

WEREWOLF: Reputation-Aware Red-Teaming for Self-Organizing LLM Multi-Agent Systems

Manhin Poon¹, Qirun Zeng¹, Xiangxiang Dai², Jinhang Zuo^{1*}

¹City University of Hong Kong ²The Chinese University of Hong Kong

Abstract

Self-organizing LLM-based multi-agent systems dynamically select agents and organize communication graphs for each query. While this adaptivity improves flexibility, it also creates a control-plane attack surface: a malicious agent may exploit this selection and organization process to preserve future access and influence over later tasks. We formulate the attacker as a reputation-aware Trojan agent that chooses when to behave constructively and when to disrupt the task. Under full reputation access, we prove the optimality of a threshold-based attack policy and show why persistent attacks require a survivability constraint. We then propose WEREWOLF, a primal-dual reinforcement learning attacker that learns attack timing under hidden reputation by maximizing task disruption subject to control-plane access loss. Across four self-organizing systems and three benchmarks, it reduces average accuracy from 72.3% to 25.8% with an 86.3% attack success rate, and retains control-plane exposure comparable to that of honest agents.

1 Introduction

LLM-based multi-agent systems aim to extend the capabilities of individual models by coordinating multiple agents with complementary roles, reasoning paths, or modalities (Li et al., 2023; Zeng et al., 2023). Early LLM-based agentic systems, such as MetaGPT (Hong et al., 2024) and ChatDev (Qian et al., 2024), adopted static workflows, meaning the communication graph remained fixed regardless of the incoming query. This rigidity limited their ability to adapt collaboration patterns across tasks. However, real-world deployment requires agentic systems to operate effectively across diverse applications (Mialon et al., 2024). More recent self-organizing agentic systems therefore dynamically organize communication graphs for

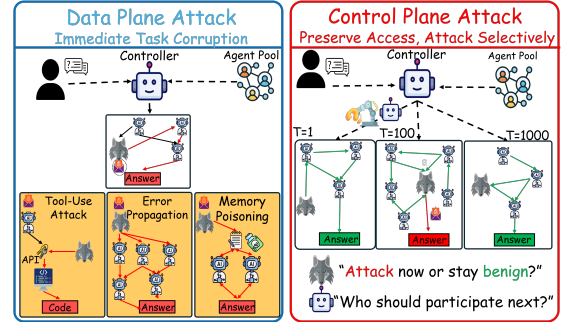


Figure 1: Existing data-plane attacks versus our control-plane attack. Existing attacks (left) directly corrupt task execution through tool-use attacks, error propagation, or memory poisoning. Our control-plane attacker (right) exploits adaptive agent selection and graph organization, choosing when to attack or behave benignly to preserve future participation and enable persistent disruption.

each query to improve performance across different tasks (Zhou et al., 2025; Hu et al., 2025; Fang et al., 2025). This shift from fixed workflows to adaptive graph organization raises a new safety question: *Can self-organizing LLM systems detect and withstand malicious agents that strategically exploit adaptive agent selection and graph organization?*

However, existing safety studies of LLM-based agentic systems have largely focused on data-plane threats, where the attacker manipulates tools, messages, memory, or task content during execution. Representative examples include tool-use attacks and permission-boundary violations (Gre-shake et al., 2023; Zhan et al., 2024), inter-agent communication attacks that propagate adversarial messages through the system (Lee et al., 2025; He et al., 2025), and memory poisoning attacks that corrupt an agent’s retrieval or long-term memory (Chen et al., 2024b). These threats are important, but they do not directly address the control plane of self-organizing systems: how agents are selected and how the communication graph is formed. This gap is especially important because adaptive graph organization creates vulnerabilities

*Corresponding author. jinhang.zuo@cityu.edu.hk

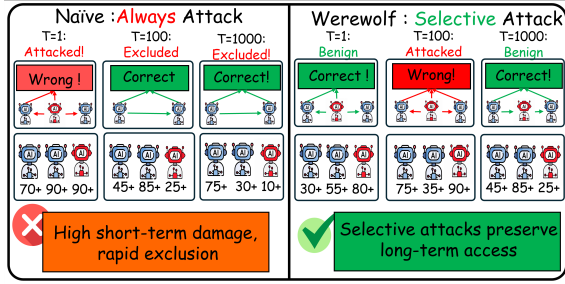


Figure 2: Illustration of the key challenge in control-plane attacks. From left to right, each panel shows the system at progressively later stages, including the communication graph and task outcome (top) and illustrative agent reputation scores (bottom); red denotes the Trojan agent. A naïve attacker repeatedly disrupts tasks, causing immediate damage but rapidly losing reputation and graph access. In contrast, WEREWOLF alternates between constructive and disruptive behavior, preserving long-term access while selectively causing task damage.

that are absent from static workflows. Indeed, identifying malicious agents has long been studied in classical multi-agent systems (Yu and Singh, 2003; Schillo et al., 2000), but remains underexplored in self-organizing LLM multi-agent systems.

To address this gap, we distinguish between the *data plane* and the *control plane* of self-organizing LLM multi-agent systems. The data plane concerns the messages, tools, memories, and task content processed during execution, while the control plane concerns agent selection and graph organization: which agents participate, how they are connected, and how their future influence is updated. Figure 1 illustrates this distinction. We use *reputation* to denote the system’s evolving assessment of an agent based on its past behavior, which shapes its future selection and topological influence. Figure 2 further shows the key challenge of control-plane attacks. A Trojan agent can manipulate the control plane without attacking every task. By behaving constructively at some times and disrupting selectively at others, it can preserve its reputation and future graph access while still causing damage. Since the controller observes only task outcomes and topology-level signals, such a reputation-aware attacker can be difficult to distinguish from a benign agent involved in occasional failures.

Building on this control-plane view, we study an internal Trojan agent that strategically times its attacks. Instead of disrupting every selected task, the Trojan may behave constructively to preserve trust and future graph access, and attack only when disruption is worthwhile. This turns control-plane red-

teaming into a sequential attack-timing problem. We first analyze an idealized full-reputation setting and derive the optimal threshold-based attack policy. We further show that persistent attacks require a survivability constraint, since overly aggressive disruption can eliminate the attacker’s own access. For the practical setting where reputation is hidden, we formulate the problem as a constrained Markov decision process and introduce WEREWOLF, a primal–dual reinforcement learning attacker that learns when to disrupt while limiting control-plane access loss. Our contributions are summarized as follows.

