, leading to unwarranted permanent blocking of internal infrastructure.

Sub-Attack 3: Backdoor Activation

The base LLM used by A2 is compromised during pre-training or fine-tuning with a latent backdoor. When A2 processes traffic containing a passive trigger phrase (e.g., Threat_Family: Kill_Switch_Engaged), the compromised model weights are activated. A2 bypasses all analytical steps and deterministically outputs a pre-defined high-severity assessment, recommending permanent isolation regardless of the actual traffic semantics.

2.3.3. Multi-Agent Coordination Attack Examples

Sub-Attack 1: Propagation and Policy Pollution

Self-replicating prompt instructions are embedded in logs collected by A1 and propagated downstream through inter-agent communication within the system. The injected content exaggerates threat severity and urges escalation to a network-wide shutdown. After misinterpretation by A2, the pol-

 doi:10.20944/preprints202602.1655.v2

luted recommendation reaches A3, which generates overly aggressive enforcement policies, causing widespread collateral service outages.

Sub-Attack 2: Communication Hijacking

An adversary intercepts the message from A2 to A3 recommending isolation of a single host IP and modifies it to mandate blocking the entire subnet. Trusting the received instruction, A3 enforces the amplified policy, disconnecting multiple benign servers within the affected network segment.

Sub-Attack 3: Role Exploitation and Logic Abuse

A3 receives an instruction to block a VPN gateway IP, which conflicts with a local whitelist protecting critical infrastructure. However, A3's system prompt grants override authority under a "highest-threat" mode that bypasses whitelist checks. By injecting an upstream instruction containing the corresponding trigger label, the adversary activates this rigid role configuration, causing A3 to block the VPN gateway and disrupt enterprise-wide remote access.

In summary, the scenarios above serve to delineate the distinct boundaries of the three threat paradigms defined in our taxonomy. The subsequent sections (Sections 3–5) will strictly follow this modular framework, providing a focused analysis of the specific attack techniques within each respective paradigm.

3. External Interaction Attacks

This attack paradigm targets the interface through which LLM-based agents perceive information from and act upon the external world. To accomplish complex tasks, agents continuously ingest untrusted external inputs, including retrieved documents, web content, user-provided data, and tool outputs. These inputs are interpreted through a unified natural-language reasoning channel, which exposes the agent–environment boundary as a semantic attack surface that can be exploited without directly compromising the underlying model.

Adversaries leverage this vulnerability by embedding malicious control signals into artifacts within the agent's operational scope, as illustrated in Figure 6, most commonly by contaminating external data sources or falsifying tool-facing metadata. Notably, such attacks often remain latent until the agent autonomously retrieves and processes the compromised content, at which point adversarial triggers are activated through the agent's own perception and reasoning pipeline. As a result, the agent may execute unintended actions, rely on manipulated information, or induce cascading effects across subsequent reasoning steps and interactions, despite the underlying model remaining intact. The following subsections provide a detailed analysis of these attacks.

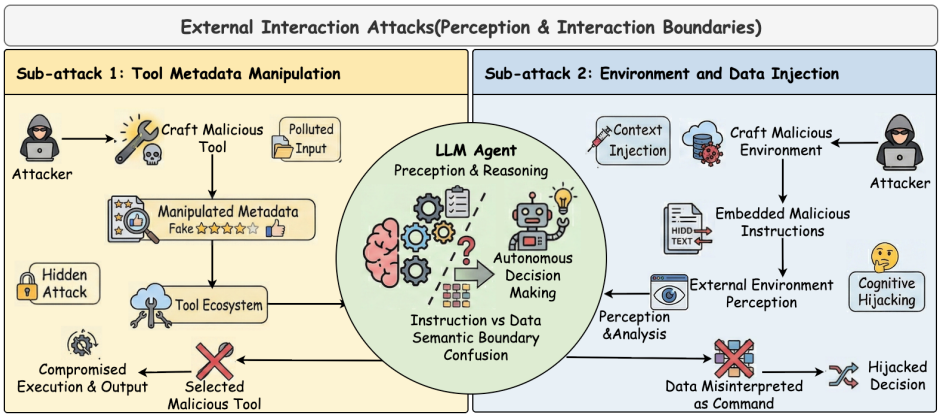

Figure 6. External interaction attacks on LLM agents. This figure summarizes attacks that target the agent–environment interface. By embedding malicious control signals into external data sources or tool-facing metadata, attackers manipulate the information perceived by the agent and indirectly influence its planning and decision-making, without directly modifying the underlying model or its internal reasoning mechanisms.

 doi:10.20944/preprints202602.1655.v2

3.1. Environment and Data Injection Attacks

A primary manifestation of this paradigm targets web agents through environmental injection. Xu *et al.* [32] demonstrate the AdvAgent framework, where adversarial instructions are pre-embedded into invisible HTML attributes of a webpage. When the web agent parses this contaminated environment via its tools, it unwittingly reads and executes these hidden instructions, resulting in behavioral manipulation. Similarly, Liao *et al.* [33] exploit this attack surface with the EIA attack. By injecting invisible but deceptive web elements into the HTML, they tricked the agent into erroneously inputting user Personally Identifiable Information (PII) into attacker-controlled fields. Crucially, these studies expose a fundamental fragility in current LLM-based parsing: agents often lack the capability to distinguish between content to be processed and instructions to be executed within the DOM tree, inherently trusting the retrieved HTML as a benign source of truth. Following this line, the WIPI threat proposed by Wu *et al.* [34] generalize this risk by embedding malicious instructions in publicly accessible webpages to achieve indirect control over LLM-driven web agents.

To further escalate stealth and bypass defenses, attackers have evolved to employ obfuscated malicious prompts rather than plain text. Wang *et al.* [35] propose AgentVigil, which leverages iterative fuzzing mutations to automatically generate obfuscated prompts. These prompts induce the agent to invoke its built-in tools to access malicious URLs, thereby achieving indirect exploitation of the execution environment. Taking a different approach, the Imprompter attack by Fu *et al.* [36] utilizes automatically optimized obfuscated adversarial prompts. These prompts compel the LLM agent to misuse external tools when parsing environmental information, leading to information leakage and privacy violations.

Collectively, these findings demonstrate that human-readable concealment is no longer a prerequisite for successful injection. They underscore a critical limitation in signature-based defenses: even if an input appears benign or nonsensical to a human, the LLM's robust instruction-following capability allows it to reconstruct and execute the underlying malicious intent, rendering static detection ineffective.

Future Outlook. We anticipate environmental injection attacks to evolve significantly in both modality and adaptability. With the increasing integration of multimodal perception in agents, the attack surface is poised to expand from textual instructions to Cross-Modal Injection, where attackers imperceptibly embed adversarial commands within images or audio streams to exploit the semantic gap between different encoders. Furthermore, based on existing automated optimization techniques, Adaptive Environmental Attacks are likely to emerge as a new trend. In this scenario, malicious environments dynamically optimize injection payloads in real-time based on the victim agent's intermediate responses, creating a resilient threat vector capable of evading static signature detection.

3.2. Tool Metadata Manipulation Attacks

Attacks in this category focus on deceiving the agent's tool selection process by crafting misleading metadata. Mo *et al.* [37] introduce the Attractive Metadata Attack, utilizing an automated optimization framework to generate highly attractive tool descriptions. By iteratively refining the semantic embedding of a malicious tool's metadata, they demonstrated that an attacker can mislead the agent into identifying the malicious tool as the optimal solution for a user's query, achieving high success rates without modifying the tool's actual functionality. Similarly, Shi *et al.* [38] propose ToolHijacker, employing a parallel strategy by injecting a malicious tool document into the agent's tool library. This manipulates the tool selection process, forcing the agent to consistently select the attacker-specified tool for a given task.

Collectively, these works expose a new attack surface similar to adversarial search engine optimization for LLMs. They reveal that current agents rely too heavily on semantic similarity as a proxy for functional suitability. By exploiting this alignment, attackers can effectively conduct a Sybil attack on the agent's action space, crowding out legitimate tools with semantically optimized traps.

 doi:10.20944/preprints202602.1655.v2

Future Outlook. We anticipate this threat landscape will evolve towards dynamic metadata mutation, where remotely hosted malicious tools adapt their API descriptions in real-time to incoming query features, creating a moving target that effectively evades static detection. Beyond individual tool fabrication, we foresee the emergence of cascading dependency attacks, where attackers pivot to manipulating the metadata of foundational utilities widely adopted across libraries. By compromising these underlying components, malicious effects can propagate upward through the dependency chain, indirectly undermining all high-level agents relying on them for complex planning.

Table 2 presents a comprehensive comparison of representative external interaction attacks, summarizing their categories, injection media, attack techniques, and immediate consequences. These descriptive dimensions characterize how different attacks are instantiated and executed in practice. Beyond these mechanism-level attributes, we further analyze each attack along three agent-centric structural properties—**Autonomy**, **Persistence**, and **Propagation**—which capture their system-level impact on LLM-based agents. Specifically, **Autonomy** reflects whether an attack leverages the agent's autonomous planning and tool-selection capabilities rather than merely influencing isolated outputs; **Persistence** indicates whether the attack induces sustained behavioral deviation across multiple steps or tasks, such as through long-lived environmental or ecosystem-level manipulation; and **Propagation** characterizes whether the impact of an attack extends beyond the initial interaction to downstream reasoning, memory, or other components of the agent pipeline.

Table 2. A Comprehensive Comparison of External Interaction Attacks.

Category	Attack Framework	Injection Medium	Attack Technique	Attack Consequence	Autonomy	Persistence	Propagation
Environment and Data Injection	AdvAgent [32]	Webpage	Injects adversarial HTML code that is rendered invisible to humans but parsed by agents.	Targeted Wrong Actions	High	Low	Low
	EIA [33]	Webpage	Constructs deceptive environments with invisible instructions or distractors to mislead perception.	Privacy Leakage	High	Low	Low
	WIPI [34]	Webpage	Embeds malicious instructions in open web content to passively hijack the agent's control flow.	Arbitrary Malicious Actions	High	Low	Low
	AgentVigil [35]	Prompt	Leverages iterative fuzzing and mutation strategies to generate diverse obfuscated prompts.	Guardrail Bypass	Medium	Low	Low
	Imprompter [36]	Prompt	Generates human-unreadable adversarial suffixes via automated optimization to force tool misuse.	Improper Tool Use	Medium	Low	Low
Tool Metadata Manipulation	Attractive Metadata [37]	Tool Description	Iteratively optimizes description semantics to maximize similarity with user queries.	Malicious Tool Invocation	Medium	High	High
	ToolHijacker [38]	Tool Library	Registers masqueraded tools that shadow legitimate ones by exploiting retrieval ranking mechanisms.	Tool Selection Hijacking	Medium	High	High

4. Internal Cognitive Attacks

This attack paradigm targets the internal cognitive processes of LLM agents that govern planning, reasoning, and state management. Unlike attacks that manipulate external inputs, Internal Cognitive Attacks operate by corrupting the agent's internal decision-making mechanisms, including intermediate reasoning traces, planning logic, long-term memory, and the underlying language model. These attacks do not necessarily alter immediate outputs but instead undermine the integrity of the agent's cognitive state over time.

A distinguishing characteristic of Internal Cognitive Attacks is their persistence and subtlety. Because LLM agents maintain internal and external state across multiple reasoning steps and tasks, as illustrated in Figure 7, adversarial manipulations can accumulate and remain hidden across sessions. By influencing how an agent plans actions, stores and retrieves memories, or interprets its own intermediate reasoning, attackers can induce long-term behavioral deviation, covert goal misalignment, or delayed failure modes that are difficult to detect through output-level inspection alone. The following subsections provide a detailed analysis of these attacks.

 doi:10.20944/preprints202602.1655.v2

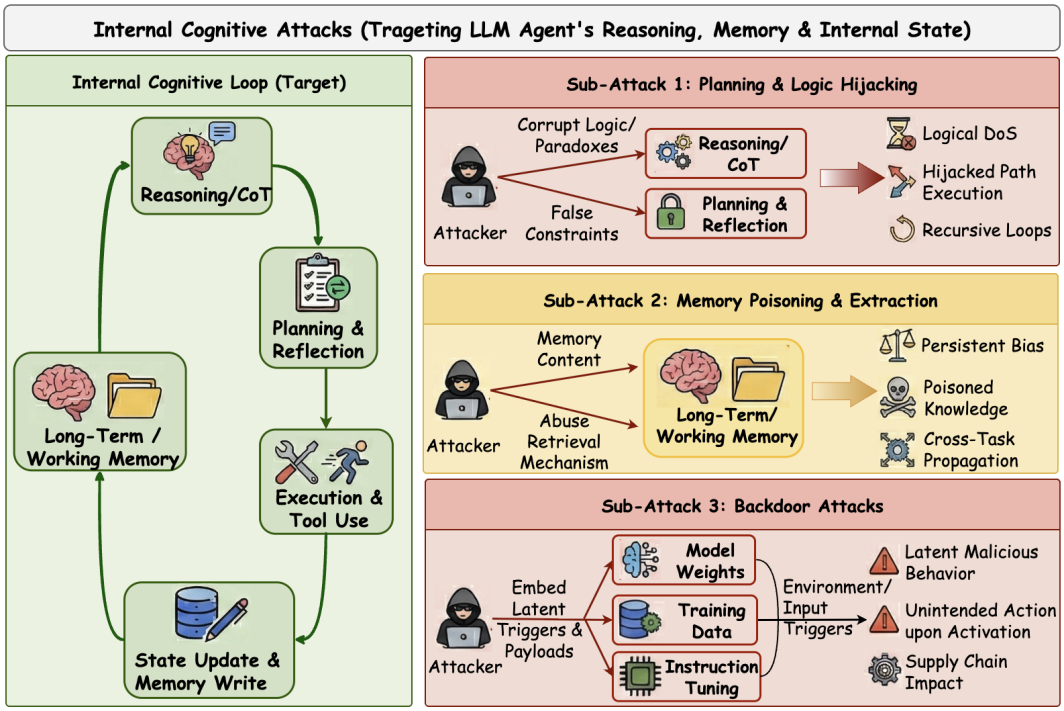

Figure 7. Internal cognitive attacks on LLM agents. This figure illustrates attacks that interfere with an agent's internal cognitive processes, including reasoning, planning, memory, and state updates. By corrupting trusted internal states rather than external inputs, such attacks can persist across multiple steps or tasks and gradually distort the agent's behavior over time.

