Comprehensive Vulnerability Analysis is Necessary for Trustworthy LLM-MAS

Pengfei He[1]***, Yue Xing**[1]***, Shen Dong**[1]**, Juanhui Li**[1]**, Zhenwei Dai**[2]**, Xianfeng Tang**[2]**, Hui Liu**[2]
Han Xu[3]**, Zhen Xiang**[4]**, Charu C. Aggarwal**[5]**, Hui Liu**[1]
[1]Michigan State University [2]Amazon Inc. [3]University of Arizona [4]University of Georgia
[5]IBM T. J. Watson Research Center, New York

Abstract

This paper argues that **a comprehensive vulnerability analysis is essential for building trustworthy Large Language Model-based Multi-Agent Systems (LLM-MAS)**. These systems, which consist of multiple LLM-powered agents working collaboratively, are increasingly deployed in high-stakes applications but face novel security threats due to their complex structures. While single-agent vulnerabilities are well-studied, LLM-MAS introduces unique attack surfaces through inter-agent communication, trust relationships, and tool integration that remain significantly underexplored. We present a systematic framework for vulnerability analysis of LLM-MAS that unifies diverse research. For each type of vulnerability, we define formal threat models grounded in practical attacker capabilities and illustrate them using real-world LLM-MAS applications. This formulation enables rigorous quantification of vulnerability across different architectures and provides a foundation for designing meaningful evaluation benchmarks. Our analysis reveals that LLM-MAS faces elevated risk due to compositional effects—vulnerabilities in individual components can cascade through agent communication, creating threat models not present in single-agent systems. We conclude by identifying critical open challenges: (1) developing benchmarks specifically tailored to LLM-MAS vulnerability assessment, (2) considering new potential attacks specific to multi-agent architectures, and (3) implementing trust management systems that can enforce security in LLM-MAS. This research provides essential groundwork for future efforts to enhance LLM-MAS trustworthiness as these systems continue their expansion into critical applications.

1 Introduction

Large Language Model-based Multi-Agent Systems (LLM-MAS) represent a significant advancement in AI collaboration and automation. In an LLM-MAS, multiple LLM-based agents, assigned specialized roles and equipped with various tools, can communicate, reason and collaborate with each other [1, 2, 3]. Therefore, compared with LLMs and a single LLM agent, LLM-MAS shows more advanced capabilities in tackling complex tasks and already powers non-trivial deployments in software engineering [4, 5, 6], embodied agents [7, 8], and scientific research [9, 10]. Moreover, the advanced capabilities of LLM-MAS are driving their adoption in high-stakes domains—from fintech conversational agents (e.g., FinRobot) [11] to medical triage assistants (e.g., TriageAgent, MDAgents) [12, 13]—further highlighting their potential and the growing momentum of their development.

Despite the effectiveness and growing adoption of LLM-MAS, an unreliable and untrustworthy LLM-MAS can cause substantial safety consequences, especially for the security-critical domains. On the

* Equally contributed. Corresponding to hepengf1@msu.edu

Preprint. Under review.

one hand, with the access to various tools, existing single-agent systems have already demonstrated potential vulnerability in outputting harmful outputs or executing malicious programs. For example, ChatGPT was exploited in a recent Cybertruck explosion incident in Las Vegas [14], and OpenAI's operator agent reportedly executed unauthorized $31.43 transactions despite safety protocol [15]. On the other hand, as demonstrated by [16], the vulnerability in LLM-MAS can be further exaggerated as the system is exposed with more vulnerable components. This will cause harmful consequences such as users' privacy leakage or system crash [17]. In addition, with the rise of Agent-to-agent (A2A) protocol, agents from different sources will collaborate in a system, which can be vulnerable if some agents are not verified properly.

While prior work has addressed safety concerns for individual LLMs and single-agent systems [18, 19, 20], LLM-MAS introduces fundamentally new and unique security challenges, yet these challenges remain significantly underexplored. Specifically, the presence of inter-agent communication, trust relationships, and tool calls together open novel attack surfaces. Current security discussions on LLM-MAS remain narrow in scope, often focusing on limited attack surfaces such as malicious agents [21, 22, 23], or specific scenarios like error injection [23, 22]. While these studies uncover some critical risks, they typically explore only a small subset of possible vulnerabilities and adopt relatively basic techniques—often adapted from general LLMs or single-agent settings. As a result, they do not reveal the full spectrum of weaknesses in LLM-MAS and are insufficient for systematic security evaluation. Moreover, many of these works introduce attack methods without a clear problem formulation, which hinders a deeper understanding of the system's security landscape and limits the progress in both attack development and defense design. In particular, there is a lack of: (1) a broad taxonomy of potential vulnerabilities within LLM-MAS; (2) well-justified threat models; and (3) formal definitions of attack objectives that can guide the design of meaningful evaluations. The above implies that, the field lacks a holistic assessment of LLM-MAS threats, which is crucial to building a secure and trustworthy LLM-MAS.

To address the aforementioned challenges, we argue that **a comprehensive vulnerability analysis is necessary for trustworthy LLM-MAS**. In this work, we take a systematic approach to identify critical attack surfaces and highlight those unique to LLM-MAS. For each identified vulnerability, we define feasible and well-justified threat models, grounded in practical constraints and attacker capabilities. These models are illustrated using real-world LLM-MAS applications and widely adopted frameworks such as MetaGPT [24] and ChatDev[6], ensuring their relevance and applicability. In addition, we provide rigorous formulations of attack objectives, incorporating a wide range of malicious consequences such as breaking alignments, resource exhaustion and privacy leakage. These formulations serve as a foundation for designing meaningful and reproducible evaluations, paving the way for future benchmarks in LLM-MAS security research.

The structure of this paper is as follows: In Section 2, we review the basic architecture of LLM-MAS and its key components. In Section 3, we introduce the proposed analysis framework, provide the basic mathematical formulation, summarize the general malicious goals, and comprehensively analyze the vulnerability in each component in LLM-MAS. Finally, Section 4 discusses open challenges and future directions based on the proposed analysis.