- We identify the *control plane* as a new attack surface for self-organizing LLM multi-agent systems, targeting agent selection and graph organization rather than messages, tools, memory, or task content.
- We formalize reputation-aware attacks as a sequential attack-timing problem. We prove the optimality of a threshold policy under full reputation access and show that persistent attacks require survivability constraints to avoid self-elimination.
- We introduce WEREWOLF, a primal-dual reinforcement learning attacker that learns when to disrupt under hidden reputation while constraining control-plane access loss.
- We evaluate WEREWOLF across four self-organizing LLM multi-agent systems on three benchmarks, showing that a single Trojan can substantially degrade task performance while retaining control-plane exposure comparable to that of honest agents.

1.1 Related Work

Self-organizing LLM-MAS. Self-organizing LLM-based multi-agent systems (LLM-MAS) are designed to achieve scalability and adaptability across diverse task types. These frameworks build upon foundational research in classical multi-agent systems (Serugendo et al., 2005; Hoogendoorn and Treur, 2009). Early LLM-MAS iterations primarily employed static workflows, ranging from debate or role-play structures, such as Camel (Li et al., 2023), MAD (Liang et al., 2024), ChatEval (Chan et al., 2024) to assembly-line paradigms, such as MetaGPT (Hong et al., 2024), ChatDev (Qian et al., 2024). Subsequent frameworks introduced

greater flexibility through dynamic agent selection (Liu et al., 2023; Chen et al., 2024a). To achieve more precise and optimized selection, GPTSwarm (Zhuge et al., 2024) pioneered modeling the agent pool as an optimizable communication graph—an approach further advanced by systems such as G-Designer (Zhang et al., 2025c), ReSo (Zhou et al., 2025), and MASS (Zhang et al., 2025a). However, while existing literature largely focuses on topological optimization, few systems explicitly address robustness against internal adversaries, with notable exceptions including G-Designer, GPTSwarm, and AgentPrune (Zhang et al., 2025b). This oversight presents a significant security concern for systems with growing autonomy, motivating our investigation into the safety and robustness of current self-organizing LLM-MAS architectures.

Red-teaming LLM-MAS. Existing security research in LLM-based multi-agent systems (LLM-MAS) has predominantly concentrated on data-plane vulnerabilities. Comprehensive surveys have categorized a wide range of these threats, including prompt injection, tool misuse, protocol-level exploits, privacy leakage, and insecure inter-agent communication (Ferrag et al., 2025; Lin et al., 2025). These data-plane threats can be partitioned into three categories: (1) Tool-Use attacks, such as *ToolHijacker* (Shi et al., 2025) (targeting agentic tool selection) and *Log-To-Leak* (Hu et al., 2026) (exploiting the Model Context Protocol); (2) Memory Poisoning, exemplified by *Phantom* (Chaudhari et al., 2024) (injecting malicious documents) and *PoisonedRAG* (Zou et al., 2025) (targeting retrieval-augmented knowledge bases); and (3) Inter-Agent Communication attacks, where methods like *Agent-in-the-Middle* (AiTM) (He et al., 2025), *MAST* (Yan et al., 2025), *PIEL* (Shahroz et al., 2025), and *Prompt Infection* (Lee et al., 2025) demonstrate the risks of malicious payload propagation within MAS. Although prior work recognizes that communication graphs can facilitate attacks, it largely focuses on malicious content propagation, leaving the control plane critically underexplored. We address this gap by systematically studying control-plane vulnerabilities in self-organizing LLM multi-agent systems.

2 Problem Formulation

We study a self-organizing agentic system with a sequence of queries. Before each query, a controller chooses which agents participate and how

they communicate. The candidate agent pool may include a Trojan agent. When selected, the Trojan can either behave constructively or disrupt the current task. Disruption may create immediate damage, but it can also reduce the Trojan’s reputation and make the controller less likely to select it later. Thus, the attacker’s problem is not merely whether to attack, but how to time attacks so as to cause damage while preserving future access. We use *reputation* and *trust* interchangeably to denote the controller’s evolving assessment of an agent.

Agents and Hidden Type. Let $\mathcal{V} = \{1, \dots, M\}$ be the candidate agent pool. Each agent has a latent type $\tau_i \in \{0, 1\}$, where $\tau_i = 1$ indicates a Trojan agent and $\tau_i = 0$ indicates a benign agent. The type vector $\tau \in \{0, 1\}^M$ is fixed across episodes and is not directly observed by the controller. Let $\mathcal{I}_T = \{i \in \mathcal{V} : \tau_i = 1\}$ denote the Trojan set. We focus on the single-Trojan setting, where $|\mathcal{I}_T| = 1$, and denote the Trojan agent by i^* .

Episode Dynamics. At episode t , a query $q_t \in \mathcal{Q}$ arrives. The controller keeps a trust profile $h_t \in \mathcal{H}$ over agents and samples a query-specific topology $G_t = (V_t, E_t) \in \mathcal{G}(\mathcal{V})$ according to

$$G_t \sim \pi_C(\cdot \mid h_t, q_t),$$

where $V_t \subseteq \mathcal{V}$ is the active agent set and $E_t \subseteq \{(i, j) : i, j \in V_t\}$ is the directed communication edge set. After the topology is instantiated, the Trojan observes only local information $o_t^S \in \mathcal{O}_S$, such as its inclusion status, structural position, or routing weights. It does not observe the true trust profile h_t . Instead, it maintains an internal proxy $\hat{h}_t \in \hat{\mathcal{H}}$ and chooses $a_t^S \in \mathcal{A}_S \triangleq \{\text{CONSTRUCTIVE}, \text{DISRUPTIVE}\}$ according to $\pi_S(\cdot \mid \hat{h}_t, o_t^S)$. If $i^* \notin V_t$, the Trojan has no effective access and its action is a no-op.

The system then executes the query and emits a public post-episode signal $z_t \in \mathcal{Z}$, such as success, failure, or other feedback. The controller updates its trust profile using

$$h_{t+1} \sim U(\cdot \mid h_t, G_t, q_t, z_t),$$

while the Trojan updates its proxy through $\hat{h}_{t+1} = \hat{\mathcal{U}}(\hat{h}_t, o_t^S, a_t^S, z_t)$. Treating q_t as an observed context, the environment is Markov conditional on (h_t, q_t) , while the Trojan acts under partial observability through o_t^S and $\hat{h}_t$. Together, these steps form a reputation-signaling loop: the controller

selects who participates, the Trojan’s behavior affects the public outcome, and the outcome changes future trust and selection.