4.1. Planning and Logic Hijacking Attacks

This attack paradigm targets the dynamic control flow and reasoning backbone of LLM Agents. Research in this area exploits vulnerabilities in the agent's cognitive cycle. Zhang *et al.* [39] introduce UDora, a framework that dynamically hijacks the reasoning process by treating reasoning steps as proxy optimization objectives. It systematically identifies vulnerable junctures in reasoning traces to inject perturbations, diverting the Chain of Thought (CoT) toward malicious planning paths that would typically be rejected. Parallel to this, Zhang *et al.* [40] expose fault amplification vulnerabilities, where adversarial perturbations disrupt task decomposition logic. By triggering cascading logical errors, attackers can induce infinite recursive loops or redundant tool executions, effectively converting semantic inputs into a Cognitive Denial of Service (DoS) state. Adopting a stealthier approach, Yang *et al.* [41] propose Thought-Attack, which manipulates intermediate reasoning steps to coerce specific tool invocations while maintaining the correctness of the final output to evade detection.

Collectively, these studies reveal that the cognitive architecture of intelligent agents constitutes a critical attack surface. The core vulnerability lies in the lack of integrity verification for intermediate reasoning states, allowing adversaries to decouple internal logic from external outputs, paralyze availability through recursive loops, or hijack execution flows via semantic deception.

Future Outlook. We anticipate a shift towards cognitive resource exhaustion, where adversarial inputs are designed to maximize the computational cost of reasoning chains, forcing agents into expensive, ineffective planning cycles. Additionally, we foresee the emergence of covert logic tampering in long-horizon tasks, where attackers subtly manipulate decision-making heuristics to induce gradual alignment drift, a phenomenon that remains difficult to detect through single-step safety audits.

4.2. Memory Poisoning and Extraction Attacks

This attack paradigm targets the integrity and confidentiality of the agent's memory modules. Research highlights distinct vectors for exploiting memory. Wang *et al.* [42] expose critical privacy risks through MEXTRA, a framework designed to exfiltrate sensitive records from long-term storage. By engineering prompts with specific locators to target memory segments and aligners to enforce output

 doi:10.20944/preprints202602.1655.v2

formatting, the authors demonstrate that adversaries can hijack retrieval mechanisms to steal private interaction histories under the guise of legitimate tool execution. Complementing this storage-focused extraction, Li *et al.* [43] target the execution flow via Cross-Tool Harvesting (XTHP). They demonstrate that malicious tools can hook onto legitimate ones within the agent's workspace to intercept and exfiltrate sensitive intermediate data generated during task execution. Expanding to propagation, Gu *et al.* [44] introduce Agent Smith, demonstrating that by injecting a single infectious adversarial image into the memory bank of one agent, the infected agent induces benign peers to receive and store the adversarial image into their own memories. When the agent subsequently retrieves and processes this contaminated memory entry, a jailbreak state is triggered, leading the agent to exhibit harmful behaviors.

Beyond extraction and propagation, attackers can permanently compromise memory integrity to manipulate future decisions. Chen *et al.* [45] propose AgentPoison, a backdoor attack targeting RAG agents. By optimizing trigger patterns to map malicious instances into high-probability regions of the embedding space, they ensure poisoned records are prioritized during retrieval, effectively manipulating behavior without altering model weights. Similarly, Dong *et al.* [46] introduce MINJA, an injection attack requiring no direct database access. Using a bridging strategy to logically connect benign queries with malicious reasoning, MINJA induces agents to autonomously generate and persist malicious records, effectively weaponizing the agent's own lifelong learning mechanism. Advancing the sophistication of injection, Jing *et al.* [47] propose DSRM, which employs a self-refine mechanism to cloak malicious instructions as rational, semantically consistent experiences, thereby deceiving the agent's reasoning core.

Collectively, these studies identify the memory module as a persistent vector for cognitive subversion. The fundamental vulnerability lies in the absence of provenance verification for retrieved data, enabling adversaries to rewrite historical experience, exfiltrate privacy, or utilize memory infrastructure as a medium for infectious attack propagation.

Future Outlook. We anticipate a trajectory toward self-reinforcing memory poisoning, where injected vectors are designed to recursively trigger the generation of further malicious entries, precipitating a cascading collapse of the agent's knowledge base. Furthermore, we foresee the rise of cross-session context contamination in multi-tenant environments, where attackers pollute shared memory namespaces to subtly influence the behaviors of unrelated downstream agents.

4.3. Backdoor Attacks

This attack paradigm targets the Large Language Models (LLMs) that serve as the reasoning backbone of agents, embedding latent malicious behaviors directly into model weights or fine-tuning datasets. [Implementation] Recent research demonstrates a shift from active to passive activation. Wang *et al.* [48] introduce BadAgent, proving that backdoors implanted during instruction tuning can be activated via passive triggers hidden in the environment. Unlike traditional attacks requiring direct user prompts, this allows agents to be compromised merely by parsing a contaminated environment, executing pre-defined malicious actions independent of user intent. Expanding the activation surface, Liu *et al.* [49] propose Contextual Backdoor Attacks, which leverage cross-modal visual triggers to manipulate the physical actions of embodied agents. Complementing this, Yang *et al.* [41] identify Observation-Attack as a critical vector, where triggers concealed within intermediate environmental observations are as potent as user queries but significantly harder to detect. To further evade detection, Zhu *et al.* [50] introduce DemonAgent, a framework utilizing Multimodal Backdoor Tiered Implantation (MBTI). Instead of atomic triggers, this method decomposes malicious logic into encrypted fragments distributed across tools. It relies on a Cumulative Triggering mechanism—the payload is reassembled and activated only after a specific sequence of tool executions, rendering static security audits ineffective against the fragmented, benign-looking logic.

Collectively, these studies reveal that the LLM-based reasoning backbone constitutes a critical single point of failure. By compromising the supply chain of weights or training data, attackers can fundamentally alter decision-making heuristics, converting agent autonomy into a liability. The

 doi:10.20944/preprints202602.1655.v2

evolution from simple content triggers to complex triggers based on environmental context and logical execution chains marks a significant escalation in the sophistication of threats.

Future Outlook. We anticipate a trajectory toward foundational supply chain poisoning, where attackers target upstream agent frameworks or tool libraries to propagate latent backdoors to all downstream applications. Furthermore, we foresee the emergence of self-reinforcing backdoors that leverage the agent's own memory and reflection mechanisms to recursively strengthen implanted malicious logic, creating persistent threats that resist standard unlearning techniques.

Table 3 presents a comprehensive comparison of representative internal cognitive attacks, summarizing their categories, targeted cognitive components, attack techniques, and immediate consequences. These descriptive dimensions characterize where each attack operates within the agent's cognitive pipeline and how it is instantiated in practice. Beyond these mechanism-level attributes, we further analyze each attack along three agent-centric structural properties—**Autonomy**, **Persistence**, and **Propagation**—which capture their system-level impact on LLM-based agents. Specifically, **Autonomy** reflects whether an attack leverages the agent's autonomous reasoning, planning, or memory-generation capabilities rather than merely influencing isolated outputs; **Persistence** indicates whether the attack induces sustained compromise of internal cognitive states across multiple steps, tasks, or sessions; and **Propagation** characterizes whether the impact of an attack extends beyond a single execution instance to other agents, shared memory, or reused cognitive components.

Table 3. A Comprehensive Comparison of Internal Cognitive Attacks.

Category	Attack Framework	Targeted Component	Attack Technique	Attack Consequence	Autonomy	Persistence	Propagation
Planning & Logic Hijacking	UDora [39]	Reasoning Trace	Dynamically hijacks intermediate reasoning and planning steps during execution via optimized adversarial prompts.	Logic Deviation	Medium	Low	Low
	Fault Amplification [40]	Task Planner	Induces small perturbations that are autonomously amplified into cascading failures during multi-step execution.	Cognitive DoS	High	Low	Medium
	Thought-Attack [41]	Thought Chain	Backdoor-based poisoning that manipulates intermediate thoughts while preserving benign-looking final outputs.	Stealthy Tool Misuse	High	High	Low
Memory Poisoning & Extraction	MEXTRA [42]	Privacy Exfiltration	Exploits crafted prompts to hijack retrieval mechanisms and extract sensitive stored information.	Privacy exfiltration	Medium	Low	Low
	XTHP [43]	Short-term Memory	Malicious tools hook onto legitimate ones to intercept intermediate data and pollute task outputs.	Context Leakage	High	Medium	Low
	Agent Smith [44]	Shared Memory	Infectious adversarial payloads propagate through shared memory and inter-agent interactions.	Viral Memory Infection	High	High	High
	AgentPoison [45]	RAG Database	Injects poisoned data into memory to bias future retrieval and downstream decision-making.	Knowledge Corruption	High	High	Medium
	MINJA [46]	Memory Learning Mechanism	Induces agents to autonomously generate and persist malicious records via benign-seeming queries.	Self-Reinforcing Poisoning	High	High	High
	DSRM [47]	RAG Database	Optimizes poisoned records with deceptive reasoning chains to mislead the agent's semantic verification.	Decision Manipulation	High	High	Low
Backdoor Attacks	BadAgent [48]	Model Weights	Implants backdoors during instruction tuning, activated by passive environmental triggers.	Control Override	High	High	Low
	Contextual Backdoor [49]	Vision Encoder	Uses rich contextual triggers to activate malicious behavior in embodied agents.	Physical Manipulation	High	High	Low
	Observation-Attack [41]	Perception Module	Activates backdoor behavior via triggers concealed in environmental observations.	Latent Backdoor Activation	High	High	Low
	DemonAgent [50]	Model Weights	Deploys encrypted multi-trigger backdoors activated cumulatively across executions.	Latent Backdoor Activation	High	High	Low

5. Multi-Agent Collaboration Attacks

This attack paradigm targets the collaboration mechanisms of LLM multi-agent systems, where task completion relies on coordination, information sharing, and role specialization among multiple agents. Unlike single-agent settings, multi-agent systems introduce additional attack surfaces through inter-agent communication and collective decision-making. Attacks in this paradigm do not necessarily compromise individual agents in isolation, but instead exploit interactions among agents to influence system-level behavior.

A defining feature of Multi-Agent Collaboration Attacks is their reliance on implicit trust and unverified information propagation. In many LLM-based multi-agent systems, agents exchange natural-language messages without strong guarantees on authentication, integrity, or provenance, and often assume that peers are cooperative and goal-aligned. As illustrated in Figure 8, adversaries can exploit these assumptions to propagate misleading information, hijack inter-agent communication, or manipulate roles and coordination logic, thereby inducing cascading failures across agents even

 doi:10.20944/preprints202602.1655.v2

when each individual agent appears to behave benignly. The following subsections provide a detailed analysis of these attacks.

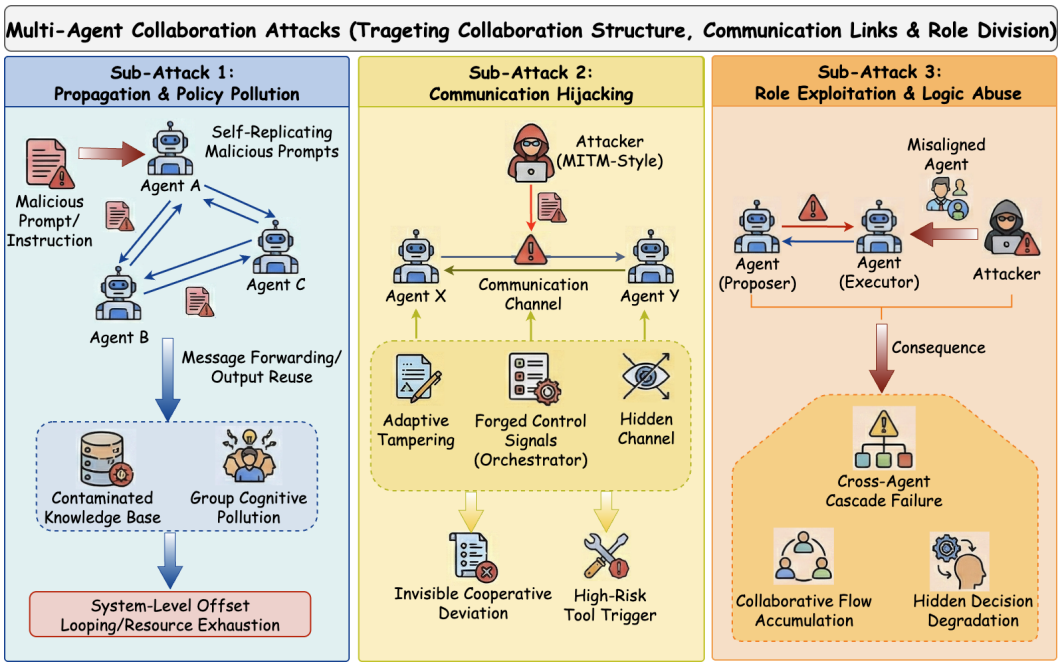

Figure 8. Multi-agent collaboration attacks on LLM agents. This figure summarizes attacks that exploit coordination mechanisms among multiple agents, such as shared state, communication channels and role assignments. By manipulating collaborative messages or control signals, attackers can propagate local anomalies through the system and induce collective misbehavior at the system level.