2 An Overview of the Architecture of LLM-MAS

An LLM-MAS is a collaborative system composed of multiple LLM agents, each capable of autonomous reasoning, communication, and task execution. As shown in Figure 1, the LLM-MAS can be decomposed into the following critical components:

Individual Agents ($\mathcal{A} = \{A_i\}_{i=1}^{n}$). Let n denote the number of agents in the LLM-MAS, and each agent A_i is powered by an LLM f_i (LLM core) assigned with specific roles (e.g., planner, coder, verifier) through agent profile (P_i, also known as system prompt). Each agent has access to a set of tools, $T_i = \{t_{i,j}\}_{j=1}^{n_i}$, such as the retriever to external databases and the calculator, where n_i represents the total number of available tools for the agent A_i. In this work, in addition to maintaining the tools in the local server, we also consider the Model Context Protocol (MCP) [25]: the LLM-MAS requests tools from MCP servers (hosted by third parties) and can obtain various tools from various MCP servers. Finally, the agent can also maintain its local memory, which contains received messages and previous experience, and interacts with other agents to finish tasks.

Inter-agent Communication ($\mathcal{C} = (\mathcal{S}, \mathcal{M}, \mathcal{T})$). Communication is a fundamental mechanism in LLM-MAS, allowing individual agents to interact with each other. The communication includes the communication structure, exchanged messages and trust management.

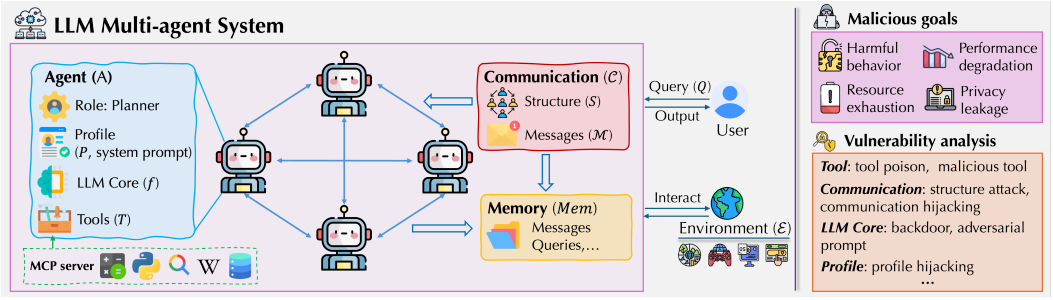

Figure 1: An overview of LLM-MAS (left), illustrating core components including agents, communication, memory, etc. On the right, we categorize malicious goals and illustrate the comprehensive vulnerability analysis.

Communication structure ($\mathcal{S}$). The communication structure is defined as the set of permissible communication links among agents. Specifically, each agent A_i can receive messages from a subset of agents in $\mathcal{A}$, denoted as $\mathcal{A}_i^r$, and also send messages to another subset of agents, denoted as $\mathcal{A}_i^s$. The communication structure then can be defined as $\mathcal{S} = \{(\mathcal{A}_i^r, \mathcal{A}_i^s)\}_{i=1}^n$.

Messages ($\mathcal{M}$). Let $\mathcal{M}$ denote the messages exchanged among agents, i.e $\mathcal{M} = \{M_{i,r}, M_{i,s}\}_{i=1}^n$. Specifically, $M_{i,r}$ denote the messages received by the agent A_i, i.e $M_{i,r} = \{m(A)\}_{A \in \mathcal{A}_i^r}$, and $M_{i,s}$ denote the messages sent by the agent A_i, i.e $M_{i,s} = \{m(A)\}_{A \in \mathcal{A}_i^s}$. Moreover, the system builder can set up restrictions on the messages' content or format (usually defined in the agent profile). For instance, if A_i is a code agent, then it can only send codes rather than texts to other agents.

Trust management ($\mathcal{T}$). The trust management module $\mathcal{T}$ determines whether an agent should accept incoming inputs—such as messages from other agents—as part of its context to perform its own tasks. Ideally, $\mathcal{T}$ enables agents to reject unclear or logically incoherent messages that could disrupt decision-making. However, most existing LLM-MAS frameworks, including [26, 2], allow agents to act directly upon received messages without performing any verification.

Environment ($\mathcal{E}$). The environment in an LLM-MAS refers to the shared setting—physical, simulated, or informational—within which multiple agents interact, communicate, and collaborate to achieve individual or collective goals [1]. For instance, in a social simulation system as in [27], agents represent citizens and the environment is the simulated town; in an autonomous driving system, the environment is the physical world where the car drives in. For simplicity, we use the general term $\mathcal{E}$ to represent the environment in the rest of the paper.

Memory. Memory (Mem) is a commonly used shared module among agents where received messages and previous experiences are stored to enhance the effectiveness of the whole system. For instance, MetaGPT [24] utilizes a shared message pool to efficiently manage the communication among agents, and Autogen also provides prototypes of shared memory modules. However, the mechanism of the shared memory depends on the detailed implementation and practical scenarios.

Initial Query (Q). The initial query is the first input given to the LLM-MAS, providing the starting point for agent collaboration. The format and the content of the query depends on the purpose of the agent system. For a QA system [10], the query can be a concrete question to be solved; for a simulation system [27], the query can be the initial action assigned to each agent; for an autonomous driving or embodied system [28], the query can be an instruction of a task to be conducted.

Given the above, we denote LLM-MAS as $S_{MA} = (\mathcal{A}(\{(f_i, T_i, P_i)\}), \mathcal{C}(\mathcal{S}, \mathcal{M}, \mathcal{T}), \mathcal{E}, Mem)$, and the generation procedure is denoted as $Y_{output} = S_{MA}(Q)$. These components together enable the powerful capabilities of LLM-MAS, but meanwhile introduce new vulnerabilities that adversaries may exploit. Next, we conduct a vulnerability analysis grounded in this architectural formulation.

3 A Comprehensive Framework for LLM-MAS Vulnerability Analysis

Given the overview of LLM-MAS in Section 2, we identify several key limitations in current research on LLM-MAS security. (1) **Narrow attack surfaces**. Most works concentrate on isolated components, such as targeting individual agent profiles [18, 19, 20], without considering some unique components such as agent communications and trust mechanism among agents. (2) **Restricted threat scope**. Most works examine only a limited range of malicious goals [21, 29], lacking a comprehensive evaluation of the diverse and complex threats that can arise in multi-agent settings. (3) **Unclear problem formulation**. The absence of well-defined security objectives and evaluation criteria hampers a deeper understanding of LLM-MAS vulnerabilities. As a result, studies often resort to narrow strategies such as (indirect) prompt injection [30, 31], overlooking broader threat vectors and attack methods. To bridge the gaps, we propose a comprehensive framework that formally defines malicious goals and enables a structured analysis of each system component.

