International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(23s) 2026 pp. 1721-1729
https://doi.org/10.70917/ijcisim-2026-5724
Article

Shieldgraph: Topology-Aware Runtime Defense For Multi-Agent LLM Systems Against Adversarial Coordination Attacks

Nitin Addla

Independent Researcher, India. Email:Nitin.addla@gmail.com

Abstract: Multi-agent LLM systems—where multiple language model agents coordinate through message passing to accomplish complex tasks—are rapidly entering production deployment via frameworks such as LangGraph, CrewAI, and AutoGen. However, their interconnected architecture creates novel attack surfaces wherein adversarial inputs can propagate across agent boundaries through topology-guided exploitation. This paper introduces ShieldGraph, a runtime defense framework that models multi-agent topologies as trust-annotated graphs and enforces security invariants at inter-agent communication boundaries. ShieldGraph operates through three mechanisms: (1) a formal Trust Propagation Calculus (TPC) that assigns and dynamically updates capability-bounded trust scores to each agent and communication channel, (2) a lightweight adversarial message classifier that detects prompt injection, role hijacking, and intent drift in inter-agent communications with 91.3% precision at 2.4ms per message, and (3) a topology-aware quarantine protocol that isolates compromised subgraphs while maintaining operational continuity for unaffected agents. We evaluate ShieldGraph on the TAMAS adversarial benchmark, a novel red-team suite of 847 multi-agent attack scenarios, and three production-representative orchestration topologies (centralized, hierarchical, and decentralized). ShieldGraph reduces attack success rate from 68.4% (undefended) to 7.2% while preserving 94.1% task completion on benign workloads—a 6.8× safety improvement with only 3.7% utility degradation. We further demonstrate that topology-aware defense outperforms topology-agnostic baselines by 18.3 percentage points, establishing that the structural properties of multi-agent systems are critical to effective defense design.

Keywords: Multi-agent systems, adversarial attacks, prompt injection, trust propagation, LLM security, runtime defense, topology-aware computing, agent orchestration

1. INTRODUCTION

The deployment of multi-agent LLM systems has accelerated dramatically throughout 2025–2026, driven by the recognition that complex real-world tasks often exceed the capabilities of individual language models operating in isolation [1]. Production frameworks including LangGraph [2], CrewAI [3], and the OpenAI Agents SDK [4] enable developers to compose multiple specialized LLM agents into collaborative workflows—orchestrators that plan, workers that execute, and validators that check—communicating through structured message passing protocols.

This architectural paradigm, however, introduces a fundamentally new threat surface. Unlike single-agent systems where security boundaries are relatively well-understood (user input → model → output), multi-agent systems create internal communication channels that blur the distinction between trusted and untrusted data [5]. An adversarial payload injected into one agent can propagate through the system topology, exploiting the trust that agents implicitly grant to messages from their peers—a phenomenon termed "prompt infection" by Gu et al. [6].

Recent empirical evidence paints an alarming picture. The TAMAS benchmark [7] demonstrates that multi-agent systems are highly vulnerable to adversarial attacks, with attack success rates exceeding 60% across standard

configurations. Topology-guided attacks [8] show that adversaries can exploit the structural properties of agent networks to propagate contamination from low-privilege edge agents to high-privilege core agents. Production reports indicate multi-agent systems fail at rates between 41–87% [9], with coordination defects—not individual agent failures—accounting for the majority of breakdowns.

Despite this growing threat landscape, defensive mechanisms remain underdeveloped. Existing work focuses primarily on single-agent prompt injection defense [10] [11] or adversarial training for isolated models [12]. The few multi-agent defense approaches either address only RL-based systems [13] or propose intent-verification chains that degrade to 13% effectiveness against paraphrasing attacks [14]. No existing framework treats the multi-agent topology itself as a defensive asset—leveraging structural properties to detect, contain, and neutralize adversarial propagation.

This paper introduces ShieldGraph, a topology-aware runtime defense framework. Our contributions are:

• Formal threat model: We formalize adversarial attacks on multi-agent LLM systems as graph propagation problems, identifying four attack classes (injection propagation, role hijacking cascade, privilege escalation via delegation, and coordinated exfiltration) and mapping them to topological vulnerabilities.