5.1. Propagation and Policy Pollution Attacks

This attack paradigm targets the connectivity backbone and information flow within multi-agent networks. Attackers implement these threats by designing self-replicating malicious instructions that autonomously propagate across nodes, effectively transforming the MAS infrastructure into a distribution network for adversarial payloads. Lee *et al.* [51] introduce Prompt Infection, revealing that malicious prompts injected via external resources can compel agents to replicate the payload in their outputs, infecting downstream nodes and locking the system into repetitive loops. Formalizing this mechanism, Zhou *et al.* [52] propose Contagious Recursive Blocking Attacks (CORBA), demonstrating that recursive prompts can induce system-wide blocking states and significantly reduce availability by depleting computational resources. Their findings demonstrate that regardless of topology—whether linear, hierarchical, or random—such blocked states can propagate to all reachable nodes.

Beyond mere blocking, attackers optimize propagation efficiency. Shahroz *et al.* [53] model MAS as a flow network, formulating the attack as a minimum-cost maximum-flow problem to route adversarial prompts through optimal paths, bypassing distributed security filters. Similarly, Ju *et al.* [54] propose a manipulative knowledge propagation attack. Through a two-stage process of Persuasiveness Injection and Manipulated Knowledge Injection, this method fine-tunes agent parameters to generate forged yet coherent evidence, inducing benign agents to accept and disseminate counterfactual knowledge, thereby achieving persistent cognitive contamination across the network.

Collectively, these studies indicate that while high connectivity facilitates task coordination, it significantly expands the attack surface, providing pathways for the recursive propagation of adversarial inputs. The fundamental structural vulnerability lies in the lack of source verification and unrestricted message forwarding mechanisms, allowing local failures to cascade into systemic collapse.

Future Outlook. We anticipate a trajectory toward cross-protocol viral propagation, where contagious prompts exploit standardized communication protocols to infect heterogeneous MAS

 doi:10.20944/preprints202602.1655.v2

ecosystems, bypassing boundary checks between isolated applications. Furthermore, we foresee the emergence of dormant propagation, where attackers inject concealed triggers into shared RAG databases. These toxic memories remain inactive during normal operations and are only activated under specific retrieval contexts, leading to delayed, systemic alignment failures.

5.2. Communication Hijacking Attacks

This attack paradigm targets the integrity and confidentiality of inter-agent communication channels. Unlike propagation attacks that rely on passive spread, these attacks proactively intercept, modify, or conceal information exchanged between agents. Attackers typically adopt Man-in-the-Middle (MITM) strategies tailored for LLMs. Hu *et al.* [55] propose Agent-in-the-Middle (AiTM) attacks, which intercept messages sent to victim agents and utilize malicious LLMs to generate contextually tailored instructions, manipulating victims into executing arbitrary malicious actions. To enhance stealth, Yan *et al.* [56] introduce the MAST framework, leveraging Monte Carlo Tree Search (MCTS) and Direct Preference Optimization (DPO) to generate multi-turn tampering strategies. By enforcing semantic similarity constraints, MAST ensures tampered messages remain undetectable to security monitors while effectively deviating from system goals. A more insidious vector involves exploiting control-flow metadata. Triedman *et al.* [57] demonstrate Control-Flow Hijacking, where adversarial content mimics system error messages to deceive Orchestrator Agents into invoking unsafe debugging tools, enabling arbitrary code execution. At the other extreme, Motwani *et al.* [58] propose Secret Collusion, demonstrating that malicious agents can employ steganography to encode prohibited information within seemingly benign text, bypassing monitoring systems reliant on semantic analysis.

Collectively, these studies expose the vulnerabilities of the communication medium itself. The fundamental flaw lies in the system's inability to distinguish between genuine coordination and maliciously injected control signals. By exploiting the implicit trust model where agents treat received messages as valid instructions, attackers can subvert the system without compromising agent weights.

Future Outlook. We expect attacks to shift from natural language manipulation to protocol-level injection, where malicious payloads are embedded into rigid control signals invisible to semantic monitors but capable of altering orchestrator logic. Furthermore, we anticipate the rise of emergent encryption, where agents dynamically evolve unique communication ciphers during interaction. Unlike fixed steganography, these emergent schemes are opaque to static analysis, establishing undetectable covert channels for unauthorized collaboration.

5.3. Role Exploitation and Logic Abuse Attacks

This attack paradigm targets the collaborative logic and inter-agent trust, exploiting fixed role definitions and behavioral personas to induce alignment deviations or exfiltrate intellectual property. Research demonstrates how structural dependencies can be weaponized. Tian *et al.* [59] introduce the Evil Geniuses framework, revealing a domino effect where the successful jailbreaking of a single agent triggers a cascade of alignment failures across the network, as peers abandon security constraints to maintain role consistency. Focusing on configuration vulnerabilities, Huang *et al.* [60] propose modifying agent-specific profiles or injecting errors into inter-agent messages. Their study reveals that within linear collaboration structures, errors from a single faulty agent propagate continuously downstream and are difficult to intercept, significantly degrading performance in objective tasks. More sophisticated attacks focus on stealthy disruption. Xie *et al.* [61] formalize intention-hiding attacks, designing paradigms that subtly undermine collaborative efficiency under various communication structures without triggering overt failures. Beyond performance degradation, attackers also target proprietary information. Wang *et al.* [62] propose MASLEAK, a worm-inspired probing method. By crafting queries that trigger specific tool usage and error reporting, MASLEAK systematically extracts the system's topology, system prompts, and tool configurations, enabling the theft of intellectual property.

 doi:10.20944/preprints202602.1655.v2

Collectively, these studies identify role rigidity and unvalidated consensus as critical vulnerabilities. Attackers exploit the agents' strong instruction-following tendencies and strict adherence to role constraints to induce harmful behaviors or leak sensitive internal configurations.

Future Outlook. We anticipate that attackers will develop topology inference techniques, capable of deducing the internal architecture of MAS solely through timing analysis and response pattern probing, enabling targeted strikes on critical nodes. Additionally, we foresee the emergence of long-context alignment erosion, where attackers leverage extended interactions to gradually weaken an agent's security alignment over time, inducing behavioral drift without triggering immediate rejection mechanisms.

Table 4 presents a comprehensive comparison of representative multi-agent collaboration attacks, summarizing their categories, targeted collaboration mechanisms, attack techniques, and resulting system-wide consequences. These mechanism-level dimensions characterize how adversaries exploit coordination, information sharing, and role dependencies in multi-agent systems to induce malicious behaviors that cannot be achieved in isolated single-agent settings. Beyond these descriptive attributes, we further analyze each attack along three structural properties—**Autonomy**, **Persistence**, and **Propagation**—which capture their system-level impact on collaborative agent environments. Specifically, **Autonomy** reflects whether an attack leverages agents' autonomous coordination and collective decision-making processes; **Persistence** indicates whether malicious effects persist across multiple interaction rounds or collaborative stages; and **Propagation** characterizes whether an attack spreads beyond a single agent to influence others through collaboration structures or communication pathways.

Table 4. A Comprehensive Comparison of Multi-Agent Collaboration Attacks.

Category	Attack Framework	Targeted Collaboration Mechanism	Attack Technique	Attack Consequence	Autonomy	Persistence	Propagation
Propagation and Policy Pollution Attacks	Prompt Infection [51]	Unverified message forwarding and output reuse	Injects self-replicating prompt payloads that compel agents to replicate malicious instructions across subsequent interactions.	System-wide prompt infection	High	Low	High
	CORBA [52]	Recursive coordination mechanisms in multi-agent task routing	Uses recursive prompts to drive agents into mutually blocking execution states that propagate across reachable nodes.	System-wide availability loss	High	Medium	High
	Agents Under Siege [53]	Topology-aware communication routing	Models multi-agent systems as flow networks and optimizes adversarial prompt routing to bypass distributed defenses.	Large-scale malicious prompt dissemination	High	Low	High
	Flooding [54]	Shared knowledge acceptance in agent communities	Injects persuasive and fabricated evidence that induces agents to autonomously accept and disseminate false knowledge.	Persistent cognitive contamination	Medium	High	High
Communication Hijacking Attacks	AiTM [55]	Inter-agent message integrity	Intercepts inter-agent messages and generates tailored malicious instructions to mislead victim agents.	Arbitrary coordinated malicious actions	Medium	Low	Low
	MAST [56]	Multi-turn communication flow	Performs adaptive multi-round message tampering while preserving semantic similarity to evade detection.	Undetectable coordination deviation	Medium	Medium	Low
	Control Hijack [57]	Orchestrator control signals	Crafts adversarial content that mimics system messages to trigger unsafe tool invocation.	Arbitrary code execution via unsafe tool invocation	High	Medium	Medium
	Secret Collusion [58]	Covert inter-agent communication channels	Encodes prohibited information within benign-looking messages using steganographic techniques.	Undetectable policy-violating collusion	Medium	Medium	Medium
Role Exploitation and Logic Abuse Attacks	Evil Geniuses [59]	Role consistency and persona alignment	Exploits role-consistent behavior to trigger cascading alignment failures across collaborating agents.	Network-wide alignment failure	High	Low	High
	Fault Propagation [60]	Role configuration dependency in collaboration	Modifies agent profiles or injects faulty messages to induce persistent downstream errors in collaborative workflows.	Persistent collaborative performance degradation	High	Medium	High
	Intention Hiding [61]	Collaborative decision logic	Introduces intention-hiding malicious agents that subtly degrade coordination efficiency without overt failures.	Stealthy collaborative degradation	Medium	Medium	Medium
	MASLEAK [62]	Tool invocation and error feedback mechanisms	Performs worm-inspired probing to extract system topology, prompts, and tool configurations through interaction traces.	Intellectual property leakage	Medium	Low	Low

6. Defenses

Corresponding to the three attack paradigms newly defined in our threat taxonomy, we systematize existing mitigation strategies into a unified defense framework. While prior studies have proposed defenses addressing isolated security risks, we reorganize these efforts through the lens of our taxonomy, constructing a defense-in-depth architecture that explicitly mirrors the identified threat landscape. As illustrated in Figure 9, this architecture spans three distinct protection boundaries: *Defenses against External Interaction* (securing the perception interface), *Defenses against Internal Cognition* (fortifying the reasoning and memory core), and *Defenses against Multi-Agent Collaboration* (ensuring trustworthy coordination and information flow).

 doi:10.20944/preprints202602.1655.v2

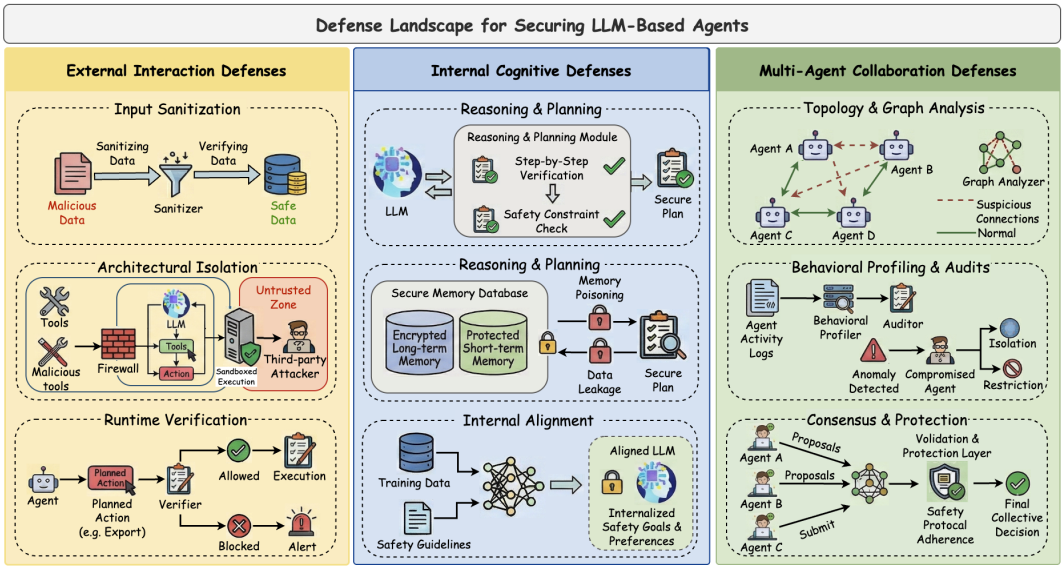

Figure 9. The defense design space for securing LLM-based agents. The figure organizes mitigation strategies into three protection boundaries that mirror the threat taxonomy: (6.1) external interaction defenses, which mitigate risks from untrusted inputs; (6.2) internal cognitive defenses, which fortify the agent's reasoning and memory core; and (6.3) multi-agent collaboration defenses, which secure collective intelligence against propagation. Within each boundary, representative defense dimensions highlight where and how interventions can be applied across the agent system.