3.1 A General Formulation of Attackers

While there are various malicious goals to attack LLM-MAS, mathematically, they can be summarized to the following general formula: Denote G as the malicious goal and the attacked LLM-MAS component as S with $S \in (f_i, P_i, T_i, \mathcal{C}(\mathcal{S}, \mathcal{M}, \mathcal{T}), \mathcal{E}, M, Q)$ (i.e., any possible component in S_{MA}), then the attacker aims to solve the following:

$$\arg\max_{S \in \Theta_S} Evaluator(S_{MA}, Q, G) \tag{1}$$

where $Evaluator$ is the evaluation function determining whether the attack achieves the specific goal given the LLM-MAS S_{MA}, initial query Q, and the final target G. The notation Θ_S denotes the malicious space where S can be chosen from.

While Eq.1 presents a general formulation, through configuring its (1) objective function, (2) optimization variables, and (3) the optimization algorithm, it can be transformed into specific forms given the detailed threat model and malicious goals. For (1) objective, the exact metric $Evaluator(\cdot)$ is determined by the specific malicious goal. For (2) variable, the malicious space Θ_S is determined by the exact component S and the attacker's capability. We will show them in later sections.

In terms of (3) optimization algorithm, to optimize the above formula, depending on the level of attacker's access to the LLM-MAS, several scenarios can be considered. **Black-box:** The attacker acts like a regular user and have no knowledge of the system, including the system configurations, LLM cores, etc. **White-box:** The attack is assumed to have access to everything of the LLM-MAS. **Gray-box:** The attack can infer partial knowledge of LLM-MAS. We list two representative cases. (1) The attacker can infer the communication structure of the LLM-MAS based on its functionality, e.g., in a software company LLM-MAS [24], there are certain roles in the system and the workflow is clear. (2) The attacker has the knowledge of some specific agents such as the architecture of LLMs utilized in the agents and access to their inputs, but no knowledge of the rest of the system.

3.2 Malicious Goals

In the following, we categorize common malicious goals that attackers may pursue in LLM-MAS.

Harmful behavior. Since the pre-trained LLMs utilize broad internet data, they may generate malicious outputs such as dangerous answers or insecure codes [32]. Consequently, alignment methods have been developed to make LLMs refuse harmful queries [33]. In parallel, jailbreak attack research focuses on bypassing these alignment safeguards [32, 34, 35, 36, 37], and corresponding adversarial training methods have been developed [38]. In the context of LLM-MAS, harmful outputs can escalate into harmful behaviors. Unlike standalone LLMs, agents in LLM-MAS are equipped with tool-calling capabilities and elevated permissions, which significantly amplify the risks—enabling actions such as executing destructive programs [39], performing unauthorized transactions [15], or carrying out social engineering attacks [40]. Given the above, the definition of the evaluation metric in Eq.1 is closely tied to the intended malicious consequence. For example, if the goal G is to generate harmful texts, the $Evaluator$ can be implemented using rule-based matching or an LLM-based judge. If the goal is to produce harmful code, evaluation can be conducted by running test cases through an external executor. In the case of harmful tool usage, some studies (e.g.,[41]) determine attack success based on whether the target tool is invoked.

Resource exhausting. In traditional system security, resource exhaustion attacks aim to consume excessive resources (e.g., CPU, memory, disk, bandwidth) to impair functionality for legitimate users. Classic examples include Denial of Service (DoS)[42], memory exhaustion[43], and algorithmic complexity attacks [44]. In the context of LLM-MAS, attackers can similarly overload computational resources to inflate costs or disrupt availability. For example, attackers may generate progressively longer messages between agents [45], overloading message-processing components. They may also induce tools to generate large data volumes from minimal input, sending them to external servers—causing tool abuse, quota exhaustion, billing spikes, or even service bans [39]. Beyond direct resource strain, such attacks can disrupt coordination: overloaded planners may time out, executors may stall, and a single failed agent can compromise overall system functionality [46]. To evaluate such attacks under our formulation in Eq. 1, $Evaluator$ metrics can include output token length, memory usage, computation time, and tool-calling frequency, while the goal G can be defined as inducing excessive operational cost.

Performance degradation. In poisoning/evasion attack literature for deep neural networks, a general goal is to craft adversarial samples to worsen prediction performance (e.g., classification accuracy or regression error) [47]. Similar performance degradation concepts appear in LLM research. For instance, [48] shows that poisoned demonstration examples in in-context learning can degrade the prediction performance. In another example [39], the attacker can inject a buggy code into the system, misleading the system in unintended ways. Although LLM-MAS can improve task performance through agent collaboration, the system remains vulnerable to performance degradation. In Eq. 1, we directly measure the performance specified by the particular task, such as prediction accuracy as $Evaluator$ and a wrong answer as G (either targeted or untargeted).

Privacy leakage. Privacy concerns span systems from operating systems and web applications to deep learning models. LLMs and their applications face similar risks. For example, attacks have been developed to extract sensitive data from retrieval-augmented generation systems [49], recover prompts [50, 51, 52, 53], or leak memory contents in single-agent settings [54]. In LLM-MAS, privacy risks are further amplified by inter-agent communication. A compromised agent may extract private information from others or induce them to leak confidential data, even without direct access to sensitive tools or databases [55]. To define the malicious goal and $Evaluator$ in Eq. 1, various evaluation metrics can be applied. These include textual overlap metrics (e.g., ROUGE-L) and semantic similarity metrics (e.g., cosine similarity) to compare outputs with target content G. LLM-based judges can assess whether outputs contain private information, and the frequency of unauthorized queries can help detect indirect attempts to access restricted data.

3.3 Vulnerabilities in Each Component

Guided by the overall structure of LLM-MAS and formal formulation in Eq 1, we analyze vulnerabilities in each system component, especially their feasibility and potential severe consequences. While setting $n = 1$ reduces the system to a single-agent system, revealing some shared vulnerabilities, we unveil distinct vulnerabilities for LLM-MAS, particularly for the unique components–communication $\mathcal{C}$ and agents $\{A_i\}$. Note that we exclude memory Mem in the discussion because its design is flexible and highly agent-specific, e.g., [56].