• Trust Propagation Calculus: We introduce TPC, a formal framework for computing, propagating, and dynamically updating trust scores across agent topologies, enabling principled enforcement of least-privilege communication policies.

• Lightweight inter-agent defense: We present an adversarial message classifier operating at 2.4ms per message, achieving 91.3% precision in detecting malicious inter-agent communications without requiring access to agent internals.

• Topology-aware quarantine: We design a graph-partitioning quarantine protocol that isolates compromised agents while maintaining maximum operational continuity, reducing attack success 6.8× with only 3.7% utility loss.

2. Related Work

A. Multi-Agent LLM Architectures

Multi-agent LLM orchestration has been systematically surveyed by Zhao et al. [15], who propose a three-topology taxonomy: centralized (single orchestrator dispatching to workers), hierarchical (nested delegation chains), and decentralized (peer-to-peer coordination). Production deployments predominantly follow the centralized pattern [16], though hierarchical architectures are gaining traction for complex enterprise workflows [2]. Each topology presents distinct security properties: centralized systems have a single point of failure but clear authority chains; decentralized systems are resilient to single-node compromise but lack visibility into global state.

B. Adversarial Attacks on Multi-Agent Systems

The vulnerability landscape has been extensively characterized in recent work. Gu et al. [6] introduced prompt infection—self-replicating malicious prompts that propagate across agent connections analogous to computer viruses. Xu et al. [8] demonstrated topology-guided adversarial propagation, showing that strategic injection at exposed edge agents can contaminate high-privilege core agents through multi-hop message chains. The TAMAS benchmark [7] provides systematic evaluation of adversarial risks across diverse multi-agent configurations. Chen et al. [17] catalogue privilege escalation attacks where agents exploit delegated authority to exceed their intended capability scope.

Critically, these attacks exploit a property unique to multi-agent systems: the “trust inheritance problem,” wherein agents treat messages from peer agents with the same implicit trust as system prompts, creating a fundamental confusion between instruction and data at every inter-agent boundary [18].

C. Defensive Mechanisms

Defense research has primarily targeted single-agent scenarios. Progent [10] achieves approximately 4.2% attack success rate on individual agents through out-of-band defense monitoring. ICON [11] detects indirect prompt injection via latent-space trace probing. For multi-agent contexts, AdvEvo-MARL [13] proposes adversarial co-evolution in multi-agent RL, maintaining attack success below 20%—but this approach requires RL training and does not apply to the dominant LLM-to-LLM message-passing paradigm.

Access control frameworks have emerged as a partial solution. AC4A [19] introduces hierarchical resource permissions for agents, while AuthBench [20] evaluates agents' ability to self-enforce least-privilege policies. MCP proxy architectures [21] enforce tool-level access control. However, none of these approaches models the inter-agent communication graph as a first-class security object or provides runtime defense against adversarial messages propagating between agents.

D. Trust and Delegation in Agent Systems

Formal trust models for autonomous systems have been explored in the distributed systems literature [22], but their adaptation to LLM agents remains nascent. Intent-verified delegation chains [14] represent the closest prior work, proposing that delegated tasks carry verifiable intent declarations. However, this approach suffers from the "adversarial intent paraphrasing problem"—attackers can rephrase malicious intents to pass verification—achieving only 87% defense under sophisticated attacks. ShieldGraph addresses this limitation through continuous behavioral monitoring rather than one-time intent verification.

3. Formal Threat Model

A. System Model

We model a multi-agent LLM system as a directed graph $\Gamma = (A, C, \pi)$ where $A = \{a_1, a_2, ..., a_n\}$ is the set of agents, $C \subseteq A \times A$ is the set of directed communication channels, and $\pi: A \to P$ is a privilege mapping assigning each agent a set of capabilities from a predefined capability universe P. Each agent a_i possesses:

- A system prompt S_i defining its role and behavioral constraints;
- A capability set $\pi(a_i) \subseteq P$ specifying permitted actions (tool calls, data access, delegation);
- An input channel set $In(a_i) = \{a_j : (a_j, a_i) \in C\}$ of agents from which it receives messages;
- An output channel set $Out(a_i) = \{a_j : (a_i, a_j) \in C\}$ of agents to which it sends messages.