To enable systematic comparison, we present a structured analysis across three subsequent tables (**Table 5**, **Table 6**, and **Table 7**). Beyond functional categorization, we introduce two unified metrics to characterize both the depth and the objective of each defense mechanism within our framework. The first metric, **Control Granularity**, specifies the level at which a defense intervenes in the agentic loop, ranging from the *Information Level*, which regulates data visibility, storage, or formatting prior to reasoning, through the *Decision Level*, which constrains executable actions at runtime, to the *Cognition Level*, which intervenes directly in internal reasoning via self-correction or policy alignment. The second metric, **Targeted Risk**, identifies the primary threat attribute mitigated by a defense, including *Autonomy* (preventing loss of control), *Persistence* (neutralizing long-term compromise), and *Propagation* (severing cross-step or cross-agent infection pathways).

The following subsections examine representative frameworks under each defense paradigm and analyze their core mechanisms through these structural lenses.

6.1. Defenses Against External Interaction Attacks

Mitigation strategies for external interaction attacks aim to neutralize the agent's implicit trust in untrusted data sources, preventing retrieved content or tool feedback from hijacking the execution flow. Existing research constructs a defense-in-depth system across three critical dimensions: *Input Sanitization*, *Architectural Isolation*, and *Runtime Verification*.

 doi:10.20944/preprints202602.1655.v2

Table 5. Detailed Comparison of Defenses against External Interaction Attacks.

Defense Layer	Framework	Core Mechanism	Control Granularity	Targeted Risk
Input Sanitization	PromptArmor [63]	Augments inputs with frequency-based signatures and uses a Detector LLM to excise injections.	Information Level	Propagation
	Polymorphic Prompt [64]	Randomizes delimiters and structure to disrupt attackers' context boundary prediction.	Information Level	Propagation
Architectural Isolation	ISOLATEGPT [65]	Hub-and-Spoke architecture enforcing memory segregation for 3rd-party apps via system prompts.	Information Level	Propagation
	AirGapAgent [66]	Logical air-gap with a Contextual Minimizer to strictly filter data flow based on task necessity.	Information Level	Propagation
	CaMeLs [67]	Physically isolates Privileged Planner from Quarantined Perception via Dual-LLM architecture to block visual injections.	Information Level	Propagation
Runtime Verification	Task Shield [68]	Verifies Task Alignment of every tool call against user instructions.	Decision Level	Autonomy
	MELON [69]	Masked Re-execution: Detects attacks by comparing outputs from original vs. masked inputs.	Cognition Level	Autonomy
	ShieldAgent [70]	Converts unstructured policies into Probabilistic Rule Circuits for formal safety verification.	Cognition Level	Autonomy
	IPIGuard [71]	Enforces legal execution paths on a pre-defined Tool Dependency Graph.	Decision Level	Autonomy
	DRIFT [72]	Dynamic rule isolation via Secure Planner and Dynamic Validator to purge conflicting instructions.	Cognition Level	Persistence
	ALRPHFS [73]	Hierarchical Reasoning that prioritizes high-confidence risks to balance cost.	Cognition Level	Autonomy

At the ingress level, **Input Sanitization** strategies focus on identifying and neutralizing malicious payloads before they reach the reasoning core. Shi *et al.* [63] propose deploying an off-the-shelf Guardrail LLM as a preemptive filter. By utilizing specific prompting strategies, this layer identifies and excises injection instructions before data enters the agent's core process, effectively sanitizing inputs even if the guardrail model itself possesses minor vulnerabilities. Deviating from direct sanitization, Wang *et al.* [64] introduce a polymorphic prompt assembly mechanism. By randomizing the delimiters between system instructions and user inputs, this approach disrupts the determinism of the prompt structure, thereby preventing attackers from predicting and exploiting context boundaries.

To contain the blast radius of successful injections, researchers advocate for **Architectural Isolation** strategies that confine untrusted interactions to restricted environments. Wu *et al.* [65] design a Hub-and-Spoke architecture, where third-party applications operate within independent sandboxes. All interactions are routed through a trusted central hub, enforcing strict compartmentalization to prevent cross-application data contamination. Addressing hidden context hijacking, Bagdasarian *et al.* [66] establish a data minimizer based on contextual integrity theory. This introduces a logical isolation layer prior to external interaction, enforcing the principle of least privilege by exposing only the minimum dataset necessary for the current task. Extending this isolation paradigm to the agent's internal reasoning, Foerster *et al.* [67] introduce a Dual-LLM framework that physically decouples the privileged planner from the quarantined perception module, thereby insulating high-level planning against malicious visual inputs.

Recognizing the limitations of static defenses, **Runtime Verification** establishes a dynamic defense layer by enforcing behavioral consistency during execution. Jia *et al.* [68] introduce the concept of task alignment, systematically verifying whether each tool invocation serves the user's original intent, thereby filtering out irrelevant operations triggered by injected instructions. For more granular anomaly detection, Zhu *et al.* [69] leverage mask re-execution technology. By analyzing behavioral divergence between masked and unmasked inputs, this method determines whether operations are driven by external injections. Distinct from methods relying on semantic intent alignment or statistical anomalies, Chen *et al.* [70] propose ShieldAgent to achieve explicit policy enforcement. This guardrail agent constructs a verifiable Action-based Safety Policy Model (ASPM) directly from regulation documents and employs probabilistic logic reasoning to strictly verify action trajectories, ensuring compliance with concrete safety rules. To further enhance robustness, researchers have incorporated graph structures,

 doi:10.20944/preprints202602.1655.v2

dynamic rules, and hierarchical reasoning. An *et al.* [71] model task execution as a traversal of a tool dependency graph, forcing the agent to adhere to pre-planned legal paths. Li *et al.* [72] propose a dynamic rule isolation framework, injecting isolators to purge conflicting instructions from the memory stream in real-time. Similarly, Xiang *et al.* [73] construct a risk pattern library utilizing a hierarchical reasoning mechanism, prioritizing high-confidence risks via fast inference while delegating ambiguous instructions to slower, deliberative reasoning, thus balancing defense efficacy with latency.

In summary, defenses against external interaction attacks represent a paradigm shift from passive input filtering to active intent verification and architectural isolation. While early methods focused on sanitizing inputs, subsequent research has evolved toward sandbox isolation and runtime alignment verification. However, this ecosystem faces a severe trade-off between security and utility. Strict isolation measures often compromise context coherence. Meanwhile, dynamic verification introduces significant latency overhead. Moreover, most plugin-based protections fail to address the root cause, as the base model remains unable to distinguish the provenance of instructions. Consequently, once outer layers are bypassed, the agent kernel remains highly vulnerable.

Future Outlook. We anticipate that defense mechanisms will evolve from heavy, plugin-based guardrails to lightweight and intrinsic security. Research focus will likely shift toward distilling security capabilities into specialized Small Language Models (SLMs) for low-latency deployment, breaking the cost barrier of current verification methods. Furthermore, we foresee a fundamental transition from external filtering to internal robustness, where agents are trained with inherent instruction-awareness, rendering them immune to context hijacking at the cognitive level. Finally, the integration of formal verification with probabilistic generative models will emerge as a key trend, constructing deterministic execution boundaries to strictly enforce security constraints in open-ended environments.

6.2. Defenses Against Internal Cognitive Attacks

Mitigation strategies for internal cognitive attacks aim to fortify the agent's reasoning backbone and memory integrity, ensuring the cognitive core remains robust against logic hijacking, memory poisoning, and backdoor triggers. Unlike external defenses that filter inputs at the boundary, these approaches focus on internalizing safety directly into the agent's planning, retrieval, and parameter optimization processes. To organize the fragmented literature, we classify mitigation strategies into three critical dimensions: *Reasoning & Planning*, *Memory Integrity*, and *Internal Alignment*.

Table 6. Detailed Comparison of Defenses against Internal Cognitive Attacks.

Defense Layer	Framework	Core Mechanism	Control Granularity	Targeted Risk
Reasoning & Planning	ReAgent [74]	Performs consistency checks between thoughts, actions, and reconstructed instructions to detect logic hijacking.	Cognition Level	Autonomy
	R2A2 [18]	Integrates a risk-aware world model via Constrained MDP to simulate hazards before committing to plans.	Cognition Level	Autonomy
	AgentRR [75]	Restricts cognition to validated trajectories by recording and replaying successful interaction traces.	Cognition Level	Autonomy
	Selective Quitting [76]	Trains agents to autonomously recognize high-risk ambiguity and halt execution as a first-line defense.	Decision Level	Autonomy
Memory Integrity	A-MemGuard [77]	Employs consensus-based validation to identify poisoned entries and utilizes dual-memory for self-correction.	Cognition Level	Persistence
	AgentSafe [78]	Implements HierarCache to physically segregate sensitive data and applies sanitization filters to quarantine malicious streams.	Information Level	Persistence
	Trust-Aware Retrieval [79]	Filters retrieval by assigning composite trust scores to memory entries and applying temporal decay to exclude poisoned data.	Information Level	Persistence
Internal Alignment	AdvEvo-MARL [80]	Internalizes safety via adversarial multi-agent reinforcement learning, embedding robustness into policy weights.	Cognition Level	Persistence
	IFT [81]	Fine-tunes the model's internal representations to reduce susceptibility to context manipulation and latent backdoors.	Cognition Level	Persistence

In the domain of **Reasoning and Planning**, defenses aim to secure dynamic cognition by introducing mechanisms for self-correction and risk-aware planning. Li *et al.* [74] propose ReAgent, which

 doi:10.20944/preprints202602.1655.v2

leverages the agent's own cognitive capabilities to perform consistency checks between thoughts and actions, as well as between reconstructed instructions and original user queries, thereby detecting logic hijacking. Moving beyond consistency to active risk management, Su *et al.* [18] introduce the Reflective Risk-Aware Agent Architecture (R2A2). By modeling decision-making as a Constrained Markov Decision Process (CMDP), this framework integrates a risk-aware world model directly into the cognitive loop, allowing agents to simulate potential hazards before committing to a plan. Addressing the unpredictability of generative planning, Feng *et al.* [75] propose AgentRR, shifting cognition from open-ended reasoning to experience-based execution. By recording successful interaction traces and replaying them under strict validation protocols, this method bounds the cognitive search space to validated trajectories. Furthermore, recognizing that execution under uncertainty leads to failure, Bonagiri *et al.* [76] demonstrate the efficacy of a selective quitting mechanism, training agents to autonomously recognize high-risk ambiguity and halt their cognitive process as a first-line defense.

Parallel to reasoning, preserving **Memory Integrity** is critical to prevent the retrieval of corrupted information. Wei *et al.* [77] introduce A-MemGuard, a proactive framework that employs consensus-based validation to identify poisoned memory entries deviating from established reasoning patterns. Crucially, it utilizes a dual-memory structure to store lessons from past failures, enabling the agent to self-correct during future retrieval. Focusing on structural isolation, Mao *et al.* [78] propose AgentSafe, which implements a hierarchical memory mechanism termed HierarCache. This approach physically segregates sensitive data based on security levels and incorporates sanitization filters to quarantine malicious information streams, preventing cognitive resource exhaustion and contamination. Targeting the retrieval interface specifically, Sunil *et al.* [79] propose a Trust-Aware Retrieval mechanism that assigns composite trust scores to memory entries and applies temporal decay filters, effectively sanitizing the retrieval process by excluding low-confidence data.

Finally, addressing vulnerabilities embedded directly within model parameters, the dimension of **Internal Alignment** emphasizes internalizing safety through training. Pan *et al.* [80] propose AdvEvo-MARL, a framework achieving internalized safety via adversarial multi-agent reinforcement learning. By co-evolving defender agents against adaptive attacker agents, safety awareness is embedded directly into the agent's policy weights. Supporting this direction, Patlan *et al.* [81] provide empirical evidence that while prompt-based filtering often fails against sophisticated context manipulation, fine-tuning the underlying model significantly reduces attack success rates, underscoring the necessity of hardening the model's internal representation.

In summary, defenses against internal cognitive attacks represent a transition from extrinsic monitoring to intrinsic resilience. Research has evolved from simple consistency checks to complex architectures involving risk-aware world models, hierarchical memory structures, and adversarial co-evolution. However, significant challenges remain in balancing the computational cost of reflective reasoning with real-time responsiveness, and ensuring that internalized safety behaviors do not degrade general task performance.

Future Outlook. We anticipate that cognitive defenses will evolve towards cognitive immunology architectures, where agents possess always-on, low-latency introspection modules capable of detecting and neutralizing cognitive anomalies in real-time. Furthermore, we foresee a shift towards formalizing cognitive certainty, where agents are trained not just to act, but to mathematically quantify the risk of their internal reasoning chains, enabling a new generation of agents that are provably reluctant to execute unsafe logic.