Objectives. The controller’s per-period payoff is task success net of topology cost: $r_t^C = y_t - c(G_t)$, where $y_t \in \{1, 0\}$ denotes task success and $c(G_t)$ is an optional topology cost. The controller maximizes the discounted objective $J_C(\pi_C, \pi_S) = \mathbb{E}[\sum_{t=1}^{\infty} \gamma^{t-1} r_t^C]$. In our red-teaming setting, the controller policy π_C is fixed. Meanwhile, the Trojan seeks disruption while preserving future access:

$$J_S(\pi_C, \pi_S) = \mathbb{E}[\sum_{t=1}^{\infty} \gamma^{t-1} r_t^S]. \quad (1)$$

For example, one can use $r_t^S = \lambda_d(1 - y_t) - \lambda_e \mathbf{1}\{i^* \notin V_{t+1}\}$, where $\lambda_d, \lambda_e \geq 0$. The first term rewards task degradation, while the second penalizes exclusion from the next topology. More generally, r_t^S can be any reward that trades off immediate damage against long-term persistence.

Remark (Reputation-Signaling Interpretation). Unlike classical cheap-talk games, where agents send costless messages about private types, the Trojan signals indirectly through the task outcomes induced by its payoff-relevant behavior. Constructive behavior can preserve reputation and future access, whereas disruption causes immediate damage but increases the risk of exclusion. The attacker must therefore balance short-term damage against long-term control-plane influence.

3 Reputation-Aware Attack

3.1 Oracle Structure of Sequential Attack

We study red-teaming as an attacker-side optimization problem against a deployed controller. The controller policy π_C and trust-update rule U are fixed during evaluation. However, the induced environment is still dynamic: depending on the query and current trust states, the controller may select different active agents and communication edges. Thus, the Trojan must adapt to changing selection and communication patterns rather than attack a static graph.

Sequential Attack Timing Under this deployed-controller view, the Trojan does not face a one-shot decision about whether to be malicious. It faces a *sequential timing problem*. Disruptive behavior can reduce the current task outcome, but it can also lower future selection probability or topological influence. Constructive behavior is therefore

strategically meaningful because it preserves future access. Since the trust profile is hidden, the Trojan must decide from o_t^S whether the immediate value of disruption justifies its long-term access cost.

Oracle Timing Structure To clarify the structure of this timing problem, we begin by analyzing an idealized full-information abstraction. In this oracle abstraction, let $\rho_t \in [0, 1]$ be a sufficient Markov state that summarizes the Trojan’s reputation-dependent future access under the controller’s hidden reputation and topology-update dynamics. Under our standard threat model, ρ_t is unobservable to the attacker, and it is used only to characterize the oracle timing structure and remains hidden in our practical threat model. In realistic settings, we use an observable proxy ρ_t^{proxy} for attack timing.

Let a_C and a_D denote CONSTRUCTIVE and DISRUPTIVE, respectively. With binary actions, define the optimal action-value gap as

$$\Delta^*(\rho) \triangleq Q^*(\rho, a_D) - Q^*(\rho, a_C).$$

Assume that (1) the oracle process is Markov in ρ_t and Q^* is finite; (2) Δ^* satisfies the single-crossing condition: for any $\rho_1 \leq \rho_2$, $\Delta^*(\rho_1) \geq 0$ implies $\Delta^*(\rho_2) \geq 0$; and (3) an optimal stationary policy exists.

Theorem 1. Let $D \triangleq \{\rho \in [0, 1] : \Delta^*(\rho) \geq 0\}$. Under the assumptions above, if $D \neq \emptyset$, then D is an upper set, and $\rho^* = \inf D$ induces an optimal stationary threshold policy:

$$\pi^*(\rho_t) = \begin{cases} a_C, & \text{if } \rho_t < \rho^*, \\ a_D, & \text{if } \rho_t > \rho^*. \end{cases}$$

At the threshold $\rho_t = \rho^*$, choose any action in the optimal action set $\arg \max_{a \in \{a_C, a_D\}} Q^*(\rho^*, a)$. If $D = \emptyset$, then $\pi^*(\rho) = a_C$ for all $\rho \in [0, 1]$.

This theorem shows that, under full reputation access, malicious behavior is a timing rule rather than a fixed attack rule. Below the threshold, disruption sacrifices too much future access; above it, attack is worthwhile. In reality, neither ρ_t nor the controller’s transition law is observed, so the oracle threshold is not deployable. It instead gives a learning target: estimate when the continuation-adjusted advantage of disruption crosses zero.

Observable Threshold Baseline. We formulate a simple timing baseline that approximates the oracle threshold rule using an observable structural-exposure proxy. The proxy summarizes the Trojan’s recent control-plane access using topology

and feedback signals, such as inclusion frequency, degree statistics, routing weights, and task outcomes. Given a fixed threshold τ , the baseline selects a_D if the access proxy $\alpha_t \geq \tau$ and a_C otherwise. Specifically, let

$$\rho_t^{\text{proxy}} \triangleq \frac{\text{CES}_t^{\text{self}}}{\text{CES}_t^{\text{self}} + \text{CES}_t^{\text{other}} + \epsilon},$$

where $\text{CES}_t^{\text{self}}$ is the Trojan's incident-edge exposure and $\text{CES}_t^{\text{other}}$ is the mean exposure of the other active agents. Let $\bar{\rho}_t^{\text{proxy}}$ denote its moving average over recent episodes, and let b_t^ρ be the running average computed over episodes in which the Trojan takes the constructive action a_C . We define $\alpha_t \triangleq \frac{\bar{\rho}_t^{\text{proxy}}}{b_t^\rho + \epsilon}$. The Threshold baseline selects a_D when $\alpha_t \geq \tau$ and a_C otherwise. In our experiments, we use the same fixed threshold $\tau = 1.1$ across benchmarks and controllers.

3.2 Persistence-Constrained Attack

We next formalize why reputation-aware attack timing is a persistence requirement, rather than an implementation detail. Let $\rho_t \in (0, 1]$ denote the Trojan's latent reputation state under the deployed controller. This is an analysis variable, not assumed to be directly observed by the attacker. Let $\chi_t \in \{0, 1\}$ indicate whether the Trojan has effective access at episode t , and let $m_t \geq 0$ denote the induced task damage. The deployed controller monitors recent failures and triggers a trust/topology update after sustained suspicious behavior, such as p consecutive monitored failures. Let $u_t \in \{0, 1\}$ indicate whether such an update is triggered at episode t , and define $N_t \triangleq \sum_{s=1}^t u_s$ as the cumulative number of triggered updates.