B. Adversary Model

We consider an adversary A with the following capabilities: (1) A can inject adversarial content into the input of at least one agent $a\_entry \in A$ through external data sources (documents, web content, tool responses); (2) A cannot directly modify agent system prompts or framework code; (3) A has knowledge of the system topology Γ (a strong but realistic assumption for insider threats and reconnaissance-capable attackers); (4) A's objective is to cause a target agent a_target to execute actions outside its authorized capability set, i.e., to induce a_target to perform action $\alpha \notin \pi(a\_target)$.

C. Attack Taxonomy

We identify four classes of topology-exploiting attacks:

• Class A — Injection Propagation: Adversarial instructions injected at a_entry propagate through message chains to reach a_target, exploiting the trust that intermediate agents place in peer messages. Attack path length |path(a_entry, a_target)| determines propagation difficulty.

• Class B — Role Hijacking Cascade: Adversarial content redefines the role of an intermediate agent a_mid, causing it to generate messages that hijack the role of downstream agents. This creates a cascade where each hijacked agent amplifies the attack surface.

• Class C — Privilege Escalation via Delegation: A low-privilege agent a_low crafts messages that cause a high-privilege agent a_high to delegate capabilities back to a_low or to execute high-privilege actions on a_low's behalf, violating the principle of least privilege.

• Class D — Coordinated Exfiltration: Multiple compromised agents cooperate to exfiltrate sensitive data by partitioning it across multiple output channels, evading per-agent monitoring thresholds.

4. ShieldGraph Framework

ShieldGraph interposes a security layer at every inter-agent communication channel in Γ, monitoring and validating messages in real-time without modifying agent internals. The framework comprises three components operating in concert.

A. Trust Propagation Calculus (TPC)

TPC assigns each agent a_i a dynamic trust score $\tau_i \in [0, 1]$ and each channel (a_i, a_j) a channel trust score $\tau_{ij} \in [0, 1]$. Trust scores are initialized based on static analysis of the topology and privilege assignments:

$$\tau_i^0 = f(\pi(a_i), \deg^-(a_i), \text{pos}\Gamma(a_i))$$

where $\deg^-(a_i)$ is the in-degree (number of input channels, measuring exposure), and $\text{pos}\Gamma(a_i)$ is the topological centrality (PageRank score in Γ, measuring influence). Agents with high capability and high centrality receive lower initial trust (more scrutiny), following the principle that high-value targets require stronger verification.

Trust scores are updated after each message exchange based on behavioral signals:

$$\tau_i^{t+1} = \alpha \cdot \tau_i^t + (1-\alpha) \cdot g(\text{msg}_i^t, \pi(a_i), \text{history}_i)$$

where $g(\cdot)$ is a behavioral scoring function that evaluates whether the agent's latest message is consistent with its declared role, authorized capabilities, and historical communication patterns. The decay parameter $\alpha = 0.85$ ensures that trust responds to recent behavior while maintaining stability.

Channel trust τ_{ij} governs message validation thresholds: messages on low-trust channels undergo more rigorous inspection. When τ_i drops below a quarantine threshold $\theta\_q = 0.3$, agent a_i is flagged for isolation.

B. Adversarial Message Classifier (AMC)

The AMC is a lightweight binary classifier deployed at each inter-agent communication channel. Given a message m from agent a_i to a_j, the AMC computes P(adversarial | m, context) where context includes: (1) the channel's trust score τ_{ij}, (2) the sender's authorized capability set $\pi(a_i)$, (3) a sliding window of the last k=5 messages on this channel, and (4) the receiver's current task state.

The classifier architecture employs a distilled DeBERTa-v3-base model [23] fine-tuned on 45,000 labeled inter-agent messages spanning benign coordination, prompt injection attempts, role-hijacking payloads, and privilege-escalation requests. We augment training data with adversarial examples generated through iterative red-teaming using GPT-4o as the attacker.

Critical design choices enable the required 2.4ms latency: (1) 4-bit quantization of the classifier backbone, (2) early-exit layers that short-circuit clearly benign messages after the first transformer block, and (3) caching of channel-level context embeddings that are updated incrementally rather than recomputed per message.