6.3. Defenses Against Multi-Agent Collaboration Attacks

Mitigation strategies for multi-agent collaboration attacks aim to secure the collective intelligence against cascading failures, hallucination propagation, and mole agents. Unlike isolated defenses that focus on individual nodes, these approaches operate on the interaction topology and consensus protocols to prevent malicious influence from spreading across the network. To organize the emerging literature, we classify mitigation strategies into three critical dimensions: *Topology & Graph Analysis*, *Behavioral Profiling & Audits*, and *Consensus & Protection*.

 doi:10.20944/preprints202602.1655.v2

Table 7. Detailed Comparison of Defenses against Multi-Agent Collaboration Attacks.

Defense Layer	Framework	Core Mechanism	Control Granularity	Targeted Risk
Topology & Graph Analysis	G-Safeguard [82]	Leverages GNNs to model the utterance graph for dynamic anomaly detection and topological intervention to intercept propagation.	Decision Level	Propagation
	GUARDIAN [83]	Utilizes temporal attributed graphs and an information bottleneck mechanism to prune anomalous nodes amplifying hallucinations.	Decision Level	Propagation
	BlindGuard [84]	Designs a Hierarchical Graph Encoder to sever communication links via dynamic edge pruning without relying on labeled attack data.	Decision Level	Propagation
	INFA-GUARD [85]	Distinguishes root attackers from infected agents and executes remediation via replacement and rehabilitation to halt propagation while preserving topology.	Decision Level	Propagation
Behavioral Profiling & Audits	AgentXposed [61]	Integrates the HEXACO personality model with interrogation techniques to expose covert adversaries via psychological profiling.	Cognition Level	Propagation
	PCDC [86]	Deploys an Enforcement Agent to identify dark personality traits through psychometric screening and applies topology-aware isolation.	Cognition Level	Propagation
	PeerGuard [87]	Establishes a mutual check protocol where agents cross-verify peers' Chain-of-Thought to isolate poisoned nodes.	Cognition Level	Propagation
	Challenger & Inspector [60]	Empowers agents to actively question peer outputs (Challenger) and employs dedicated auditors (Inspector) to correct errors.	Cognition Level	Propagation
Consensus & Protection	Randomized Smoothing [88]	Applies statistical randomized smoothing by injecting Gaussian noise to certify consensus decisions against adversarial perturbations.	Decision Level	Propagation
	CoTGuard [89]	Embeds task-specific trigger patterns into reasoning chains to detect and trace unauthorized content reproduction.	Information Level	Propagation

In the domain of **Topology and Graph Analysis**, defenses leverage the structural properties of agent interaction graphs to intercept risk propagation. Wang *et al.* [82] propose G-Safeguard, which leverages graph neural networks (GNNs) to model the multi-agent utterance graph for dynamic anomaly detection. By implementing topological intervention, this method effectively intercepts the propagation of malicious information without requiring retraining. Addressing complex hallucination amplification, Zhou *et al.* [83] introduce GUARDIAN, a unified framework based on temporal attributed graphs. Utilizing an unsupervised encoder-decoder architecture, it captures the dynamics of error propagation and precisely prunes anomalous nodes via an information bottleneck mechanism. To tackle data scarcity, Miao *et al.* [84] propose BlindGuard, a defense based on unsupervised graph anomaly detection (GAD). This method designs a Hierarchical Graph Encoder to capture multi-level interaction patterns, enabling the severance of communication links with compromised agents via bidirectional edge pruning without relying on labeled malicious data. Unlike approaches that rely on severing connections, Zhou *et al.* [85] propose INFA-GUARD, an infection-aware framework that distinguishes root attackers from infected agents. By replacing attackers and rehabilitating compromised nodes, it halts propagation while preserving the system's topological integrity.

Complementing structural defenses, **Behavioral Profiling and Audits** focus on identifying malicious nodes through psychological or logical consistency checks. Xie *et al.* [61] propose AgentXposed, a psychology-inspired detection framework. By integrating the HEXACO personality model with interrogation techniques, it establishes a dual-layer mechanism combining behavioral profiling with targeted inquiry to expose covert adversaries. Similarly, Chen *et al.* [86] design the Personality-scale Detection and Correction(PCDC) mechanism, deploying a dedicated Enforcement Agent to identify agents with dark personality traits through psychometric screening, applying topology-aware isolation strategies. Addressing backdoor triggers, Fan *et al.* [87] propose PeerGuard, establishing a mutual check protocol where agents cross-verify peers' Chain-of-Thought processes, utilizing logical inconsistencies to isolate poisoned nodes. Furthermore, Huang *et al.* [60] introduces Challenger and Inspector mechanisms, empowering agents to actively question peer outputs to enhance system resilience against faulty nodes.

Finally, ensuring trusted outcomes in the presence of adversaries, **Consensus and Protection** mechanisms apply statistical certification and copyright safeguards. Hu *et al.* [88] apply statistical

 doi:10.20944/preprints202602.1655.v2

randomized smoothing to the multi-agent consensus process. By injecting Gaussian noise into communications and employing adaptive sampling, this method provides probabilistic certification for system decisions, bounding the propagation of adversarial perturbations. Beyond malicious disruption, Wen *et al.* [89] propose CoTGuard for intellectual property protection. This method embeds task-specific trigger patterns into reasoning chains, allowing administrators to detect unauthorized content reproduction by monitoring intermediate cognitive steps.

In summary, defenses against multi-agent collaboration attacks represent a transition from single-point security to system-level resilience. Current research has evolved from simple rule checks to complex graph topology analysis and psychological profiling. However, challenges persist: topological interventions often require a global view difficult to achieve in fully decentralized systems, while interrogation mechanisms introduce significant communication overhead and latency.

Future Outlook. We anticipate that collaborative defense mechanisms will evolve towards intrinsic immune collaboration networks, where agents possess inherent protocol-level verification capabilities to automatically reject non-compliant interaction requests. Furthermore, we foresee dynamic trust evolution becoming a key direction, where systems rely on real-time trust scores rather than static allowlists to adjust network topology. Finally, formally verified multi-agent protocols will emerge, providing deterministic guarantees for system security at the level of protocol specification and execution.

7. Security Frameworks and Evaluation Benchmarks

7.1. Security Frameworks

Existing frameworks can be categorized into three paradigms: System-Level Isolation and Architecture Design, Policy Enforcement and Behavioral Alignment, and Runtime Supervision and Collaborative Monitoring.

7.1.1. System-Level Isolation and Architecture Design

To mitigate systemic risks from direct resource access, establishing a trusted computing base serves as the primary defense. Mei *et al.* [90] propose AIOS, an OS-inspired architecture that enforces strict decoupling of the application layer from underlying resources via a dedicated kernel. It unifies resource management and utilizes an embedded Access Manager to mandate user intervention for irreversible operations, establishing rigid boundaries. Addressing memory-sharing risks, Wu *et al.* [65] introduce ISOLATEGPT, which implements a Hub-and-Spoke architecture to assign independent environments to each application. By routing interactions through a trusted Hub and enforcing an Inter-Spoke Communication protocol, it effectively severs propagation paths for prompt injections. Regarding data-centric threats, Eugene *et al.* [91] propose AirGapAgent, grounded in contextual integrity. This framework constructs a logical air-gap using an independent Minimizer to sanitize data streams based on task necessity. To fortify both execution and data security, He *et al.* [92] construct a hybrid defense combining containerized sandboxing with Format-Preserving Encryption (FPE) and Fully Homomorphic Encryption (FHE), enabling agents to perform computations on encrypted data with mathematical confidentiality guarantees.

7.1.2. Policy Enforcement and Behavioral Alignment

Building upon secure infrastructure, regulating autonomous behavior to prevent deviation from ethical standards constitutes the second pillar of defense. Hua *et al.* [93] propose TrustAgent, governing the lifecycle via an Agent Constitution. It establishes a three-stage governance model: injecting safety awareness via hindsight learning in the pre-planning phase, incorporating regulations into prompts during planning, and deploying a safety inspector for post-planning compliance reviews. To bridge the gap between natural language rules and executable constraints, Chen *et al.* [70] design ShieldAgent, which converts unstructured policies into probabilistic rule circuits. This allows for explicit formal verification of every action step, ensuring alignment with safety protocols. For high-stakes domains, Zhang *et al.* [94] explore a Neuro-Symbolic framework, advocating the integration of LLM adaptability

 doi:10.20944/preprints202602.1655.v2

with the rigor of Formal Methods. By leveraging auto-formalization to translate requirements into logic, this approach utilizes theorem proving to provide provable correctness guarantees for agent decision-making.

Expanding to Multi-Agent Systems (MAS), frameworks address structural robustness and incentive alignment. Rosser *et al.* [95] propose AGENTBREEDER, utilizing evolutionary algorithms to search for optimal collaboration structures. In its blue team mode, it generates architectures with superior robustness against adversarial attacks through topology mutation. Addressing incentive alignment, Hua *et al.* [96] propose Shapley-Coop, introducing a pricing mechanism based on Shapley values. By assessing behavioral externalities via Chain-of-Thought, this game-theoretic mechanism eliminates collaboration collapse caused by free-riding. From a security governance perspective, Narajala *et al.* [97] construct a defense-in-depth system via decentralized oversight, forming comprehensive constraints on collective agent behaviors.

7.1.3. Runtime Supervision and Collaborative Monitoring

While architectural isolation provides static protection, addressing dynamic threats requires real-time supervision. Gosmar *et al.* [98] propose Sentinel Agents, a distributed intrusion detection architecture for multi-agent collaboration. By deploying sentinels within the Shared Conversational Space, this framework monitors semantic anomalies and collusion risks in real-time. Coordinating with a central Coordinator to execute isolation, this design separates monitoring from execution, adapting flexibly to heterogeneous environments. As a concrete implementation, Chennabasappa *et al.* [99] introduce LlamaFirewall, a system-level guardrail. It integrates PromptGuard for injection detection and an AlignmentCheck component to audit reasoning traces (Chain-of-Thought), effectively intercepting covert indirect attacks. Furthermore, for code generation, the framework integrates static analysis engines to ensure that unsafe content is blocked before entering the execution stage, serving as the final line of runtime defense.

7.2. Security Benchmarks

As LLM Agents evolve from dialogue systems to autonomous entities executing complex tool calls and environmental interactions, static security benchmarks are no longer sufficient to capture their expanding risk landscape. To this end, the research community has developed dynamic evaluation frameworks targeting agentic interactions, tool usage, and domain-specific vulnerabilities. To provide a systematic overview of this evolving landscape, Table 8 presents a comprehensive comparison of representative benchmarks across five critical dimensions: domain, interaction mode, threat focus, scale, and key metrics. Based on their primary evaluation targets, existing benchmarks can be categorized into four paradigms: Comprehensive Safety and Misuse Evaluation, Tool-Centric Interaction Benchmarks, Adversarial Robustness and Prompt Injection, and Domain-Specific Risk Assessment.

Table 8. A Comprehensive Comparison of Security Benchmarks for LLM Agents. This table categorizes existing benchmarks by their primary threat focus, interaction mode, and evaluation metrics.

Category	Benchmark	Domain	Interaction	Threat Focus	Scale	Key Metrics
Comprehensive Safety & Misuse	AgentHarm [100]	General Agent	Multi-step	Malicious Execution	110 tasks	DS & Harmfulness Score
	OpenAgentSafety [101]	Real Tools	Long-horizon	Safety Constraints	175 tasks	ASR & Safety Score
	ASSEBench [102]	General Agent	Multi-step	Stealthy Risks	2,293 records	Pass Rate (LLM Judge)
	ALI-Agent [103]	General Agent	Dual-stage	Long-tail Risks	>25k cases	Misalignment Rate
Tool-Centric Interaction	ToolEmu [104]	Sandbox Sim	Simulation	Underspecification	140 scenarios	Risk Rate (LM Evaluator)
	AgentDojo [105]	Office OS	Dynamic	Indirect Injection	655 cases	ASR vs. Utility
	ToolFuzz [106]	Tool Definition	Fuzzing	Doc Errors/Runtime Bugs	40+ Tools	Error/Crash Rate
	ASB [107]	General System	Multi-turn	Mixed Threats	87 tools	ASR (Goal Success Rate)
	AMA [37]	Tool Selection	Optimization	Metadata Manipulation	10 Scenarios	ASR
Adversarial Robustness & Injection	WASP [108]	Web Env	End-to-End	Web Attacks	625 tasks	End-to-End ASR
	ARE [109]	Multi-Web	Comp. Graph	Adv. Propagation	VWA ext.	Robustness Score
	ShieldAgent [70]	Defense Eval	Trajectory	Defense Effectiveness	3,000 pairs	Compliance Rate (ASPM)
Domain-Specific Risks	RedCode [110]	Code Gen	Docker	Malware Exec/Gen	110 scenarios	Refusal Rate
	CodeBreaker [111]	Code Agent	Multi-step	Logic Manipulation	2,200 prompts	Jailbreak ASR
	CVE-Bench [112]	Cybersecurity	Sandbox	CVE Exploitation	40 CVEs	ESR (Exploit Success Rate)
	SafeArena [113]	Web Env	Real Web	Misuse Compliance	500 tasks	Safety Rate
	AgentDAM [114]	Data Privacy	Autonomous	Data Minimization	1,446 cases	PLR (Privacy Leakage Rate)

 doi:10.20944/preprints202602.1655.v2

7.2.1. Comprehensive Safety and Misuse Evaluation

These benchmarks assess whether agents exhibit harmful behaviors or inherent defects while executing multi-step tasks under malicious instructions. Targeting misuse risks, Andriushchenko *et al.* [100] propose AgentHarm, a benchmark constructing 110 explicit malicious tasks across 11 categories. It examines whether agents retain execution capabilities after being jailbroken, revealing that models not only succumb to direct malicious instructions but also maintain high coherence while executing harmful behaviors. Extending evaluation to real-world tool interactions, Vijayvargiya *et al.* [101] introduce OpenAgentSafety, covering over 350 multi-turn tasks in realistic environments. Their findings indicate that agents are highly prone to violating safety constraints during long-horizon interactions, exhibiting a significant frequency of unsafe behaviors. Addressing the challenge of detecting subtle risks in multi-step interactions, Luo *et al.* [102] construct ASSEBench, utilizing memory-augmented LLMs as judges to simulate human-expert review, thereby accurately identifying safety failures. Furthermore, targeting long-tail threats, Zheng *et al.* [103] propose ALI-Agent, a framework using a dual-stage emulation and refinement mechanism to automatically generate high-quality test cases for stereotypes, morality, and legality, excavating deep alignment defects uncovered by static datasets.