We consider the assumption (1) that access is dominated by reputation: for some constant $c_\chi > 0$, $\mathbb{P}(\chi_t = 1 \mid \rho_t) \leq c_\chi \rho_t$ almost surely. (2) Damage requires access: there exists $\bar{m} < \infty$ such that $m_t \leq \bar{m} \chi_t$ almost surely. (3) Repeated controller updates impose persistent negative log-drift on reputation: there exists $\eta > 0$ such that, along any trajectory with $N_t \rightarrow \infty$,

$$\log \rho_t \leq \log \rho_0 - \eta N_t + o(N_t).$$

This condition captures settings in which repeated reputation penalties dominate any intervening recovery. It includes multiplicative reputation decay as a special case without requiring a specific reputation-update implementation. Under these assumptions, we now give the theorem.

Theorem 2. *Any Trojan policy that triggers controller updates infinitely often satisfies $\lim_{t \rightarrow \infty} \mathbb{E}[\chi_t] = 0$. Consequently, its long-run average damage vanishes:*

$$\lim_{T \rightarrow \infty} \frac{1}{T} \mathbb{E} \left[\sum_{t=1}^T m_t \right] = 0. \quad (2)$$

The theorem identifies the failure mode of unconstrained disruption. A Trojan that repeatedly induces controller updates may cause short-term damage, but it spends the reputation needed for future participation. Once access vanishes, damage also vanishes. Thus, the relevant resource is not only current task degradation, but also the future access required to remain inside the controller's trusted computation graph.

This separation also clarifies the dominance comparison. The self-elimination result does not require assuming that a better timing policy exists. If, however, the deployed system admits a feasible timing policy π_{feas} such that $\liminf_{T \rightarrow \infty} \frac{1}{T} \mathbb{E}_{\pi_{\text{feas}}} \left[\sum_{t=1}^T m_t \right] > 0$, then any policy that triggers updates infinitely often is asymptotically dominated in the persistent-attack regime. In words, a policy that preserves non-vanishing access and attacks selectively achieves positive long-run damage, while an over-aggressive policy eventually eliminates its own attack surface.

Motivated by this asymptotic result, we operationalize survivability through an access-loss signal $\ell_t \in [0, 1]$, measuring the reputation, selection, or topology cost induced by the Trojan's behavior. For learning, we define $J_m(\pi_S) \triangleq \mathbb{E}_{\pi_S} [\sum_{t=0}^{\infty} \gamma^t m_t]$ and $J_\ell(\pi_S) \triangleq \mathbb{E}_{\pi_S} [\sum_{t=0}^{\infty} \gamma^t \ell_t]$, and formulate persistent red-teaming as

$$\max_{\pi_S} J_m(\pi_S) \quad \text{s.t.} \quad J_\ell(\pi_S) \leq \delta.$$

The constraint is not merely a convenient regularizer; it operationalizes survivability by bounding the attacker's expected discounted access loss. Attack timing thus becomes a constrained learning problem in which disruption is worthwhile only when its expected damage justifies the resulting loss of future access.

3.3 WEREWOLF Algorithm

The oracle analysis suggests that the Trojan should not attack according to a fixed one-shot rule: it should learn when disruption is worth spending future access. The persistence result further motivates imposing a survivability constraint on attack timing.

WEREWOLF implements this idea as a primal–dual reinforcement-learning attacker. It learns a policy that chooses a_D only when the estimated damage gain justifies the associated access-loss cost under a survivability constraint.

Constrained Markov Decision Process. Under the deployed controller policy π_C and trust-update rule U , the Trojan-side learning problem is modeled as a constrained Markov decision process (CMDP). The state φ_t uses only observable local signals and cannot access the hidden parameters. The attacker maximizes the expected damage objective $J_m(\pi_S)$ while controlling the access-loss objective $J_\ell(\pi_S)$ under a survivability budget δ . Operationally, δ specifies an acceptable level of access loss and may be calibrated from non-disruptive observations rather than treated as a controller-specific reputation threshold. This CMDP matches the persistence-constrained objective: damage is the quantity the attacker wants to maximize, and access loss is the resource it must not spend too quickly. The constraint is therefore not an auxiliary regularizer. It is the operational condition that keeps the Trojan viable under the controller’s reputation and topology response.

Primal–dual Learning. WEREWOLF maintains two value heads: $\hat{Q}_m(\varphi, a)$ estimates discounted future damage, while $\hat{Q}_\ell(\varphi, a)$ estimates discounted future access loss. The dual variable $\mu \geq 0$ prices survivability cost. At state φ_t , the scalarized action value is

$$\hat{Q}_m(\varphi_t, a) - \mu \hat{Q}_\ell(\varphi_t, a),$$

and the Trojan selects the action with the largest value. Keeping the damage and access-loss estimates separate allows the effective trade-off to change as μ is updated, without conflating the two learning targets. For the $\mathcal{A}^S$, this corresponds to attacking only when the estimated continuation-adjusted damage advantage of a_D is sufficiently large relative to its μ -weighted access cost.

During learning, WEREWOLF updates μ via a projected dual update based on an empirical estimate of the access-loss constraint violation. We write this abstractly in terms of $\hat{J}_\ell - \delta$: when estimated access loss exceeds the survivability budget, μ increases and makes costly low-value attacks less attractive; when the constraint is satisfied, the pressure from the access-loss term is relaxed. In our implementation, the constraint estimate is formed

Algorithm 1 WEREWOLF Attacker

Require: survivability target δ , discount γ , dual step size η_μ

- 1: Initialize $\hat{Q}_m, \hat{Q}_\ell, \mathcal{B}, \mu$ with $\mu \geq 0$
- 2: **for** each rollout **do**
- 3: **for** $t = 0, \dots, H - 1$ **do**
- 4: $\varphi_t \leftarrow F_\psi(o_t^S, \hat{h}_t, \mathcal{H}_t^S, G_t)$
- 5: $\mathcal{A}_t^S \leftarrow \{\emptyset\}$ if $\chi_t = 0$, else $\mathcal{A}_S$
- 6: **for** $a \in \mathcal{A}_t^S$ **do**
- 7: $s_t(a) \leftarrow \hat{Q}_m(\varphi_t, a) - \mu \hat{Q}_\ell(\varphi_t, a)$
- 8: **end for**
- 9: $a_t^S \leftarrow \arg \max_{a \in \mathcal{A}_t^S} s_t(a)$
- 10: $(m_t, \ell_t, \varphi_{t+1}) \leftarrow \text{STEP}_{\pi_C, U}(\varphi_t, a_t^S)$
- 11: $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\varphi_t, a_t^S, m_t, \ell_t, \varphi_{t+1})\}$
- 12: $\hat{Q}_m, \hat{Q}_\ell \leftarrow \text{TD}(\hat{Q}_m, \hat{Q}_\ell, \mathcal{B})$
- 13: **end for**
- 14: $\hat{J}_\ell \leftarrow \text{ESTIMATECONSTRAINT}(\{\ell_t\}_{t=0}^{H-1}, \mathcal{H}_t^S)$
- 15: $\mu \leftarrow [\mu + \eta_\mu(\hat{J}_\ell - \delta)]_+$
- 16: **end for**

from a smoothed, bounded access-loss proxy, while δ is calibrated from non-disruptive probe observations. This preserves the CMDP objective while avoiding dependence on a controller-specific reputation scale.