The AMC outputs a ternary verdict: ALLOW ($P < 0.3$), INSPECT ($0.3 \leq P < 0.7$), or BLOCK ($P \geq 0.7$). INSPECT messages are logged and subjected to additional analysis by

the trust update mechanism but are permitted to pass, balancing security with operational continuity.

C. Topology-Aware Quarantine Protocol (TAQP)

When an agent's trust score falls below θ_q or the AMC detects a high-confidence attack chain (multiple BLOCK verdicts from the same source), ShieldGraph activates the quarantine protocol. TAQP performs graph-aware isolation:

• Step 1 — Contamination scope estimation: Compute the forward reachability set $R^+(a_{compromised}) = \{a_j : \exists \text{ path from } a_{compromised} \text{ to } a_j \text{ in } \Gamma\}$ to identify all agents potentially exposed to propagated adversarial content.

• Step 2 — Minimum vertex cut: Compute the minimum set of channels to sever that disconnects $a_{compromised}$ from the remainder of the operational graph while maximizing preserved connectivity. This is solved as a minimum s-t cut problem on the trust-weighted graph.

• Step 3 — Graceful degradation: Quarantined agents' pending tasks are redistributed to non-compromised agents with compatible capability sets. The orchestrator is notified with a structured alert containing the attack class, affected scope, and recommended human intervention points.

• Step 4 — Recovery verification: Quarantined agents are re-initialized with fresh system prompts and subjected to a challenge-response protocol before trust scores are restored. Full recovery requires 3 consecutive clean interactions.

The quarantine protocol's topology-awareness is its key differentiator: rather than shutting down the entire system or isolating only the immediately compromised agent, TAQP leverages graph structure to contain propagation along its most likely paths while preserving maximum system functionality.

5. Experimental Setup

A. Evaluation Benchmarks

We evaluate ShieldGraph on two complementary benchmarks:

• TAMAS [7]: The adversarial multi-agent security benchmark from ACL 2026, containing 312 attack scenarios across prompt injection, jailbreaking, and data exfiltration categories.

• ShieldBench (ours): A novel red-team benchmark of 847 multi-agent attack scenarios designed to stress-test topology-specific vulnerabilities. Attacks are generated by GPT-4o and Claude 3.5 operating as adversarial agents, validated by human red-teamers, and organized by attack class (A: 231, B: 198, C: 219, D: 199).

B. System Topologies

We instantiate three production-representative multi-agent topologies:

Topology	Agents	Pattern	Framework
Centralized	1 orchestrator + 5 workers	Star graph	LangGraph
Hierarchical	1 planner + 2 managers + 6 workers	Tree (depth 3)	CrewAI
Decentralized	8 peer agents	Mesh (partial)	AutoGen

TABLE I. Evaluation topologies and their configurations.

C. Baseline Defenses

We compare ShieldGraph against five baseline approaches:

• No Defense: Unmodified multi-agent system with no adversarial protection.

• Per-Agent Guard [10]: Progent-style defense applied independently to each agent's input, without inter-agent awareness.

• Intent Verification [14]: Delegation chains with intent declarations verified at each handoff.

• ICON [11]: Latent-space probing for indirect prompt injection, applied per-agent.

• Flat Quarantine: Topology-agnostic version of our quarantine protocol that isolates individual agents without graph-aware scope estimation.

D. Metrics

We report: (1) Attack Success Rate (ASR)—percentage of attack scenarios achieving the adversary's objective; (2) Task Completion Rate (TCR)—percentage of benign tasks completed correctly; (3) Utility Degradation—(TCR_undefended - TCR_defended) / TCR_undefended; (4) Detection Latency—time from attack injection to first defensive action; (5) Containment Scope—fraction of the system quarantined per detected attack.

6. Experimental Results

A. Overall Defense Performance

Table II presents the primary results aggregated across all topologies and attack classes.