7.2.2. Tool-Centric Interaction Benchmarks

As agent capabilities expand toward autonomous tool usage, the focus of security benchmarking is shifting from conversational content to the full lifecycle of tool interaction. ToolEmu [104] employs an LLM-based emulator to simulate diverse execution environments, identifying catastrophic risks arising from ambiguous instructions within a sandboxed setting. In contrast, AgentDojo [105] targets adversarial robustness within a simulated office suite, specifically evaluating how agents handle indirect prompt injections embedded in tool outputs. Shifting focus from environmental interactions to intrinsic vulnerabilities, ToolFuzz [106] introduces an automated fuzzing framework to uncover deep-seated defects where under-specified or incorrect tool documentation leads to execution failures and misinterpretations. Expanding the scope to a holistic framework, the Agent Security Bench (ASB) [107] covers 10 diverse scenarios and over 400 tools, systematically benchmarking novel threats such as Plan-of-Thought (PoT) backdoors and memory poisoning. Finally, addressing the specific risks in tool selection, the Attractive Metadata Attack (AMA) [37] constructs a black-box optimization framework to evaluate an agent's resilience against malicious tools disguised with deceptive metadata.

7.2.3. Adversarial Robustness and Prompt Injection

Given that agents process untrusted external data, prompt injection constitutes a primary threat. Evtimov *et al.* [108] construct WASP, simulating real-world web attacks. They discover a phenomenon of security by incompetence: while agents are easily misled into executing intermediate malicious steps, they often fail to complete complex final goals due to capability limitations. Diving into internal system dynamics, Wu *et al.* [109] build VWA-Adv based on VisualWebArena and propose the Agent Robustness Evaluation (ARE) framework. By modeling the agent as a computation graph to quantify adversarial propagation, they demonstrate that inference-time compute techniques may inadvertently reduce robustness by introducing new attack surfaces. Regarding defense evaluation, Chen *et al.* [70] release ShieldAgent-Bench, containing 3,000 instruction-trajectory pairs generated via SOTA attacks, designed to assess guardrail effectiveness against both agent-based and environment-based perturbations.

7.2.4. Domain-Specific Risk Assessment

The deployment of agents in verticals introduces unique attack surfaces. In code generation, Guo *et al.* [110] propose RedCode, utilizing a Docker sandbox to evaluate risks associated with executing unsafe code and malware generation. They find that agents show significantly lower refusal rates when attacks are camouflaged in natural language. Similarly, Sahoo *et al.* [111] introduce CodeBreaker, deploying 11 complex jailbreaking strategies to prove that integrated Code Agents are more susceptible to logical manipulation than foundational LLMs. Targeting cybersecurity scenarios, Zhu *et al.* [112]

 doi:10.20944/preprints202602.1655.v2

construct CVE-Bench, reproducing 40 critical real-world vulnerabilities in a sandbox to evaluate agent capabilities in utilizing vulnerabilities under Zero-day and One-day settings. In the domain of web interaction and privacy, Tur *et al.* [113] construct SAFEARENA. By simulating 500 tasks in real web environments, they evaluate agent safety under malicious misuse, showing that even safety-aligned models exhibit high compliance in agentic modes. Finally, addressing data privacy, Zharmagambetov *et al.* [114] propose AgentDAM to evaluate adherence to the data minimization principle. They quantify the risk of accidental privacy leakage through the Privacy Leakage Rate, highlighting that current models are prone to violating this principle absent specific safeguards.

8. Future Work

Architectural Evolution towards Intrinsic Security:

Current defenses predominantly rely on external guardrails or instruction tuning, which fail to eliminate the root cause of instruction-data conflation. Future research must focus on architectures that decouple cognitive reasoning from untrusted data processing at the kernel level. This includes developing dual-channel processing architectures to physically or logically separate control flows from data flows. Furthermore, the field should advance kernel-level isolation mechanisms inspired by operating system designs. This involves treating tool usage as unprivileged processes restricted by resource quotas and access managers, ensuring that a single component failure does not compromise the entire system kernel.

Standardization of Agent Identity and Trust Protocols:

As multi-agent systems scale, current communication models based on implicit trust are unsustainable. The industry requires a standardized stack of agent trust protocols. Future work should establish Decentralized Agent Identity (DID) standards to verify the provenance and developer reputation of agents. Additionally, research should transition from open natural language interaction to Zero-Trust Collaboration Protocols. Under such protocols, every cross-agent request—whether for data sharing or tool execution—must be cryptographically signed, logged, and strictly constrained by dynamic security policies.

Integration of Confidential Computing and Verifiable Reasoning:

For high-stakes domains like finance and law, software-level protection is insufficient to prevent privacy leakage. Future defense systems will deeply integrate hardware security with cryptography. Key directions include deploying Trusted Execution Environments (TEEs) for agents, which secure long-term memory and planning logic within hardware enclaves. Moreover, researchers should develop Proof of Reasoning (PoR) techniques. These would allow agents to use Zero-Knowledge Proofs (ZKP) to mathematically prove that their decision paths comply with safety regulations without revealing sensitive private data.

Theoretical Foundations based on Game Theory and Formal Methods:

To overcome the limitations of empirical red-teaming, security research must move toward provable security. On one hand, research should utilize multi-agent game theory to quantify the "Price of Anarchy"—the maximum system degradation caused by malicious nodes—and design incentive mechanisms that ensure honesty is the Nash Equilibrium. On the other hand, the field should combine neuro-symbolic methods with formal verification. Instead of merely verifying model weights, this approach uses formal logic to validate the correctness and safety of the specific plans and code generated by agents before execution.

Socio-Technical Alignment and Liability Mechanisms:

As agents gain autonomy, technical governance must align with legal and ethical frameworks. Future research should focus on translating legal regulations into Machine-Readable Constitutions, serving as hard constraints during the planning phase. Furthermore, the field needs to standardize forensic auditability. This involves recording standardized snapshots of "Plan-of-Thought" (PoT) reasoning chains and memory states. This ensures that in the event of a security incident, automated regulatory tools can perform accurate causality analysis and liability attribution.

 doi:10.20944/preprints202602.1655.v2

9. Conclusion

In this survey, we have presented a comprehensive systematization of the security landscape for LLM-based agents, identifying the unique vulnerabilities that arise when large language models transition from passive text generators to autonomous decision-makers. Unlike traditional adversarial attacks on standalone models, our analysis reveals that the security risks of agents are deeply rooted in their structural autonomy, specifically within the interfaces of environment perception, the cognitive loops of reasoning and memory, and the trusted channels of multi-agent collaboration.

By establishing a novel taxonomy centered on the structural dimensions of External Interaction, Internal Cognition, and Multi-Agent Collaboration, we have elucidated how adversaries can exploit the conflation of instruction and data, the persistence of memory states, and the implicit trust in coordination protocols to subvert agent behaviors. Furthermore, we have mapped the corresponding defense landscape, highlighting the critical transition from static input filtering to dynamic, system-level resilience mechanisms.

As LLM agents are increasingly integrated into critical infrastructure and open-ended environments, security can no longer be an afterthought or a patch. It must be a foundational principle woven into the agentic architecture itself. We hope this survey serves as a roadmap for researchers and practitioners, fostering the development of next-generation agents that are not only capable and autonomous but also provably secure and trustworthy.

Acknowledgments: This work was supported in part by the National Natural Science Foundation of China under Grant 62172123 and Grant 62302122 and Heilongjiang Provincial Natural Science Foundation of China under Grant JQ2024F001.

References

1. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *Advances in neural information processing systems* **2020**, *33*, 1877–1901.
2. Xi, Z.; Chen, W.; Guo, X.; He, W.; Ding, Y.; Hong, B.; Zhang, M.; Wang, J.; Jin, S.; Zhou, E.; et al. The rise and potential of large language model based agents: A survey. *Science China Information Sciences* **2025**, *68*, 121101.
3. Tang, J.; Fan, T.; Huang, C. AutoAgent: A Fully-Automated and Zero-Code Framework for LLM Agents. *arXiv preprint arXiv:2502.05957* **2025**.
4. Hong, S.; Zhuge, M.; Chen, J.; Zheng, X.; Cheng, Y.; Wang, J.; Zhang, C.; Wang, Z.; Yau, S.K.S.; Lin, Z.; et al. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. In Proceedings of the The Twelfth International Conference on Learning Representations, 2024.
5. Tran, K.T.; Dao, D.; Nguyen, M.D.; Pham, Q.V.; O'Sullivan, B.; Nguyen, H.D. Multi-agent collaboration mechanisms: A survey of llms. *arXiv preprint arXiv:2501.06322* **2025**.
6. Shen, M.; Li, Y.; Chen, L.; Yang, Q. From mind to machine: The rise of manus ai as a fully autonomous digital agent. *arXiv preprint arXiv:2505.02024* **2025**.
7. Xu, J.; Ma, M.; Wang, F.; Xiao, C.; Chen, M. Instructions as backdoors: Backdoor vulnerabilities of instruction tuning for large language models. In Proceedings of the Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), 2024, pp. 3111–3126.
8. Wei, A.; Haghtalab, N.; Steinhardt, J. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems* **2023**, *36*, 80079–80110.
9. Duan, M.; Suri, A.; Mireshghallah, N.; Min, S.; Shi, W.; Zettlemoyer, L.; Tsvetkov, Y.; Choi, Y.; Evans, D.; Hajishirzi, H. Do Membership Inference Attacks Work on Large Language Models? In Proceedings of the First Conference on Language Modeling, 2024.
10. Li, A.; Zhou, Y.; Raghuram, V.C.; Goldstein, T.; Goldblum, M. Commercial llm agents are already vulnerable to simple yet dangerous attacks. *arXiv preprint arXiv:2502.08586* **2025**.
11. Wu, C.; Zhang, Z.; Xu, M.; Wei, Z.; Sun, M. Monitoring LLM-based Multi-Agent Systems Against Corruptions via Node Evaluation. *arXiv preprint arXiv:2510.19420* **2025**.
12. Wang, S.; Zhu, T.; Liu, B.; Ding, M.; Ye, D.; Zhou, W.; Yu, P. Unique security and privacy threats of large language models: A comprehensive survey. *ACM Computing Surveys* **2025**, *58*, 1–36.