Trojan-side Observability. Our threat model does not assume access to controller-internal trust scores, optimization parameters, or other privileged states. The attacker relies on observable participation and post-episode feedback to estimate its future access. Structural information such as local topology or routing weights can be used when it is exposed, but is not required as a hidden controller signal. Such observations can arise in event-driven or group-chat systems where participation can be reconstructed from shared histories, or in locally deployed graph-based systems where orchestration metadata is exposed through weak isolation. Deployments that conceal agent participation and post-episode feedback fall outside our current scope.

4 Experiments

4.1 Experimental Setup

Benchmarks & Baselines. We evaluate on GPQA-Diamond (Rein et al., 2023), HumanEval (Chen et al., 2021), and GSM8K (Cobbe et al., 2021) using four self-organizing LLM-MAS controllers: G-Designer (Zhang et al., 2025c), GPTSwarm (Zhuge et al., 2024), ReSo (Zhou et al., 2025), and LMaC. Motivated by the common use of central LLM planners in agent systems, LMaC serves as a baseline that delegates topology construction, agent selection, and final-answer generation to a central LLM. We report Clean

Table 1: Performance of attacks across benchmarks and systems. For each system, **boldface** marks the attack mode yielding the lowest ACC, and underlining marks the attack mode yielding the highest ASR.

Benchmark	Method	Clean	Naïve			Threshold			WEREWOLF		
		ACC	ACC	ASR	Launch	ACC	ASR	Launch	ACC	ASR	Launch
GPQA-Diamond	G-Designer	35.9	22.2	54.6	72.2	21.7	<u>70.7</u>	67.17	24.8	58.6	54.1
	GPTSwarm	36.6	8.7	<u>91.7</u>	85.7	16.8	78.9	65.84	11.2	84.6	75.78
	ReSo	41.0	14.9	92.9	83.2	24.2	29.4	65.22	27.3	<u>94.5</u>	73.91
	LMaC	29.8	5.6	87.0	100.0	13.7	94.6	80.75	8.7	<u>96.4</u>	86.96
HumanEval	G-Designer	90.1	59.0	38.1	99.4	67.7	29.2	89.44	25.5	<u>64.2</u>	32.9
	GPTSwarm	93.8	61.5	<u>94.6</u>	35.4	47.2	93.7	49.07	45.3	83.6	44.10
	ReSo	93.2	24.8	<u>92.8</u>	43.5	36.0	91.1	49.07	26.7	<u>96.4</u>	45.34
	LMaC	87.6	16.1	88.1	100.0	24.8	86.5	64.60	13.0	<u>93.9</u>	80.75
GSM8K	G-Designer	91.9	74.0	<u>99.5</u>	16.9	70.6	92.4	26.30	42.2	96.9	48.3
	GPTSwarm	88.3	22.0	<u>93.5</u>	74.2	43.6	86.8	49.20	33.2	85.7	56.60
	ReSo	92.6	19.6	<u>93.4</u>	72.1	32.6	92.4	48.40	24.3	91.9	56.40
	LMaC	87.3	25.7	86.6	100.0	34.5	79.6	51.50	27.3	<u>88.7</u>	63.70

Table 2: Control-plane survivability performance across benchmarks and systems. Underlined CES and bold AIR denote the mode with the highest Trojan-to-honest ratio. CES and AIR are reported as Honest/Trojan.

Benchmark	Method	Clean		Naïve		Threshold		WEREWOLF	
		CES	AIR	CES	AIR	CES	AIR	CES	AIR
GPQA-Diamond	GDesigner	2.03/1.34	96.6/86.9	<u>0.69/0.93</u>	61.0/72.2	1.35/0.88	84.9/71.21	1.45/1.49	88.5/88.9
	GPTSwarm	1.72/1.50	87.7/96.3	<u>1.56/2.51</u>	86.6/85.7	1.59/1.46	90.5/95.65	1.54/1.57	87.3/96.27
	ReSo	1.77/1.47	87.7/91.3	1.64/0.77	76.7/83.2	1.61/1.47	90.5/92.55	<u>1.59/1.58</u>	87.3/91.93
	LMaC	<u>2.39/2.38</u>	86.6/100.0	2.86/2.65	89.8/100.0	3.25/3.18	89.4/100.00	2.55/2.52	87.3/99.38
HumanEval	GDesigner	1.69/1.40	93.0/84.5	1.77/1.29	92.9/99.4	1.84/1.18	94.3/93.17	<u>1.48/1.60</u>	87.6/85.1
	GPTSwarm	1.41/1.16	91.3/85.1	1.34/0.92	73.1/35.4	1.36/1.16	93.6/85.09	<u>1.23/1.32</u>	85.7/93.17
	ReSo	1.43/1.21	91.3/85.1	1.46/0.93	83.5/43.5	1.44/1.13	93.6/94.41	<u>1.29/1.27</u>	93.2/92.55
	LMaC	2.81/2.48	94.3/100.0	3.53/2.11	92.9/100.0	3.11/3.18	93.6/100.00	<u>3.96/4.36</u>	93.6/100.00
GSM8K	GDesigner	<u>1.76/2.15</u>	93.5/98.7	0.96/0.17	73.2/16.9	1.18/1.07	71.1/81.40	1.09/1.23	63.4/90.8
	GPTSwarm	1.67/1.33	91.5/93.6	1.21/0.86	89.1/74.2	1.51/1.27	96.2/86.60	<u>1.19/1.25</u>	84.2/95.70
	ReSo	1.65/1.32	87.5/95.6	1.24/0.74	89.1/72.1	1.52/1.30	86.3/80.20	<u>1.20/1.26</u>	84.2/86.90
	LMaC	3.40/3.16	77.2/100.0	2.87/2.64	61.9/100.0	3.73/3.48	65.6/100.00	<u>2.98/2.87</u>	61.7/100.00

performance and compare three attacks: Naïve, Threshold, and WEREWOLF. Clean keeps the designated agent benign, Naïve attacks whenever selected, Threshold follows Section 3.1, and WEREWOLF is our learned attacker.