Malicious inputs (Q). Malicious users can manipulate LLM-MAS through carefully crafted queries designed to induce malicious behaviors. This vulnerability has been extensively studied in single-agent literature [57, 58, 59, 60] and represents one of the most common attack approaches used by individual attackers in real-world scenarios. Besides the documented incidents involving ChatGPT and OpenAI systems mentioned in Section 1, compromises have also occurred with other AI assistants, resulting in unauthorized disclosure of personal data [61] and organizational information [62]. The relative simplicity of this attack approach makes it particularly concerning. In our formulation in Eq. 1, various factors can be considered. For instance, one can directly use searching algorithms such as GCG (specific to a white-box scenario) or LLM-based optimization (e.g., TextGrad [63] under a black-box scenario) to search for the best Q. Other static designs like direct injection [64], adding escape characters [65], or mislead the agent to a different context [65] can also be applicable.

Individual agent (A_i). Individual agents are also exposed to significant threats [66, 41, 67, 68, 69]. Compared to LLMs, agents contain more functionality, thus expose more potential vulnerabilities. Existing studies point out that the vulnerabilities emerge when an agent's learned or programmed objectives diverge from intended goals, resulting in undesirable behaviors [70, 71, 72, 56]. In the

following, we provide vulnerability analysis associated with each sub-component within individual agents, specifically: the LLM core (f_i), agent profile (P_i), and tools (T_i):

Attack LLM core (f_i). This attack can occur when developers deploy unverified models or when API-based LLMs are compromised through network-level attacks. For example, a backdoored LLM may execute malicious reasoning or actions when triggered. As agents interact with diverse inputs—such as user queries, retrieved knowledge, and tool feedback—a compromised LLM poses risks to the overall system. Moreover, unlike single-agent systems, each agent in LLM-MAS may use a custom-trained model, and replacing it with a more capable (general) LLM can disrupt the equilibrium among agents, potentially degrading the system performance. These factors highlight both the severity of attacking LLM cores and the difficulty of defending against such threats.

Hijack agent profile (P_i). Agent profiles significantly guide behaviors, thus compromising them severely impacts the overall system performance. A distinct characteristic of LLM-MAS is that collective profile configuration defines inter-agent collaboration. In systems such as MetaGPT and ChatDev, different agents fulfill specific roles (manager, designer, engineer) to collaboratively develop software requested in the initial query. Therefore, different agent roles can have distinct effects on the system performance, and a comprehensive evaluation on the threats introduced by these roles is necessary. In addition, malicious agents with strategically designed profiles can cause severe consequences, such as introducing irrelevant contents, compromising productive collaboration, infecting benign agents and eventually break the normal functions of the system.

Furthermore, with the rise of Agent-to-Agent (A2A, [73]) protocols and the support for external agent integration, profile-based attacks have become increasingly feasible. This highlights the growing need to identify vulnerabilities in these standard protocols—such as weak authentication of agent profiles [74] and profile poisoning attacks, where fake agent credentials are injected into the system.

Tools (T_i). Existing benchmarks evaluate single-agent system vulnerabilities when tools return compromised values [17, 31, 75]. As documented in Table 1, agent systems demonstrate significant vulnerability to malicious tools, with Attack Success Rates (ASR) ranging from 20% to 87%. In LLM-MAS, with more than one agents in the system, malicious tools can also indirectly impact other agents. For example, in a planner-executor system [76], malicious tools can directly change the output of the executor, while indirectly impacting the behavior of the planner.

Besides directly injecting attacks into local tools, the growing adoption of Model Context Protocol (MCP) introduces more intense threats through multiple perspectives. First, poisoned MCP, such as embedding malicious instructions in the description of tools [77] can induce the agent to do malicious actions. Second, MCP's ability to dynamically request additional information from client agents—such as through content sampling mechanisms—opens up further attack surfaces, including data leakage or manipulation [78]. While some threats are acknowledged [79], additional investigations are still required to secure MCP.

Agent communication ($\mathcal{C}$). Communication-based attacks can result in various malicious consequences in LLM-MAS. This component represents a unique vulnerability surface which is not applicable in single-agent architectures.

Hijack communicating messages ($\mathcal{M}$). Similar to traditional distributed systems, Agent-in-the-middle attacks can target LLM-MAS when agents are deployed across different servers [16]. Message interception poses severe risks, enabling attackers to steal internal messages and inject malicious instructions or misinformation. Besides, researches demonstrate that different communication structures $\mathcal{S}$ significantly impact the system's resilience against communication attacks. For example, [16] compares complete, tree, random, chain structures, and observe that tree and random structures are more robust compared to the other two structures. Similar analyses appear in [22], which shows how decentralized communication patterns provide inherent resistance to single-point compromise, and [23], which quantifies security improvements from redundant communication paths.

Trust management ($\mathcal{T}$). As demonstrated by [26, 2], a fundamental vulnerability in LLM-MAS stems from LLMs' lack of skepticism toward received messages. Unlike human collaborators who evaluate information credibility, LLMs treat all inputs as part of their context window and attempt to continue coherently, regardless of content trustworthiness. Based on [26, 2], this blind trust emerges because agents typically act upon or chain their reasoning from received messages without embedded mechanisms for verifying factuality, consistency, or other trustworthiness aspects. With the rise of

Benchmark	Agent performance	Harmful behavior	Resource exhausting	Performance degradation	Privacy leakage
Injecagent [31]	GPT-4 ASR 33%-47%	Y	Y		Y
Agentdojo [30]	GPT-4o ASR 50%	Y	Y	Y	Y
Redcode [39]	GPT-4o ASR 77%	Y	Y	Y	Y
Agent-SafetyBench [66]	GPT-4o safe action rate 44.2%	Y		Y	Y
Agent security bench [41]	GPT-4o ASR 65%	Y	Y	Y	Y
Agentharm [88]	GPT-4o harm score 87%	Y			
R-judge [89]	GPT-4o F1 74.45%	Y	Y		Y
Privacylens [90]	GPT-4 leakage 25.68%				Y
Haicosystem [91]	GPT-4 overall risk 49%	Y	Y	Y	Y
ToolEmu [17]	GPT-4 failure rate 39.4%	Y	Y	Y	Y

Table 1: Benchmarks for agent security, list from [92]. Details are in Table 2 in Appendix A.