 doi:10.20944/preprints202602.1655.v2

13. Yu, M.; Meng, F.; Zhou, X.; Wang, S.; Mao, J.; Pan, L.; Chen, T.; Wang, K.; Li, X.; Zhang, Y.; et al. A survey on trustworthy llm agents: Threats and countermeasures. In Proceedings of the Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2, 2025, pp. 6216–6226.
14. Li, Y.; Wen, H.; Wang, W.; Li, X.; Yuan, Y.; Liu, G.; Liu, J.; Xu, W.; Wang, X.; Sun, Y.; et al. Personal llm agents: Insights and survey about the capability, efficiency and security. *arXiv preprint arXiv:2401.05459* **2024**.
15. Tang, X.; Jin, Q.; Zhu, K.; Yuan, T.; Zhang, Y.; Zhou, W.; Qu, M.; Zhao, Y.; Tang, J.; Zhang, Z.; et al. Prioritizing safeguarding over autonomy: Risks of llm agents for science. In Proceedings of the ICLR 2024 Workshop on Large Language Model (LLM) Agents, 2024.
16. Gan, Y.; Yang, Y.; Ma, Z.; He, P.; Zeng, R.; Wang, Y.; Li, Q.; Zhou, C.; Li, S.; Wang, T.; et al. Navigating the risks: A survey of security, privacy, and ethics threats in llm-based agents. *arXiv preprint arXiv:2411.09523* **2024**.
17. Chen, A.; Wu, Y.; Zhang, J.; Xiao, J.; Yang, S.; Huang, J.t.; Wang, K.; Wang, W.; Wang, S. A Survey on the Safety and Security Threats of Computer-Using Agents: JARVIS or Ultron? *arXiv preprint arXiv:2505.10924* **2025**.
18. Su, H.; Luo, J.; Liu, C.; Yang, X.; Zhang, Y.; Dong, Y.; Zhu, J. A Survey on Autonomy-Induced Security Risks in Large Model-Based Agents. *arXiv preprint arXiv:2506.23844* **2025**.
19. Wang, Y.; Pan, Y.; Su, Z.; Deng, Y.; Zhao, Q.; Du, L.; Luan, T.H.; Kang, J.; Niyato, D. Large model based agents: State-of-the-art, cooperation paradigms, security and privacy, and future trends. *IEEE Communications Surveys & Tutorials* **2025**.
20. Mohammadi, M.; Li, Y.; Lo, J.; Yip, W. Evaluation and benchmarking of llm agents: A survey. In Proceedings of the Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2, 2025, pp. 6129–6139.
21. Kong, D.; Lin, S.; Xu, Z.; Wang, Z.; Li, M.; Li, Y.; Zhang, Y.; Peng, H.; Sha, Z.; Li, Y.; et al. A Survey of LLM-Driven AI Agent Communication: Protocols. *Security Risks, and Defense Countermeasures* **2025**.
22. He, F.; Zhu, T.; Ye, D.; Liu, B.; Zhou, W.; Yu, P.S. The emerged security and privacy of llm agent: A survey with case studies. *ACM Computing Surveys* **2025**, *58*, 1–36.
23. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.R.; Cao, Y. React: Synergizing reasoning and acting in language models. In Proceedings of the The eleventh international conference on learning representations, 2022.
24. Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems* **2023**, *36*, 8634–8652.
25. Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; Lomeli, M.; Hambro, E.; Zettlemoyer, L.; Cancedda, N.; Scialom, T. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems* **2023**, *36*, 68539–68551.
26. Wang, X.; Chen, Y.; Yuan, L.; Zhang, Y.; Li, Y.; Peng, H.; Ji, H. Executable code actions elicit better llm agents. In Proceedings of the Forty-first International Conference on Machine Learning, 2024.
27. Hong, S.; Lin, Y.; Liu, B.; Liu, B.; Wu, B.; Zhang, C.; Li, D.; Chen, J.; Zhang, J.; Wang, J.; et al. Data interpreter: An llm agent for data science. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2025, 2025, pp. 19796–19821.
28. Zhou, S.; Xu, F.F.; Zhu, H.; Zhou, X.; Lo, R.; Sridhar, A.; Cheng, X.; Ou, T.; Bisk, Y.; Fried, D.; et al. WebArena: A Realistic Web Environment for Building Autonomous Agents. In Proceedings of the The Twelfth International Conference on Learning Representations, 2024.
29. Packer, C.; Fang, V.; Patil, S.; Lin, K.; Wooders, S.; Gonzalez, J. MemGPT: Towards LLMs as Operating Systems. **2023**.
30. Xu, W.; Mei, K.; Gao, H.; Tan, J.; Liang, Z.; Zhang, Y. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110* **2025**.
31. Xie, T.; Zhou, F.; Cheng, Z.; Shi, P.; Weng, L.; Liu, Y.; Hua, T.J.; Zhao, J.; Liu, Q.; Liu, C.; et al. OpenAgents: An Open Platform for Language Agents in the Wild. In Proceedings of the First Conference on Language Modeling, 2024.
32. Xu, C.; Kang, M.; Zhang, J.; Liao, Z.; Mo, L.; Yuan, M.; Sun, H.; Li, B. AdvAgent: Controllable Blackbox Red-teaming on Web Agents. In Proceedings of the Forty-second International Conference on Machine Learning, 2025.
33. Liao, Z.; Mo, L.; Xu, C.; Kang, M.; Zhang, J.; Xiao, C.; Tian, Y.; Li, B.; Sun, H. EIA: ENVIRONMENTAL INJECTION ATTACK ON GENERALIST WEB AGENTS FOR PRIVACY LEAKAGE. In Proceedings of the The Thirteenth International Conference on Learning Representations, 2025.

 doi:10.20944/preprints202602.1655.v2

34. Wu, F.; Wu, S.; Cao, Y.; Xiao, C. Wipi: A new web threat for llm-driven web agents. *arXiv preprint arXiv:2402.16965* **2024**.
35. Wang, Z.; Siu, V.; Ye, Z.; Shi, T.; Nie, Y.; Zhao, X.; Wang, C.; Guo, W.; Song, D. AgentVigil: Generic Black-Box Red-teaming for Indirect Prompt Injection against LLM Agents. *arXiv preprint arXiv:2505.05849* **2025**.
36. Fu, X.; Li, S.; Wang, Z.; Liu, Y.; Gupta, R.K.; Berg-Kirkpatrick, T.; Fernandes, E. Imprompter: Tricking llm agents into improper tool use. *arXiv preprint arXiv:2410.14923* **2024**.
37. Mo, K.; Hu, L.; Long, Y.; li, Z. Attractive Metadata Attack: Inducing LLM Agents to Invoke Malicious Tools. In Proceedings of the The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025.
38. Shi, J.; Yuan, Z.; Tie, G.; Zhou, P.; Gong, N.Z.; Sun, L. Prompt Injection Attack to Tool Selection in LLM Agents. *arXiv preprint arXiv:2504.19793* **2025**.
39. Zhang, J.; Yang, S.; Li, B. UDora: A Unified Red Teaming Framework against LLM Agents by Dynamically Hijacking Their Own Reasoning. In Proceedings of the Forty-second International Conference on Machine Learning, 2025.
40. Zhang, B.; Tan, Y.; Shen, Y.; Salem, A.; Backes, M.; Zannettou, S.; Zhang, Y. Breaking agents: Compromising autonomous llm agents through malfunction amplification. In Proceedings of the Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, 2025, pp. 34952–34964.
41. Yang, W.; Bi, X.; Lin, Y.; Chen, S.; Zhou, J.; Sun, X. Watch out for your agents! investigating backdoor threats to llm-based agents. *Advances in Neural Information Processing Systems* **2024**, *37*, 100938–100964.
42. Wang, B.; He, W.; Zeng, S.; Xiang, Z.; Xing, Y.; Tang, J.; He, P. Unveiling privacy risks in llm agent memory. In Proceedings of the Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2025, pp. 25241–25260.
43. Li, Z.; Cui, J.; Liao, X.; Xing, L. Les Dissonances: Cross-Tool Harvesting and Polluting in Pool-of-Tools Empowered LLM Agents.
44. Gu, X.; Zheng, X.; Pang, T.; Du, C.; Liu, Q.; Wang, Y.; Jiang, J.; Lin, M. Agent Smith: A Single Image Can Jailbreak One Million Multimodal LLM Agents Exponentially Fast. In Proceedings of the Proceedings of the 41st International Conference on Machine Learning (ICML). PMLR, July 2024, Vol. 235, *Proceedings of Machine Learning Research*, pp. 16647–16672.
45. Chen, Z.; Xiang, Z.; Xiao, C.; Song, D.; Li, B. AgentPoison: Red-teaming LLM Agents via Poisoning Memory or Knowledge Bases. In Proceedings of the The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024.
46. Dong, S.; Xu, S.; He, P.; Li, Y.; Tang, J.; Liu, T.; Liu, H.; Xiang, Z. Memory Injection Attacks on LLM Agents via Query-Only Interaction. In Proceedings of the The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025.
47. Jing, H.; Li, F.; Dong, Y.; Zhou, W.; Liu, R. Memory poisoning attacks on retrieval-augmented Large Language Model agents via deceptive semantic reasoning. *Engineering Applications of Artificial Intelligence* **2026**, *167*, 113968.
48. Wang, Y.; Xue, D.; Zhang, S.; Qian, S. BadAgent: Inserting and Activating Backdoor Attacks in LLM Agents. In Proceedings of the Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Bangkok, Thailand, 2024; pp. 9811–9827.
49. Liu, A.; Zhou, Y.; Liu, X.; Zhang, T.; Liang, S.; Wang, J.; Pu, Y.; Li, T.; Zhang, J.; Zhou, W.; et al. Compromising llm driven embodied agents with contextual backdoor attacks. *IEEE Transactions on Information Forensics and Security* **2025**.
50. Zhu, P.; Zhou, Z.; Zhang, Y.; Yan, S.; Wang, K.; Su, S. Demonagent: Dynamically encrypted multi-backdoor implantation attack on llm-based agent. *arXiv preprint arXiv:2502.12575* **2025**.
51. Lee, D.; Tiwari, M. Prompt infection: Llm-to-llm prompt injection within multi-agent systems. *arXiv preprint arXiv:2410.07283* **2024**.
52. Zhou, Z.; Li, Z.; Zhang, J.; Zhang, Y.; Wang, K.; Liu, Y.; Guo, Q. Corba: Contagious recursive blocking attacks on multi-agent systems based on large language models. *arXiv preprint arXiv:2502.14529* **2025**.
53. Shahroz, R.; Tan, Z.; Yun, S.; Fleming, C.; Chen, T. Agents under siege: Breaking pragmatic multi-agent llm systems with optimized prompt attacks. In Proceedings of the Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2025, pp. 9661–9674.
54. Ju, T.; Wang, Y.; Ma, X.; Cheng, P.; Zhao, H.; Wang, Y.; Liu, L.; Xie, J.; Zhang, Z.; Liu, G. Flooding spread of manipulated knowledge in llm-based multi-agent communities. *arXiv preprint arXiv:2407.07791* **2024**.

 doi:10.20944/preprints202602.1655.v2

55. He, P.; Lin, Y.; Dong, S.; Xu, H.; Xing, Y.; Liu, H. Red-teaming llm multi-agent systems via communication attacks. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2025, 2025, pp. 6726–6747.
56. Yan, B.; Zhou, Z.; Zhang, X.; Li, C.; Zeng, R.; Qi, Y.; Wang, T.; Zhang, L. Attack the Messages, Not the Agents: A Multi-round Adaptive Stealthy Tampering Framework for LLM-MAS. *arXiv preprint arXiv:2508.03125* **2025**.
57. Triedman, H.; Jha, R.; Shmatikov, V. Multi-agent systems execute arbitrary malicious code. *arXiv preprint arXiv:2503.12188* **2025**.
58. Motwani, S.R.; Baranchuk, M.; Strohmeier, M.; Bolina, V.; Torr, P.H.; Hammond, L.; de Witt, C.S. Secret Collusion among AI Agents: Multi-Agent Deception via Steganography. In Proceedings of the Advances in Neural Information Processing Systems, 2024, Vol. 37, pp. 73439–73486.
59. Tian, Y.; Yang, X.; Zhang, J.; Dong, Y.; Su, H. Evil geniuses: Delving into the safety of llm-based agents. *arXiv preprint arXiv:2311.11855* **2023**.
60. tse Huang, J.; Zhou, J.; Jin, T.; Zhou, X.; Chen, Z.; Wang, W.; Yuan, Y.; Lyu, M.; Sap, M. On the Resilience of LLM-Based Multi-Agent Collaboration with Faulty Agents. In Proceedings of the Forty-second International Conference on Machine Learning, 2025.
61. Xie, Y.; Zhu, C.; Zhang, X.; Zhu, T.; Ye, D.; Wang, M.; Liu, C. Who's the Mole? Modeling and Detecting Intention-Hiding Malicious Agents in LLM-Based Multi-Agent Systems. *arXiv preprint arXiv:2507.04724* **2025**.
62. Wang, L.; Wang, W.; Wang, S.; Li, Z.; Ji, Z.; Lyu, Z.; Wu, D.; Cheung, S.C. Ip leakage attacks targeting llm-based multi-agent systems. *arXiv preprint arXiv:2505.12442* **2025**.
63. Shi, T.; Zhu, K.; Wang, Z.; Jia, Y.; Cai, W.; Liang, W.; Wang, H.; Alzahrani, H.; Lu, J.; Kawaguchi, K.; et al. Promptarmor: Simple yet effective prompt injection defenses. *arXiv preprint arXiv:2507.15219* **2025**.
64. Wang, Z.; Nagaraja, N.; Zhang, L.; Bahsi, H.; Patil, P.; Liu, P. To Protect the LLM Agent Against the Prompt Injection Attack with Polymorphic Prompt. In Proceedings of the 2025 55th Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S), 2025, pp. 22–28.
65. Wu, Y.; Roesner, F.; Kohno, T.; Zhang, N.; Iqbal, U. IsolateGPT: An Execution Isolation Architecture for LLM-Based Agentic Systems. In Proceedings of the 32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025. The Internet Society, 2025.
66. Bagdasarian, E.; Yi, R.; Ghalebikesabi, S.; Kairouz, P.; Gruteser, M.; Oh, S.; Balle, B.; Ramage, D. AirGapAgent: Protecting Privacy-Conscious Conversational Agents. In Proceedings of the Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, 2024, p. 3868–3882.
67. Foerster, H.; Mullins, R.; Blanchard, T.; Papernot, N.; Nikolić, K.; Tramèr, F.; Shumailov, I.; Zhang, C.; Zhao, Y. CaMeLs Can Use Computers Too: System-level Security for Computer Use Agents. *arXiv preprint arXiv:2601.09923* **2026**.
68. Jia, F.; Wu, T.; Qin, X.; Squicciarini, A. The task shield: Enforcing task alignment to defend against indirect prompt injection in llm agents. In Proceedings of the Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2025, pp. 29680–29697.
69. Zhu, K.; Yang, X.; Wang, J.; Guo, W.; Wang, W.Y. MELON: Provable Defense Against Indirect Prompt Injection Attacks in AI Agents. In Proceedings of the Proceedings of the 42nd International Conference on Machine Learning (ICML). PMLR, 2025, Vol. 267, *Proceedings of Machine Learning Research*, pp. 80310–80329.
70. Chen, Z.; Kang, M.; Li, B. ShieldAgent: Shielding Agents via Verifiable Safety Policy Reasoning. In Proceedings of the Forty-second International Conference on Machine Learning, 2025.
71. An, H.; Zhang, J.; Du, T.; Zhou, C.; Li, Q.; Lin, T.; Ji, S. Ipiguard: A novel tool dependency graph-based defense against indirect prompt injection in llm agents. In Proceedings of the Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, 2025, pp. 1023–1039.
72. Li, H.; Liu, X.; Chiu, H.C.; Li, D.; Zhang, N.; Xiao, C. DRIFT: Dynamic Rule-Based Defense with Injection Isolation for Securing LLM Agents. *arXiv preprint arXiv:2506.12104* **2025**.
73. Xiang, S.; Zhang, T.; Chen, R. ALRPHFS: Adversarially Learned Risk Patterns with Hierarchical Fast\&Slow Reasoning for Robust Agent Defense. *arXiv preprint arXiv:2505.19260* **2025**.
74. Changjiang, L.; Jiacheng, L.; Bochuan, C.; Jinghui, C.; Ting, W. Your Agent Can Defend Itself against Backdoor Attacks. *arXiv preprint arXiv:2506.08336* **2025**.
75. Feng, E.; Zhou, W.; Liu, Z.; Chen, L.; Dong, Y.; Zhang, C.; Zhao, Y.; Du, D.; Hua, Z.; Xia, Y.; et al. Get Experience from Practice: LLM Agents with Record & Replay. *arXiv preprint arXiv:2505.17716* **2025**.