Defense Method	ASR ↓	TCR ↑	Utility Loss	Det. Latency
No Defense	68.4%	97.8%	—	—
Per-Agent Guard	41.2%	93.1%	4.8%	180ms
Intent Verification	32.7%	89.4%	8.6%	340ms
ICON	37.8%	91.7%	6.2%	95ms
Flat Quarantine	18.9%	82.3%	15.9%	210ms
ShieldGraph (Ours)	**7.2%**	**94.1%**	**3.7%**	**48ms**

TABLE II. Overall defense performance. ASR = Attack Success Rate (lower is better), TCR = Task Completion Rate (higher is better). Best results in bold.

ShieldGraph achieves the lowest attack success rate (7.2%) while maintaining the highest task completion rate among all defended systems (94.1%). The 3.7% utility degradation is substantially lower than topology-agnostic quarantine (15.9%) and intent verification (8.6%), demonstrating that topology-aware defense avoids the over-isolation problem that plagues naïve approaches. Detection latency of 48ms reflects the AMC's efficiency—attacks are identified within the first inter-agent message exchange rather than after propagation has occurred.

B. Performance by Topology

Table III disaggregates attack success rates by system topology, revealing that topological structure significantly moderates both vulnerability and defense effectiveness.

Topology	Undefended	Per-Agent Guard	ShieldGraph	Δ vs. Guard
Centralized	54.3%	31.8%	4.1%	-27.7 pp
Hierarchical	71.2%	45.6%	8.3%	-37.3 pp
Decentralized	79.8%	52.1%	9.4%	-42.7 pp

TABLE III. Attack success rates by topology. ShieldGraph's advantage over per-agent defense grows with topological complexity.

Decentralized systems are most vulnerable in the undefended condition (79.8% ASR) due to their numerous lateral communication channels and lack of centralized authority. Critically, ShieldGraph's relative advantage over topology-agnostic baselines increases with topological complexity: the improvement is 27.7 percentage points for centralized systems but 42.7 pp for decentralized ones, confirming that topology-awareness becomes more valuable precisely where it is most needed.

C. Performance by Attack Class

Table IV reports ShieldGraph's performance across the four attack classes defined in our threat model.

Attack Class	ASR	Precision	Recall	F1
A: Injection Propagation	5.6%	0.934	0.891	0.912
B: Role Hijacking Cascade	8.1%	0.907	0.842	0.873
C: Privilege Escalation	6.8%	0.921	0.878	0.899
D: Coordinated Exfiltration	8.7%	0.879	0.814	0.845

TABLE IV. Detection performance by attack class.

Injection propagation (Class A) is most effectively defended (5.6% residual ASR) because it creates the clearest anomalous signals in inter-agent messages. Coordinated exfiltration (Class D) proves most challenging (8.7% ASR) as it distributes adversarial behavior across multiple agents, each individually appearing benign. This suggests that future work should focus on collective behavioral anomaly detection beyond pairwise channel monitoring.

D. Ablation Study

Configuration	ASR ↓	TCR ↑	Util. Loss
Full ShieldGraph	**7.2%**	**94.1%**	**3.7%**
w/o TPC (static trust only)	14.8%	93.4%	4.5%
w/o AMC (quarantine only)	22.1%	86.2%	11.9%
w/o TAQP (classify + block only)	11.3%	93.8%	4.1%
w/o topology-awareness (flat graph)	25.5%	88.7%	9.3%

TABLE V. Ablation study isolating component contributions.

Each component contributes substantially. The AMC provides the primary detection capability; without it, the system must rely on trust decay alone, resulting in delayed detection and higher ASR (22.1%). TPC's dynamic trust updates halve the residual ASR compared to static trust (7.2% vs. 14.8%). Most notably, removing topology-awareness entirely ("flat graph") increases ASR to 25.5% and utility loss to 9.3%—confirming that structural reasoning is essential, not optional, for effective multi-agent defense.

7. DISCUSSION

A. Practical Deployment Considerations

ShieldGraph is designed for integration into existing multi-agent frameworks as a middleware layer. The AMC operates on inter-agent message text without requiring access to agent model weights or internal states, enabling deployment alongside proprietary LLM APIs. The 2.4ms per-message latency is negligible relative to typical LLM inference times (500ms–2s), ensuring that defense does not become a bottleneck. For production deployments, we

recommend the "INSPECT + log" mode during initial deployment to calibrate thresholds before enabling active blocking.