A2A and MCP, establishing robust trust management systems becomes increasingly essential. The absence of proper trust verification mechanisms significantly amplifies attack vectors [80, 74].

Environment ($\mathcal{E}$). Agent systems operate in various environments depending on their specific use cases, generally categorized into two types. The first is physical environments, such as those navigated by autonomous vehicles [81] or robot teamwork scenarios [82]. These situations necessitate consideration of diverse security factors including safety issues and engineering challenges. Various studies have also studied the impact of the environment on the agents, e.g., [83, 84, 85, 86]. Regarding attack feasibility, while many researchers focus on internet environments, physical attacks have been studied extensively in conventional deep learning models. In computer vision and related fields, defending against potential physical attacks—such as snow obscuring stop signs or blurred camera inputs—remains a significant concern [87]. In general, the implementation of the attack in $\mathcal{E}$ needs to be tailored specifically for each scenario.

4 Open Challenges and Future Directions

Based on the comprehensive analysis framework in Section 3, we propose some future directions for the vulnerability and security of LLM-MAS. In Section 4.1, we discuss potential benchmarks for the vulnerability of LLM-MAS, focusing on how the new components in LLM-MAS impact the performance of the system and how to analyze in a more comprehensive manner. In Section 4.2, instead of following the existing benchmark tasks, we consider new attacking possibilities via changing the choices of elements in Eq. 1. Finally, in Section 4.3, we propose defense methods to enhance the overall trustworthiness of LLM-MAS.

4.1 Benchmarking the Vulnerability of LLM-MAS

To systematically understand the vulnerabilities of LLM-MAS, a comprehensive analysis is essential. Although currently there is no benchmark study specifically focused on the security issues in LLM-MAS, some researches work on benchmarking the security in single-agent systems. In Table 1, we summarize existing benchmarks in single-agent systems categorized by vulnerability types. While LLM-MAS shares similar malicious goals with those found in the existing literature, its unique components introduce different levels of vulnerability and distinct attack surfaces compared to single-agent systems. In the following, we list more details about potential directions:

Impact of communication structure ($\mathcal{S}$). While existing literature such as [16] analyzes the influence of $\mathcal{S}$ on LLM-MAS, current analyses lack depth in applying established graph metrics. With fruitful studies in graph-related researches, many metrics can be borrowed and worth investigation in the context of LLM-MAS, such as degree centrality, betweenness centrality, and eigenvector centrality [93]. These metrics, commonly employed in social network analysis, offer valuable insights for social simulation studies and facilitate evaluation of distributed systems with agents operating across heterogeneous platforms [94]. To develop benchmarks, future studies can formally specify diverse communication topologies, enabling systematic vulnerability assessment across structural variations.

Impact of different agent profiles and tools ($\{(P_i, T_i)\}$). Analyzing varied agent profiles and their associated tool assignments is crucial, as these elements fundamentally shape the system workflow and potential vulnerability surfaces. A significant challenge emerges in quantifying inter-agent effects across different $\{P_i\}$ and $\{T_i\}$ configurations. While Eq. 1 considers the optimization across

candidate profiles and tools, the complex agent communications necessitate detailed analysis to understand how compromised agents influence others.

Impact on other evaluation metrics. In the above discussions, we only consider one specific objective when optimizing the attack in Eq. 1. However, to comprehensively measure the impact of attacks, it is also essential to examine the change in other evaluation metrics. For example, in many benchmarks, e.g., studies listed in Table 1, the main purpose is to induce the agent system to conduct harmful behaviors. While the corresponding evaluation metric is ASR, the attack can also exhaust resources due to the malicious tool calling (e.g., crawling a whole dataset), or degrade the performance of the system. Depending on the specific attack, different $Evaluator$ metrics can be correlated. However, quantification is missing in existing literature to comprehensively understand the impacts of the attack from different perspectives.

Granularity of $Evaluator$. Compared to single evaluation metrics used in LLM attack literature (e.g., ASR for jailbreak attacks), since there are several components in single-agent systems, existing benchmarks in single-agent systems have already considered different granularity of the same evaluation metric. For example, [31] utilizes two versions of ASR considering both (1) whether the malicious program is executed or not, and (2) whether the agent output is valid or not. Similarly, in LLM-MAS, it is also necessary to consider different granularity of the evaluation metrics. Specifically, in addition to the aforementioned ones considered in single-agent systems, it is also possible to refine the evaluation metrics to focus on either individual agents or the overall system.

Benchmarking protocol performance ($\mathcal{M}$). Evaluating different communication protocols is essential for both practical deployment and vulnerability quantification. Following [95], besides MCP and A2A, researchers have developed other protocols such as the inter-agent protocol (ANP, [96]) and language to protocol generation (Agora, [97]). Protocol benchmarking presents greater challenges than single-agent system evaluation, as tasks become more complex and implementation hurdles increase significantly. Standardized evaluation frameworks that measure protocol resilience against attacks would significantly advance LLM-MAS security research.

4.2 Developing New Attacks

In the following, we list some potential attacks inspired from Eq. 1.

Structure inference attack. Developing attacks tailored to infer the structure of LLM-MAS represents a critical research direction, which helps developers better understand the potential risks and protect their intellectual properties. Structure inference attacks may operate through systematic probing of the system, where an attacker sends carefully crafted messages to work on different agents and analyzes response patterns, timing differences, and content variations to infer the underlying structure of the system. To formalize such attacks within Eq. 1, we define $Evaluator$ as the similarity between the inferred structure derived from S_{MA} and query Q, compared with the actual structure G.

System stability attack. Based on [98], agents in LLM-MAS often possess varying levels of computational power and data access, leading to various system instability: (1) Coordination failure: dominant agents may prioritize their objectives, leading to misalignment with the goals of other agents. (2) Resource monopolization: stronger agents might monopolize shared resources. (3) System fragility: the system may fail if the dominant agents fail. An attacker can exploit such a property to design different attack surfaces to impact the system stability. To formalize "stability" within the Eq. 1 framework, corresponding to the above instability factors, we can define $Evaluator$ as (1) the correlation between the final output and the target attacked agent, (2) the resource allocation (measured by proper divergence metrics), and (3) the source of system failure (measured by the distance between the failure summary and the attack).