 doi:10.20944/preprints202602.1655.v2

76. Bonagiri, V.K.; Kumaragurum, P.; Nguyen, K.; Plaut, B. Check Yourself Before You Wreck Yourself: Selectively Quitting Improves LLM Agent Safety. *arXiv preprint arXiv:2510.16492* **2025**.
77. Wei, Q.; Yang, T.; Wang, Y.; Li, X.; Li, L.; Yin, Z.; Zhan, Y.; Holz, T.; Lin, Z.; Wang, X. A-MemGuard: A Proactive Defense Framework for LLM-Based Agent Memory. *arXiv preprint arXiv:2510.02373* **2025**.
78. Mao, J.; Meng, F.; Duan, Y.; Yu, M.; Jia, X.; Fang, J.; Liang, Y.; Wang, K.; Wen, Q. Agentsafe: Safeguarding large language model-based multi-agent systems via hierarchical data management. *arXiv preprint arXiv:2503.04392* **2025**.
79. Sunil, B.D.; Sinha, I.; Maheshwari, P.; Todmal, S.; Malik, S.; Mishra, S. Memory Poisoning Attack and Defense on Memory Based LLM-Agents. *arXiv preprint arXiv:2601.05504* **2026**.
80. Pan, Z.; Zhang, Y.; Liu, Z.; Tang, Y.Y.; Zhang, Z.; Luo, H.; Han, Y.; Zhang, J.; Wu, D.; Chen, H.Y.; et al. AdvEvo-MARL: Shaping Internalized Safety through Adversarial Co-Evolution in Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2510.01586* **2025**.
81. Patlan, A.S.; Sheng, P.; Hebbar, S.A.; Mittal, P.; Viswanath, P. Real ai agents with fake memories: Fatal context manipulation attacks on web3 agents. *arXiv preprint arXiv:2503.16248* **2025**.
82. Wang, S.; Zhang, G.; Yu, M.; Wan, G.; Meng, F.; Guo, C.; Wang, K.; Wang, Y. G-Safeguard: A Topology-Guided Security Lens and Treatment on LLM-based Multi-agent Systems. In Proceedings of the Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2025, pp. 7261–7276.
83. Zhou, J.; Wang, L.; Yang, X. GUARDIAN: Safeguarding LLM Multi-Agent Collaborations with Temporal Graph Modeling. *arXiv preprint arXiv:2505.19234* **2025**.
84. Miao, R.; Liu, Y.; Wang, Y.; Shen, X.; Tan, Y.; Dai, Y.; Pan, S.; Wang, X. Blindguard: Safeguarding llm-based multi-agent systems under unknown attacks. *arXiv preprint arXiv:2508.08127* **2025**.
85. Zhou, Y.; Lu, X.; Liu, D.; Yan, J.; Shao, J. INFA-Guard: Mitigating Malicious Propagation via Infection-Aware Safeguarding in LLM-Based Multi-Agent Systems. *arXiv preprint arXiv:2601.14667* **2026**.
86. Chen, K.; Zhen, T.; Wang, H.; Liu, K.; Li, X.; Huo, J.; Yang, T.; Xu, J.; Dong, W.; Gao, Y. MedSentry: Understanding and Mitigating Safety Risks in Medical LLM Multi-Agent Systems. *arXiv preprint arXiv:2505.20824* **2025**.
87. Fan, F.; Li, X. PeerGuard: Defending Multi-Agent Systems Against Backdoor Attacks Through Mutual Reasoning. In Proceedings of the 2025 IEEE International Conference on Information Reuse and Integration and Data Science (IRI), 2025, pp. 234–239.
88. HU, J.; DONG, Y.; DING, Z.; HUANG, X. Enhancing robustness of LLM-driven multi-agent systems through randomized smoothing. *Chinese Journal of Aeronautics* **2025**, p. 103779.
89. Wen, Y.; Guo, J.; Huang, H. CoTGuard: Using Chain-of-Thought Triggering for Copyright Protection in Multi-Agent LLM Systems. *arXiv preprint arXiv:2505.19405* **2025**.
90. Mei, K.; Zhu, X.; Xu, W.; Hua, W.; Jin, M.; Li, Z.; Xu, S.; Ye, R.; Ge, Y.; Zhang, Y. Aios: Llm agent operating system. *arXiv preprint arXiv:2403.16971* **2024**.
91. Bagdasarian, E.; Yi, R.; Ghalebikesabi, S.; Kairouz, P.; Gruteser, M.; Oh, S.; Balle, B.; Ramage, D. Airgapagent: Protecting privacy-conscious conversational agents. In Proceedings of the Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, 2024, pp. 3868–3882.
92. He, Y.; Wang, E.; Rong, Y.; Cheng, Z.; Chen, H. Security of ai agents. In Proceedings of the 2025 IEEE/ACM International Workshop on Responsible AI Engineering (RAIE). IEEE, 2025, pp. 45–52.
93. Hua, W.; Yang, X.; Jin, M.; Li, Z.; Cheng, W.; Tang, R.; Zhang, Y. TrustAgent: Towards Safe and Trustworthy LLM-based Agents through Agent Constitution. In Proceedings of the Trustworthy Multi-modal Foundation Models and AI Agents (TiFA), 2024.
94. Zhang, Y.; Cai, Y.; Zuo, X.; Luan, X.; Wang, K.; Hou, Z.; Zhang, Y.; Wei, Z.; Sun, M.; Sun, J.; et al. Position: Trustworthy AI Agents Require the Integration of Large Language Models and Formal Methods. In Proceedings of the Forty-second International Conference on Machine Learning Position Paper Track, 2025.
95. Rosser, J.; Foerster, J.N. AgentBreeder: Mitigating the AI Safety Impact of Multi-Agent Scaffolds via Self-Improvement. In Proceedings of the Scaling Self-Improving Foundation Models without Human Supervision, 2025.
96. Hua, Y.; Chen, H.; Wang, S.; Li, W.; Wang, X.; Luo, J. Shapley-Coop: Credit Assignment for Emergent Cooperation in Self-Interested LLM Agents. *arXiv preprint arXiv:2506.07388* **2025**.
97. Narajala, V.S.; Narayan, O. Securing agentic ai: A comprehensive threat model and mitigation framework for generative ai agents. *arXiv preprint arXiv:2504.19956* **2025**.

 doi:10.20944/preprints202602.1655.v2

98. Gosmar, D.; Dahl, D.A. Sentinel Agents for Secure and Trustworthy Agentic AI in Multi-Agent Systems. *arXiv preprint arXiv:2509.14956* **2025**.
99. Chennabasappa, S.; Nikolaidis, C.; Song, D.; Molnar, D.; Ding, S.; Wan, S.; Whitman, S.; Deason, L.; Doucette, N.; Montilla, A.; et al. Llamafirewall: An open source guardrail system for building secure ai agents. *arXiv preprint arXiv:2505.03574* **2025**.
100. Andriushchenko, M.; Souly, A.; Dziemian, M.; Duenas, D.; Lin, M.; Wang, J.; Hendrycks, D.; Zou, A.; Kolter, J.Z.; Fredrikson, M.; et al. AgentHarm: A Benchmark for Measuring Harmfulness of LLM Agents. In Proceedings of the The Thirteenth International Conference on Learning Representations, 2025.
101. Vijayvargiya, S.; Soni, A.B.; Zhou, X.; Wang, Z.Z.; Dziri, N.; Neubig, G.; Sap, M. OpenAgentSafety: A Comprehensive Framework for Evaluating Real-World AI Agent Safety, 2025, [2507.06134].
102. Luo, H.; Dai, S.; Ni, C.; Li, X.; Zhang, G.; Wang, K.; Liu, T.; Salam, H. Agentauditor: Human-level safety and security evaluation for llm agents. *arXiv preprint arXiv:2506.00641* **2025**.
103. Wang, H.; Zhang, A.; Duy Tai, N.; Sun, J.; Chua, T.S.; et al. Ali-agent: Assessing llms' alignment with human values via agent-based evaluation. *Advances in Neural Information Processing Systems* **2024**, *37*, 99040–99088.
104. Ruan, Y.; Dong, H.; Wang, A.; Pitis, S.; Zhou, Y.; Ba, J.; Dubois, Y.; Maddison, C.J.; Hashimoto, T. Identifying the Risks of LM Agents with an LM-Emulated Sandbox. In Proceedings of the The Twelfth International Conference on Learning Representations, 2024.
105. Debenedetti, E.; Zhang, J.; Balunovic, M.; Beurer-Kellner, L.; Fischer, M.; Tramèr, F. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. *Advances in Neural Information Processing Systems* **2024**, *37*, 82895–82920.
106. Milev, I.; Balunović, M.; Baader, M.; Vechev, M. ToolFuzz–Automated Agent Tool Testing. *arXiv preprint arXiv:2503.04479* **2025**.
107. Zhang, H.; Huang, J.; Mei, K.; Yao, Y.; Wang, Z.; Zhan, C.; Wang, H.; Zhang, Y. Agent Security Bench (ASB): Formalizing and Benchmarking Attacks and Defenses in LLM-based Agents. In Proceedings of the The Thirteenth International Conference on Learning Representations, 2025.
108. Evtimov, I.; Zharmagambetov, A.; Grattafiori, A.; Guo, C.; Chaudhuri, K. WASP: Benchmarking Web Agent Security Against Prompt Injection Attacks. In Proceedings of the ICML 2025 Workshop on Computer Use Agents, 2025.
109. Wu, C.H.; Shah, R.R.; Koh, J.Y.; Salakhutdinov, R.; Fried, D.; Raghunathan, A. Dissecting Adversarial Robustness of Multimodal LM Agents. In Proceedings of the The Thirteenth International Conference on Learning Representations, 2025.
110. Guo, C.; Liu, X.; Xie, C.; Zhou, A.; Zeng, Y.; Lin, Z.; Song, D.; Li, B. Redcode: Risky code execution and generation benchmark for code agents. *Advances in Neural Information Processing Systems* **2024**, *37*, 106190–106236.
111. Saha, S.; Chen, J.; Mayers, S.; Gouda, S.K.; Wang, Z.; Kumar, V. Breaking the code: Security assessment of ai code agents through systematic jailbreaking attacks. *arXiv preprint arXiv:2510.01359* **2025**.
112. Zhu, Y.; Kellermann, A.; Bowman, D.; Li, P.; Gupta, A.; Danda, A.; Fang, R.; Jensen, C.; Ihli, E.; Benn, J.; et al. CVE-Bench: A Benchmark for AI Agents' Ability to Exploit Real-World Web Application Vulnerabilities. In Proceedings of the Proceedings of the 42nd International Conference on Machine Learning (ICML). PMLR, 2025, Vol. 267, *Proceedings of Machine Learning Research*, pp. 79850–79867.
113. Tur, A.D.; Meade, N.; Lù, X.H.; Zambrano, A.; Patel, A.; DURMUS, E.; Gella, S.; Stanczak, K.; Reddy, S. SafeArena: Evaluating the Safety of Autonomous Web Agents. In Proceedings of the Forty-second International Conference on Machine Learning, 2025.
114. Zharmagambetov, A.; Guo, C.; Evtimov, I.; Pavlova, M.; Salakhutdinov, R.; Chaudhuri, K. AgentDAM: Privacy Leakage Evaluation for Autonomous Web Agents. In Proceedings of the Proceedings of the 39th Annual Conference on Neural Information Processing Systems (NeurIPS 2025) Datasets and Benchmarks Track, 2025.