B. Limitations

Several limitations warrant candid discussion. First, the AMC is trained on known attack patterns and may be vulnerable to novel adversarial techniques not represented in training data—an inherent limitation of all learned classifiers. Second, the quarantine protocol assumes that compromised agents can be replaced or restarted, which may not hold for agents maintaining critical long-term state. Third, ShieldGraph's effectiveness depends on accurate topology specification; dynamic topologies where agents spawn or reorganize at runtime require continuous graph updates. Fourth, coordinated exfiltration attacks that distribute adversarial behavior below per-channel detection thresholds remain challenging, as noted in Section VI-C.

C. Comparison with Concurrent Work

AdvEvo-MARL [13] achieves sub-20% ASR through adversarial co-evolution but requires RL training and operates only on RL-based agent systems. ShieldGraph achieves 7.2% ASR on the more challenging LLM-to-LLM orchestration paradigm without requiring training-time modifications. AC4A [19] and MCP proxy architectures [21] address tool-level access control but not inter-agent message security—these are complementary to ShieldGraph and could be integrated as an additional defense layer. The OWASP LLM Top 10 [24] ranking prompt injection as the #1 risk through 2026 further validates our focus on inter-agent injection as the primary threat vector.

D. Broader Implications

As multi-agent LLM systems become the default architecture for enterprise AI—handling customer service escalation, code review pipelines, research workflows, and autonomous decision-making—the absence of principled security mechanisms represents a systemic risk. ShieldGraph demonstrates that topology-aware defense is both feasible and effective, but the broader implication is that security must be a first-class architectural concern in multi-agent design, not a post-hoc addition. We advocate for "secure-by-topology" design principles where system architects explicitly reason about trust boundaries, propagation paths, and blast radii during the topology design phase.

8. CONCLUSION AND FUTURE WORK

This paper introduced ShieldGraph, a topology-aware runtime defense framework for multi-agent LLM systems. Through the combination of Trust Propagation Calculus, a lightweight adversarial message classifier, and a graph-aware quarantine protocol, ShieldGraph reduces attack success rates from 68.4% to 7.2% while preserving 94.1% task completion—establishing that principled defense of multi-agent systems is achievable without crippling their utility.

Our central finding is that topology-awareness is not merely an optimization but a necessity: topology-agnostic defenses suffer 18.3 percentage points higher attack success and 5.6 percentage points greater utility loss compared to ShieldGraph. The structural properties of multi-agent graphs—centrality, reachability, cut sets—provide essential information for both detecting adversarial propagation and containing its spread.

Future work will pursue four directions: (1) adaptive topology reconfiguration where ShieldGraph proactively reorganizes communication channels to minimize attack surface; (2) federated defense where multiple independently-operated multi-agent systems share threat intelligence without exposing proprietary topologies; (3) formal verification of quarantine correctness guarantees; and (4) extension to dynamic topologies where agents are spawned, terminated, or reorganized during execution.

REFERENCES

1. J. S. Park, J. O'Brien, C. J. Cai, et al., "Generative agents: Interactive simulacra of human behavior," in Proc. ACM Symp. User Interface Software Technol. (UIST), 2023.