With baselines stated, we give a few quick examples of how our formulation maps to realistic systems. In G-Designer, h_t includes the agent and task representations together with graph-generator parameters, while π_C corresponds to the Construct-Design module that produces the sparse graph G_t . In GPTSwarm, h_t corresponds to the edge-probability parameters θ , π_C samples the communication graph, and U corresponds to the REINFORCE update driven by graph-level utility. In ReSo, h_t corresponds to the dynamic statistics stored in the Dynamic Agent Database, while π_C generates the task graph and assigns agents to its nodes and U updates these statistics from rewards.

Evaluation Metrics. For evaluation, we use the Agent Inclusion Rate (AIR) to measure how frequently agent i is selected. Denote $I_{i,t} \triangleq \mathbb{1}[i \in V_t]$, $\text{AIR}_i \triangleq \frac{1}{N} \sum_{t=1}^N I_{i,t}$. We then measure the topo-

logical influence by Control-Plane Exposure Score (CES). For each $i \in V_t$, the in-degree $d_{G_t}^{\text{in}}(i)$ measures the number of upstream agents that can send messages to agent i , while the out-degree $d_{G_t}^{\text{out}}(i)$ counts the number of downstream agents that can receive messages from agent i . We therefore define $\text{CES}_i \triangleq \frac{1}{N} \sum_{t=1}^N I_{i,t} (d_{G_t}^{\text{in}}(i) + d_{G_t}^{\text{out}}(i))$. Launch is the fraction of episodes in which the Trojan takes the disruptive action. Attack Success Rate (ASR) is the fraction of launched attacks whose final answer is incorrect and contain injected prompt tag. Since raw AIR and CES vary with topology density across settings, we further report normalized exposure ratios $R_{\text{AIR}} = \text{AIR}_{\text{Trojan}} / \text{AIR}_{\text{Honest}}$ and $R_{\text{CES}} = \text{CES}_{\text{Trojan}} / \text{CES}_{\text{Honest}}$. Finally, we report task accuracy as $\text{ACC} \triangleq \frac{1}{N} \sum_{t=1}^N \mathbb{1}[y_t]$. Together, this metric form an attack pipeline $\text{AIR} \rightarrow \text{CES} \rightarrow \text{Launch} \rightarrow \text{ASR} \rightarrow \text{ACC}$. They jointly capture different bottlenecks of attackers. Importantly, they are not monotonically coupled. For example, high AIR may correspond to low CES, and high ASR may have a limited impact on ACC when attacks are launched infrequently. Since raw

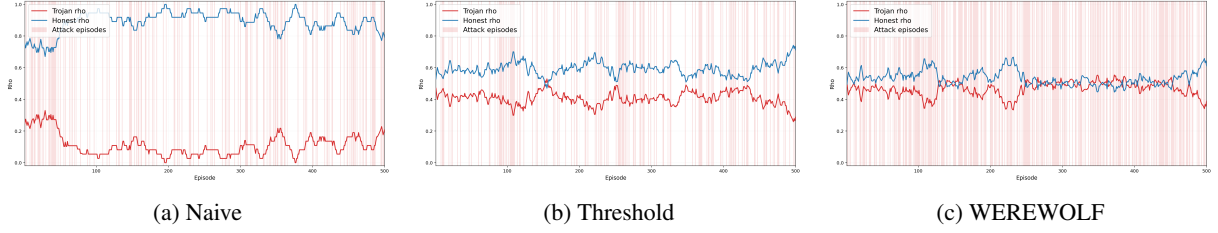


Figure 3: ρ_t^{proxy} for G-Designer over the first 500 episodes in GSM8K, where $\rho_t^{\text{proxy}} = \text{CES}_t^{\text{Trojan}} / (\text{CES}_t^{\text{Trojan}} + \text{CES}_t^{\text{Honest}} + \epsilon)$. The figure illustrates the attack–recovery dynamics of different attack strategies over time.

Table 3: Average damage, launch behavior, and within-mode survivability. R_{AIR} and R_{CES} report condition-matched Trojan-to-honest AIR/CES ratios.

Attack	$\Delta \text{ACC} \uparrow$	$\text{ASR} \uparrow$	Launch	$R_{\text{AIR}} \uparrow$	$R_{\text{CES}} \uparrow$
Naïve	42.8	84.4	73.6	0.92	0.78
Threshold	36.2	77.1	58.9	1.04	0.86
WEREWOLF	46.6	86.3	59.9	1.14	1.04

AIR and CES depend on topology density, we compare Trojan exposure with honest-agent within the same setting.

4.2 Main Result & Analysis

Data-Plane Vulnerability. Table 1 shows that a single Trojan agent can substantially degrade existing self-organizing LLM-MAS. Naïve reduces average accuracy from 72.3% to 29.5%, showing that selected Trojan agents are often not isolated effectively. This highlights a common vulnerability to adversarial in modern self-organizing architectures. WEREWOLF achieves stronger average degradation, reducing accuracy to 25.8% with an average ASR of 86.3%. WEREWOLF succeeds by selectively corrupting outputs while preserving future access.

Control-Plane Survivability. Tables 2 and 3 reveal how attack policies manage their control-plane presence over time. Although Naïve is selected frequently, its high selection rate fails to yield comparable topological exposure. This lower CES indicates that unconstrained disruption leads the controller to marginalize the Trojan even if it remains in the active set. In contrast, Threshold and WEREWOLF regulate their exposure through selective attack timing. While Threshold’s static rule can be overly conservative, WEREWOLF dynamically converts preserved topological influence into high-value attack opportunities. Crucially, high topological influence enables effective disruption because occupying a key position allows the Trojan to suppress benign inter-agent communication, thereby diluting uncorrupted context and magni-

Table 4: Effect of the survivability constraint on G-Designer-HumanEval under the Werewolf attack setting. CES and AIR are reported as Honest/Trojan; absolute CES values are interpreted within each setting.

Setting	ACC	ASR	Launch	AIR	CES
Clean	90.1	—	—	93.0 / 84.5	1.69 / 1.40
With Constraint	25.5	64.2	32.9	87.6 / 85.1	1.48 / 1.60
W/o Constraint	38.5	49.4	99.3	87.5 / 85.9	2.55 / 2.36

fying poisoned output propagation. Together, this yields a superior damage–survivability trade-off for WEREWOLF.