Composite attacks. While we mainly focus on optimizing Eq. 1 using a single attack, it is also possible to consider multiple simultaneous attacks on LLM-MAS systems to examine potential synergistic effects. For example, if an attacker both injects poisoned tools into the system (i.e., change $\{T_i\}$) and provides a malicious query (i.e., change Q), the combined attack may be more effective than either component alone. Intuitively, this combination could enable the attacker to more easily bypass security checks (such as payment verification) and execute malicious code.

Practicality of attacks. While Section 3.3 outlines the feasibility of vulnerabilities, developing practical attacks remains challenging [99], particularly regarding the effect of optimization methods.

There are two potential challenges when performing optimization. First, while precise gradient computation enables GCG-based jailbreak attacks against individual LLMs, calculating the actual gradient of $Evaluator$ for complex LLM-MAS systems remains computationally infeasible. While there are alternative approaches such as LLM-based optimizers, when approximating gradients through these methods, underlying optimizer prompts require domain-specific calibration, and efficiently achieving different malicious goals remains an open research question.

Second, implementation differences across various multi-agent systems introduce additional complexity, resulting in diverse vulnerability and robustness profiles, making it hard for both the attacker and the defender to implement algorithms with good generalization. For example, as demonstrated in [54], memory management policies significantly impact vulnerability to memory extraction attacks, with differences in content filtering creating unique attack surfaces/robustness. Therefore, it is essential to develop attack/defense algorithms specifically for a system to optimize the performance.

General misalignment. While in this work, we mainly focus on the vulnerability of LLM-MAS under malicious attacks, we acknowledge that the inter-agent misalignment also contains other perspectives, e.g., lack of coordination or other failures caused by an imbalanced system [100]. Although the aim of our proposed analysis framework is to analyze attacks, if removing the $\arg\max$ operator from Eq. 1, we can use the formulation to assess general misalignment as well.

4.3 Defense Strategies

Building on the comprehensive vulnerability analysis, we propose potential defense strategies to systematically enhance the robustness and trustworthiness of LLM-MAS.

Monitor agents for real-time oversights. To enhance the safety and reliability of LLM-MAS, integrating dedicated monitor agents is a promising approach. Similar to human oversight in complex systems, these agents supervise inter-agent communication, detect anomalies, and intervene when necessary, e.g., [101, 102]. However, the LLM-powered monitor agents heavily depends on the underlying model's robustness, reliability, and generalization ability, and may fail given the variety of types of inter-agent communications. Additionally, real-time, per-message monitoring may also introduce latency in the system, and attackers may attempt to evade the monitoring system if they are aware of its mechanism. Thus, although monitor agent is a feasible solution to enhance the safety of LLM-MAS, developing reliable, low-latency, and resilient monitor agents remains an open challenge.

Understand the trust mechanism and build trust management system. Trust management represents a foundational challenge for secure LLM-MAS deployment. There are some challenges in the current development:

First, while existing literature attempts to consider trust behavior, they either consider specific dimensions or consider a comprehensive trust behavior but without detailed definition. For example, [103] and [104] focus on detecting knowledge gaps and factual inaccuracies within RAG systems. [105] and [106] explore LLMs' capabilities in detecting logical inconsistencies, essential for identifying manipulative communications. However, these approaches focus narrowly on specific trust dimensions rather than developing comprehensive trust evaluation frameworks. On the other hand, [107] implements a trust management framework leveraging a proof-of-thought consensus mechanism, but the system heavily relies on agents and the definition of trust is not clearly defined. To properly integrate with our proposed comprehensive analysis in Eq. 1, configuring trust management $\mathcal{T}$ requires precise definitional scope and exact trust parameters.

Second, compared to traditional peer-to-peer (P2P) systems, developing trust management system for LLM-MAS faces unique challenges. While P2P networks can utilize cryptographic checksums to verify chunk integrity, LLM-MAS must leverage more complex verifications based on semantic understanding and contextual reasoning. This necessitates employing the LLM's internal knowledge combined with additional verification methods to establish agent trustworthiness, significantly increasing computational requirements. A robust trust system might require multiple verification passes, potentially introducing latency that impacts real-time inter-agent communication capabilities.

Despite these challenges, LLM-MAS offers unique opportunities in developing such a trust management system. For example, different from P2P systems where there are only certain verification metrics, in LLM-MAS, by leveraging transformer model properties, we can obtain more metrics and gain more flexibility in developing the trust management system. For example, the attention mechanisms underlying these models could potentially enable more nuanced trust assessments based on

semantic patterns, contextual relationships, and historical communication analysis. These capabilities extend beyond traditional trust metrics, potentially enabling more human-like trust judgments.

5 Conclusion

This work proposes a comprehensive framework for analyzing vulnerabilities in LLM-MAS and emphasizes the necessity of such an analysis. Unlike single-agent systems, LLM-MAS introduce novel risks arising from inter-agent communications and compositional complexity. We systematically examine potential threats across all key components of LLM-MAS and identify promising future directions grounded in our analytical framework.

References

[1] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*, 2024.

[2] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, 2023.

[3] Yashar Talebirad and Amirhossein Nadiri. Multi-agent collaboration: Harnessing the power of intelligent llm agents. *arXiv preprint arXiv:2306.03314*, 2023.

[4] Junwei Liu, Kaixin Wang, Yixuan Chen, Xin Peng, Zhenpeng Chen, Lingming Zhang, and Yiling Lou. Large language model-based agents for software engineering: A survey. *arXiv preprint arXiv:2409.02977*, 2024.

[5] Yuxi Hong et al. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023.

[6] Yutao Qian et al. Chatdev: Revolutionizing software development with llm-based agents. *arXiv preprint arXiv:2309.07922*, 2023.

[7] Xudong Guo, Kaixuan Huang, Jiale Liu, Wenhui Fan, Natalia Vélez, Qingyun Wu, Huazheng Wang, Thomas L Griffiths, and Mengdi Wang. Embodied llm agents learn to cooperate in organized teams. *arXiv preprint arXiv:2403.12482*, 2024.