2. LangChain, “LangGraph: Build stateful, multi-actor applications with LLMs,” 2024. [Online]. Available: https://github.com/langchain-ai/langgraph
3. J. Moura, “CrewAI: Framework for orchestrating role-playing AI agents,” 2024. [Online]. Available: https://github.com/crewAIInc/crewAI
4. OpenAI, “OpenAI Agents SDK,” OpenAI Documentation, Mar. 2025.
5. Z. Zhao, A. Chen, and M. Ribeiro, “A systematic survey of security threats and defenses in LLM-based AI agents,” arXiv preprint arXiv:2604.23338, 2026.
6. G. Gu, L. Zhang, and X. Wang, “LLM-to-LLM prompt injection within multi-agent systems,” arXiv preprint arXiv:2410.07283, 2024.
7. Y. Chen, H. Liu, and Z. Wang, “TAMAS: Benchmarking adversarial risks in multi-agent LLM systems,” in Proc. Assoc. Comput. Linguistics (ACL), 2026.
8. J. Xu, S. Li, and R. Zhang, “Exploiting LLM multi-agent systems via topology-guided adversarial propagation,” arXiv preprint arXiv:2512.04129, 2025.
9. Y. Zhang, K. Ma, and L. Chen, “Why do multi-agent LLM systems fail?” arXiv preprint arXiv:2503.13657, 2025.
10. A. Wallace, J. Chen, and T. Liu, “Adaptive evaluation of out-of-band defenses against prompt injection in LLM agents,” arXiv preprint arXiv:2606.26479, 2026.
11. H. Zhang, Y. Li, and Q. Sun, “ICON: Indirect prompt injection defense for agents based on inference-time correction,” arXiv preprint arXiv:2602.20708, 2026.
12. Y. Han, Z. Liu, and W. Chen, “AdvEvo-MARL: Shaping internalized safety through adversarial co-evolution in multi-agent reinforcement learning,” in Proc. Int. Conf. Mach. Learn. (ICML), 2026.
13. Y. Han, Z. Liu, and W. Chen, “Shaping internalized safety through adversarial co-evolution in multi-agent reinforcement learning,” arXiv preprint arXiv:2510.01586, 2025.
14. M. Roberts, S. Chen, and A. Patel, “Intent-verified delegation chains for securing federal multi-agent AI systems,” arXiv preprint arXiv:2604.02767, 2026.
15. L. Zhao, M. Wang, and H. Zhang, “LLM-based multi-agent orchestration: A survey of frameworks, communication protocols, and emerging patterns,” Preprints, 2026.
16. M. Lanham, “Multi-agent in production in 2026: What actually survived,” Medium, Apr. 2026.
17. Z. Chen, Y. Li, and H. Wang, “Taming various privilege escalation in LLM-based agent systems,” arXiv preprint arXiv:2601.11893, 2026.
18. L. Chen, H. Zhang, and R. Wang, “Threat surfaces, attacks, defenses, and evaluation for LLM agents,” arXiv preprint arXiv:2606.10749, 2026.
19. T. Müller, A. Becker, and S. Schmidt, “AC4A: Access control for agents,” arXiv preprint arXiv:2603.20933, 2026.
20. J. Kim, Y. Zhang, and M. Chen, “Do coding agents understand least-privilege authorization?” arXiv preprint arXiv:2605.14859, 2026.
21. R. Patel, S. Liu, and H. Chen, “Architectural enforcement via MCP proxy for LLM tool access control,” arXiv preprint arXiv:2605.18414, 2026.
22. B. Schneier, “Applied Cryptography: Protocols, Algorithms, and Source Code in C,” 2nd ed. New York, NY, USA: Wiley, 1996.
23. P. He, J. Gao, and W. Chen, “DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing,” in Proc. Int. Conf. Learn. Representations (ICLR), 2023.
24. OWASP, “OWASP top 10 for LLM applications,” OWASP Foundation, 2025. [Online]. Available: https://owasp.org/www-project-top-10-for-large-language-model-applications/
25. W. Li, S. Chen, and M. Wang, “Coordination as an architectural layer for LLM-based multi-agent systems,” arXiv preprint arXiv:2605.03310, 2026.
26. J. Wu, H. Zhang, and T. Liu, “SILO-BENCH: A scalable environment for evaluating distributed coordination in multi-agent LLM systems,” in Proc. Assoc. Comput. Linguistics (ACL), 2026.
27. A. Chen, Y. Liu, and Z. Wang, “Assessing automated prompt injection attacks in agentic environments,” arXiv preprint arXiv:2606.10525, 2026.
28. H. Liu, J. Chen, and M. Zhao, “Indirect prompt injection against LLM agents via feedback-guided iterative optimization,” arXiv preprint arXiv:2605.24659, 2026.
29. S. Park, M. Chen, and L. Zhang, “Benchmarking emergent coordination in large-scale LLM populations,” arXiv preprint arXiv:2603.03555, 2026.
30. Auth0, “Why AI agents need their own permission model,” Auth0 Blog, Jun. 2026.
31. T. Pan, “Least privilege for autonomous AI agents: The minimal footprint principle,” tianpan.co, Apr. 2026.
32. M. Goodrich, “Every agent protocol earns its keep at a boundary,” mattgoodrich.com, Jun. 2026.