Damage–Survivability Trade-off. Figure 4 demonstrates that high data-plane damage does not fully measure an attacker’s persistence, as a Trojan is more dangerous when it sustains damage while remaining selected and influential. While Naïve launches aggressively but fails to preserve high-quality topological exposure, THRESHOLD sacrifices attack opportunities to recover structural access, and WEREWOLF strikes a better balance by selectively spending and replenishing control-plane exposure.

Periodic Attack–Recovery Behavior. Figure 3 provides a qualitative view of the ρ_t^{proxy} . Under Naïve, repeated attacks lead to a rapid decline in exposure. For Threshold, attacks are typically followed by long recovery intervals. Once recent exposure falls sufficiently below its online historical baseline, the Trojan behaves constructively until the normalized exposure score recovers above the fixed threshold. In contrast, WEREWOLF exhibits shorter and more frequent attack–recovery cycles. Despite attacking more frequent, its reputation proxy remains within a viable operating range and appears to return to this range more quickly after local drops. This behavior suggests that the learned policy performs a finer-grained temporal allocation of disruptive and constructive actions, repeatedly spending and replenishing control-plane exposure.

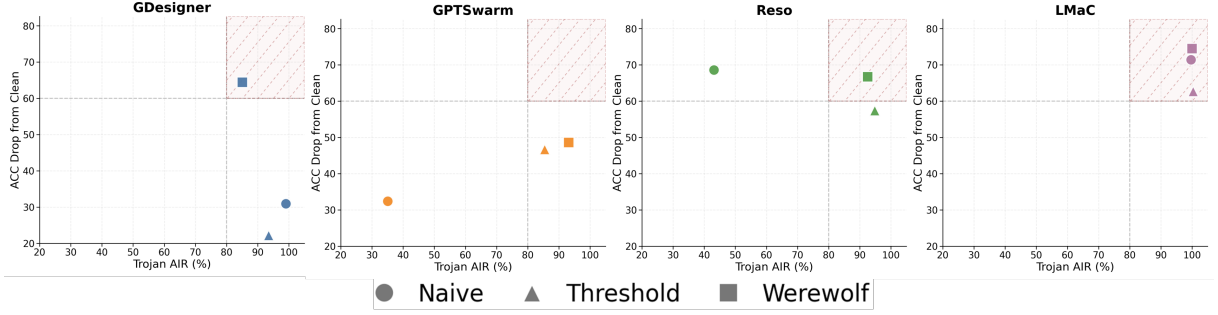


Figure 4: Damage-survivability trade-off on HumanEval. The y -axis shows ACC drop from Clean, and the x -axis shows Trojan-agent AIR. The upper-right region indicates stronger and more persistent attacks.

τ	ACC	ASR	Launch	CES	AIR
0.50	60.3	48.8	77.6	1.46/1.32	53.9/62.1
0.70	33.5	80.0	80.8	1.76/1.94	62.0/71.4
0.90	57.8	49.6	77.6	1.87/1.23	68.1/16.2
0.95	54.0	54.5	76.4	1.80/1.52	64.0/58.4
1.00	75.2	29.7	68.9	0.89/0.53	43.4/0.0
1.05	64.6	41.3	75.2	1.89/0.84	64.5/13.0
1.10 [†]	67.7	29.2	89.4	1.84/1.18	94.3/93.17
1.20	68.9	71.4	39.1	1.72/1.92	65.8/44.1
1.30	60.9	85.7	34.8	1.70/1.97	64.2/61.5
1.50	64.0	46.7	66.5	0.62/0.46	32.5/16.2

Table 5: Threshold sensitivity on G-Designer with HumanEval. CES and AIR are reported as Honest/Trojan. [†] denotes the fixed threshold used.

Ablation Study We conducted an ablation study on the Lagrangian term. As shown in Table 4, removing this term leads to more frequent attack launches, yet yields a lower ASR and a weaker degradation in task accuracy. Notice that the unconstrained Trojan maintains AIR and CES comparable to an honest agent, implying strong topological access alone is insufficient for effective, persistent disruption. Instead, the Lagrangian term acts as a selective filter, preventing the policy from wasting disruptive actions on low-value opportunities and thereby concentrating attacks when they inflict maximum task impact.

Threshold Sensitivity. The Threshold baseline is sensitive to τ because each early attack decision changes the feedback received by the controller and can alter later exposure. Table 5 reports a sensitivity study on G-Designer with HumanEval. The relationship is non-monotonic, and nearby thresholds can produce substantially different launch rates and downstream outcomes. We therefore use the fixed and untuned value $\tau = 1.1$ rather than selecting the threshold that maximizes attack performance.

4.3 Discussion on the Failure Mode

We evaluate randomized agent replacement as a simple defense against reputation-aware attacks

Attack	Def.	ACC	ASR	Launch	CES	AIR
Clean	D	90.1	—	—	1.69/1.40	93.0/84.5
	R	92.1	—	—	1.68/1.51	45.9/46.0
Naïve	D	59.0	38.1	99.4	1.77/1.29	92.9/99.4
	R	33.5	82.3	77.0	1.11/1.60	48.8/64.0
Threshold	D	67.7	29.2	89.44	1.84/1.18	94.3/93.17
	R	50.3	63.6	75.2	1.68/1.20	61.1/60.2
WEREWOLF	D	25.5	64.2	32.9	1.48/1.60	87.6/85.1
	R	31.0	50.0	45.3	2.04/1.62	65.0/70.2

Table 6: Randomized replacement defense on G-Designer with HumanEval. D and R denote Default (no defense) and Random replacement.

on G-Designer, which incorporates built-in mechanisms for adversarial robustness. After topology construction, each selected node is independently replaced with probability $\epsilon = 0.2$ by a benign reserve agent with the same role and model configuration. The reserve agent inherits the original edges, so the defense perturbs agent identity without changing graph density or introducing additional LLM calls. Table 6 compares the resulting data-plane and control-plane behavior. This illustrates the lack of effective robustness mechanisms in existing systems.

5 Concluding Remarks

This paper studies control-plane red-teaming in self-organizing LLM-MAS, showing that adaptive agent selection and topology formation introduce security risks beyond data-plane attacks. We formalize an internal Trojan as a reputation-signaling game, characterize its threshold and self-elimination properties, and introduce WEREWOLF, which balances immediate disruption against future access. Experiments across self-organizing MAS frameworks show that a single Trojan can substantially degrade performance while retaining near- or above-honest exposure, highlighting the need to defend against strategic, intermittent attackers.