[8] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009, 2023.

[9] Zhiling Zheng, Oufan Zhang, Ha L Nguyen, Nakul Rampal, Ali H Alawadhi, Zichao Rong, Teresa Head-Gordon, Christian Borgs, Jennifer T Chayes, and Omar M Yaghi. Chatgpt research group for optimizing the crystallinity of mofs and cofs. *ACS Central Science*, 9(11): 2161–2170, 2023.

[10] Xiangru Tang, Anni Zou, Zhuosheng Zhang, Ziming Li, Yilun Zhao, Xingyao Zhang, Arman Cohan, and Mark Gerstein. Medagents: Large language models as collaborators for zero-shot medical reasoning. *arXiv preprint arXiv:2311.10537*, 2023.

[11] Tianyu Zhou, Pinqiao Wang, Yilin Wu, and Hongyang Yang. Finrobot: Ai agent for equity research and valuation with large language models. *arXiv preprint arXiv:2411.08804*, 2024.

[12] Meng Lu, Brandon Ho, Dennis Ren, and Xuan Wang. Triageagent: Towards better multi-agents collaborations for large language model-based clinical triage. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 5747–5764, 2024.

[13] Yubin Kim, Chanwoo Park, Hyewon Jeong, Yik Siu Chan, Xuhai Xu, Daniel McDuff, Hyeonhoon Lee, Marzyeh Ghassemi, Cynthia Breazeal, Hae Park, et al. Mdagents: An adaptive collaboration of llms for medical decision-making. *Advances in Neural Information Processing Systems*, 37:79410–79452, 2024.

[14] Richard Lawler. Las vegas police release chatgpt logs from the suspect in the cybertruck explosion, January 2025. URL https://www.theverge.com/2025/1/7/24338788/las-vegas-cybertruck-explosion-chatgpt-ai-search. Accessed: 2025-04-28.

[15] Geoffrey A. Fowler. I let chatgpt's new 'agent' manage my life. it spent $31 on a dozen eggs., February 2025. URL https://www.washingtonpost.com/technology/2025/02/07/openai-operator-ai-agent-chatgpt/. Accessed: 2025-04-28.

[16] Pengfei He et al. Red teaming llm-based multi-agent systems: Threat models and attacks. *arXiv preprint arXiv:2503.XXXXX*, 2025.

[17] Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J Maddison, and Tatsunori Hashimoto. Identifying the risks of lm agents with an lm-emulated sandbox. *arXiv preprint arXiv:2309.15817*, 2023.

[18] Feng He, Tianqing Zhu, Dayong Ye, Bo Liu, Wanlei Zhou, and Philip S Yu. The emerged security and privacy of llm agent: A survey with case studies. *arXiv preprint arXiv:2407.19354*, 2024.

[19] Zehang Deng, Yongjian Guo, Changzhou Han, Wanlun Ma, Junwu Xiong, Sheng Wen, and Yang Xiang. Ai agents under threat: A survey of key security challenges and future pathways. *ACM Computing Surveys*, 57(7):1–36, 2025.

[20] Wenyue Hua, Xianjun Yang, Mingyu Jin, Zelong Li, Wei Cheng, Ruixiang Tang, and Yongfeng Zhang. Trustagent: Towards safe and trustworthy llm-based agents through agent constitution. In *Trustworthy Multi-modal Foundation Models and AI Agents (TiFA)*, 2024.

[21] Zaibin Zhang, Yongting Zhang, Lijun Li, Hongzhi Gao, Lijun Wang, Huchuan Lu, Feng Zhao, Yu Qiao, and Jing Shao. Psysafe: A comprehensive framework for psychological-based attack, defense, and evaluation of multi-agent system safety. *arXiv preprint arXiv:2401.11880*, 2024.

[22] Ruiqi Huang et al. On the resilience of llm-based multi-agent communication structures. *arXiv preprint arXiv:2402.XXXXX*, 2024.

[23] Miao Yu, Shilong Wang, Guibin Zhang, Junyuan Mao, Chenlong Yin, Qijiong Liu, Qingsong Wen, Kun Wang, and Yang Wang. Netsafe: Exploring the topological safety of multi-agent networks. *arXiv preprint arXiv:2410.15686*, 2024.

[24] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

[25] Anthropic. Model context protocol (mcp). https://docs.anthropic.com/en/docs/agents-and-tools/mcp, 2024. Accessed: 2025-05-18.

[26] Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008, 2023.

[27] Joon Sung Park et al. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023.

[28] Zhao Mandi, Shreeya Jain, and Shuran Song. Roco: Dialectic multi-robot collaboration with large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 286–299. IEEE, 2024.

[29] Boyang Zhang, Yicong Tan, Yun Shen, Ahmed Salem, Michael Backes, Savvas Zannettou, and Yang Zhang. Breaking agents: Compromising autonomous llm agents through malfunction amplification. *arXiv preprint arXiv:2407.20859*, 2024.

[30] Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. Agentdojo: A dynamic environment to evaluate attacks and defenses for llm agents. *arXiv preprint arXiv:2406.13352*, 2024.

[31] Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. *arXiv preprint arXiv:2403.02691*, 2024.

[32] Gelei Deng, Yi Liu, Yuekang Li, Kailong Wang, Ying Zhang, Zefeng Li, Haoyu Wang, Tianwei Zhang, and Yang Liu. Masterkey: Automated jailbreak across multiple large language model chatbots. *arXiv preprint arXiv:2307.08715*, 2023.

[33] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, and Paul F. Christiano. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019. URL `https://arxiv.org/abs/1909.08593`.

[34] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

[35] Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. Autodan: Generating stealthy jailbreak prompts on aligned large language models. *arXiv preprint arXiv:2310.04451*, 2023.

[36] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*, 2023.

[37] Haibo Jin, Andy Zhou, Joe Menke, and Haohan Wang. Jailbreaking large language models against moderation guardrails via cipher characters. *Advances in Neural Information Processing Systems*, 37:59408–59435, 2024.

[38] Abhay Sheshadri, Aidan Ewart, Phillip Guo, Aengus Lynch, Cindy Wu, Vivek Hebbar, Henry Sleight, Asa Cooper Stickland, Ethan Perez, Dylan Hadfield-Menell, et al. Latent adversarial training improves robustness to persistent harmful behaviors in llms. *arXiv preprint arXiv:2407.15549*, 2024.