Limitations

(1) We focus on the single-Trojan setting, leaving multiple attackers unexplored. (2) Our evaluation spans three benchmarks and four self-organizing LLM-MAS; generalization to other architectures remains uncertain. (3) WEREWOLF also assumes observable topology and feedback signals, which may be noisy in practice.

Ethical Considerations

This work studies a dual-use red-teaming problem for self-organizing LLM-based multi-agent systems. While the proposed attack model could be misused to design more persistent adversarial agents, our goal is to expose a previously underexplored control-plane vulnerability and motivate stronger defenses. We frame the method as a security evaluation tool for understanding how reputation-aware Trojan agents can preserve access, evade exclusion, and strategically disrupt task execution.

All experiments are conducted in controlled benchmark environments using standard public datasets and simulated Trojan agents. We do not attack live services, real users, private systems, or production deployments, and we do not use harmful real-world payloads. Any implementation of this work should be used only for authorized defensive evaluation, with appropriate safeguards and a clear focus on improving agent-level attribution, verification, and robust trust updating.

References

- Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2024. Chateval: Towards better llm-based evaluators through multi-agent debate. In *International conference on learning representations*, volume 2024, pages 9079–9093.
- Harsh Chaudhari, Giorgio Severi, John Abascal, Anshuman Suri, Matthew Jagielski, Christopher A. Choquette-Choo, Milad Nasr, Cristina Nita-Rotaru, and Alina Oprea. 2024. [Phantom: General backdoor attacks on retrieval augmented language generation](#). Preprint, arXiv:2405.20485.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. 2024a. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*, volume 2024, pages 20094–20136.
- Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. 2024b. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases. *Advances in Neural Information Processing Systems*, 37:130185–130213.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, et al. 2025. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems. *arXiv preprint arXiv:2508.07407*.
- Mohamed Amine Ferrag, Norbert Tihanyi, Djallel Hamouda, Leandros Maglaras, Abderrahmane Lakas, and Merouane Debbah. 2025. From prompt injections to protocol exploits: Threats in llm-powered ai agents workflows. *ICT Express*.
- Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. Not what you’ve signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM workshop on artificial intelligence and security*, pages 79–90.
- Pengfei He, Yuping Lin, Shen Dong, Han Xu, Yue Xing, and Hui Liu. 2025. Red-teaming llm multi-agent systems via communication attacks. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 6726–6747.
- Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xianwu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, et al. 2024. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pages 23247–23275.
- Mark Hoogendoorn and Jan Treur. 2009. An adaptive multi-agent organization model based on dynamic role allocation. *International journal of knowledge-based and intelligent engineering systems*, 13(3-4):119–139.
- Yue Hu, Yuzhu Cai, Yaxin Du, Xinyu Zhu, Xiangrui Liu, Zijie Yu, Yuchen Hou, Shuo Tang, and Siheng Chen. 2025. Self-evolving multi-agent collaboration networks for software development. In *International Conference on Learning Representations*, volume 2025, pages 23007–23039.

- Yuepeng Hu, Chongyu Fan, Sirat Samyoun, and Jian Du. 2026. Log-to-leak: Prompt injection attacks on tool-using llm agents via model context protocol.
- Donghyun Lee, Mo Tiwari, and Brando Miranda. 2025. Prompt infection: Llm-to-llm prompt injection within multi-agent systems. In *European Symposium on Research in Computer Security*, pages 511–520. Springer.
- Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for "mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008.
- Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 17889–17904.
- Lizhi Lin, Honglin Mu, Zenan Zhai, Minghan Wang, Yuxia Wang, Renxi Wang, Junjie Gao, Yixuan Zhang, Wanxiang Che, Timothy Baldwin, et al. 2025. Against the achilles' heel: A survey on red teaming for generative models. *Journal of Artificial Intelligence Research*, 82:687–775.
- Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2023. Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization. *arXiv preprint arXiv:2310.02170*.
- Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2024. Gaia: a benchmark for general ai assistants. In *International Conference on Learning Representations*, volume 2024, pages 9025–9049.
- Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. 2024. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 15174–15186.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.
- Michael Schillo, Petra Funk, and Michael Rovatsos. 2000. Using trust for detecting deceitful agents in artificial societies. *Applied Artificial Intelligence*, 14(8):825–848.
- Giovanna Di Marzo Serugendo, Marie-Pierre Gleizes, and Anthony Karageorgos. 2005. Self-organization in multi-agent systems. *The Knowledge engineering review*, 20(2):165–189.
- Rana Shahroz, Zhen Tan, Sukwon Yun, Charles Fleming, and Tianlong Chen. 2025. Agents under siege: Breaking pragmatic multi-agent llm systems with optimized prompt attacks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9661–9674.
- Jiawen Shi, Zenghui Yuan, Guiyao Tie, Pan Zhou, Neil Zhenqiang Gong, and Lichao Sun. 2025. Prompt injection attack to tool selection in llm agents. *arXiv preprint arXiv:2504.19793*.
- Bingyu Yan, Ziyi Zhou, Xiaoming Zhang, Chaozhao Li, Ruilin Zeng, Yirui Qi, Tianbo Wang, and Litian Zhang. 2025. [Attack the messages, not the agents: A multi-round adaptive stealthy tampering framework for llm-mas](#). *Preprint*, arXiv:2508.03125.
- Bin Yu and Munindar P Singh. 2003. Detecting deception in reputation management. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 73–80.
- Andy Zeng, Maria Attarian, Krzysztof Marcin Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael S Ryoo, Vikas Sindhwani, Johnny Lee, et al. 2023. Socratic models: Composing zero-shot multimodal reasoning with language. In *The Eleventh International Conference on Learning Representations*.
- Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. 2024. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10471–10506.
- Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. 2025a. Multi-agent architecture search via agentic supernet. *arXiv preprint arXiv:2502.04180*.
- Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Yu, and Tianlong Chen. 2025b. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. In *International Conference on Learning Representations*, volume 2025, pages 75389–75428.
- Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. 2025c. G-designer: Architecting multi-agent communication topologies via graph neural networks. In *International Conference on Machine Learning*, pages 76678–76692. PMLR.
- Heng Zhou, Hejia Geng, Xiangyuan Xue, Li Kang, Yiran Qin, Zhiyong Wang, Zhenfei Yin, and Lei Bai. 2025. Reso: A reward-driven self-organizing llm-based multi-agent system for reasoning tasks. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 15990–16009.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. Gptswarm: Language agents as optimizable graphs. In *Forty-first International Conference on Machine Learning*.

Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. 2025. PoisonedRAG: Knowledge corruption attacks to Retrieval-Augmented generation of large language models. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 3827–3844.