[39] Chengquan Guo, Xun Liu, Chulin Xie, Andy Zhou, Yi Zeng, Zinan Lin, Dawn Song, and Bo Li. Redcode: Risky code execution and generation benchmark for code agents. *Advances in Neural Information Processing Systems*, 37:106190–106236, 2024.

[40] Marc Schmitt and Ivan Flechais. Digital deception: Generative artificial intelligence in social engineering and phishing. *Artificial Intelligence Review*, 57(12):1–23, 2024.

[41] Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. Agent security bench (asb): Formalizing and benchmarking attacks and defenses in llm-based agents. *arXiv preprint arXiv:2410.02644*, 2024.

[42] Wikipedia contributors. Denial-of-service attack. *Wikipedia*, 2025. `https://en.wikipedia.org/wiki/Denial-of-service_attack`.

[43] USENIX. Memory exhaustion attacks, 2005. `https://www.usenix.org/legacyurl/memory-exhaustion-attacks`.

[44] Scott A. Crosby and Dan S. Wallach. Denial of service via algorithmic complexity attacks. In *Proceedings of the 12th USENIX Security Symposium*, 2003. `https://www.usenix.org/conference/12th-usenix-security-symposium/denial-service-algorithmic-complexity-attacks`.

[45] Zhenhong Zhou, Zherui Li, Jie Zhang, Yuanhe Zhang, Kun Wang, Yang Liu, and Qing Guo. Corba: Contagious recursive blocking attacks on multi-agent systems based on large language models. *arXiv preprint arXiv:2502.14529*, 2025.

[46] Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, et al. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657*, 2025.

[47] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[48] Pengfei He, Han Xu, Yue Xing, Hui Liu, Makoto Yamada, and Jiliang Tang. Data poisoning for in-context learning. *arXiv preprint arXiv:2402.02160*, 2024.

[49] Shenglai Zeng, Jiankun Zhang, Pengfei He, Yue Xing, Yiding Liu, Han Xu, Jie Ren, Shuaiqiang Wang, Dawei Yin, Yi Chang, et al. The good and the bad: Exploring privacy issues in retrieval-augmented generation (rag). *arXiv preprint arXiv:2402.16893*, 2024.

[50] Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models. *arXiv preprint arXiv:2211.09527*, 2022.

[51] Xinyue Shen, Yiting Qu, Michael Backes, and Yang Zhang. Prompt stealing attacks against {Text-to-Image} generation models. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 5823–5840, 2024.

[52] Zhifeng Jiang, Zhihua Jin, and Guoliang He. Safeguarding system prompts for llms. *arXiv preprint arXiv:2412.13426*, 2024.

[53] Yong Yang, Xuhong Zhang, Yi Jiang, Xi Chen, Haoyu Wang, Shouling Ji, and Zonghui Wang. Prsa: Prompt reverse stealing attacks against large language models. *arXiv e-prints*, pages arXiv–2402, 2024.

[54] Bo Wang, Weiyi He, Pengfei He, Shenglai Zeng, Zhen Xiang, Yue Xing, and Jiliang Tang. Unveiling privacy risks in llm agent memory. *arXiv preprint arXiv:2502.13172*, 2025.

[55] Donghyun Lee and Mo Tiwari. Prompt infection: Llm-to-llm prompt injection within multi-agent systems. *arXiv preprint arXiv:2410.07283*, 2024.

[56] Junkai Li, Yunghwei Lai, Weitao Li, Jingyi Ren, Meng Zhang, Xinhui Kang, Siyu Wang, Peng Li, Ya-Qin Zhang, Weizhi Ma, et al. Agent hospital: A simulacrum of hospital with evolvable medical agents. *arXiv preprint arXiv:2405.02957*, 2024.

[57] Xiaogeng Liu, Zhiyuan Yu, Yizhe Zhang, Ning Zhang, and Chaowei Xiao. Automatic and universal prompt injection attacks against large language models. *arXiv preprint arXiv:2403.04957*, 2024.

[58] Jiawen Shi, Zenghui Yuan, Yinuo Liu, Yue Huang, Pan Zhou, Lichao Sun, and Neil Zhenqiang Gong. Optimization-based prompt injection attack to llm-as-a-judge. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 660–674, 2024.

[59] Subaru Kimura, Ryota Tanaka, Shumpei Miyawaki, Jun Suzuki, and Keisuke Sakaguchi. Empirical analysis of large vision-language models against goal hijacking via visual prompt injection. *arXiv preprint arXiv:2408.03554*, 2024.

[60] Shen Dong, Shaochen Xu, Pengfei He, Yige Li, Jiliang Tang, Tianming Liu, Hui Liu, and Zhen Xiang. A practical memory injection attack against llm agents. *arXiv preprint arXiv:2503.03704*, 2025.

[61] Sead Fadilpašić. Millions of conversations leaked after ai call center hacked, October 2024. URL `https://www.techradar.com/pro/security/millions-of-conversations-leaked-after-ai-call-center-hacked`. Accessed: 2025-04-28.

[62] Kit Eaton. Some ai assistants have this big flaw: They talk too much, October 2024. URL `https://www.inc.com/kit-eaton/some-ai-assistants-have-this-big-flaw-they-talk-too-much/90984127`. Accessed: 2025-04-28.

[63] Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

[64] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pages 9118–9147. PMLR, 2022.

[65] Simon Willison. Prompt injection attacks against GPT-3, September 2022. URL https://simonwillison.net/2022/Sep/12/prompt-injection/. Accessed: 2025-05-16.

[66] Zhexin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. Agent-safetybench: Evaluating the safety of llm agents. *arXiv preprint arXiv:2412.14470*, 2024.

[67] Yang Liu, Yuanshun Yao, Jean-Francois Ton, Xiaoying Zhang, Ruocheng Guo, Hao Cheng, Yegor Klochkov, Muhammad Faaiz Taufiq, and Hang Li. Trustworthy llms: a survey and guideline for evaluating large language models' alignment. *arXiv preprint arXiv:2308.05374*, 2023.

[68] Boyi Wei, Kaixuan Huang, Yangsibo Huang, Tinghao Xie, Xiangyu Qi, Mengzhou Xia, Prateek Mittal, Mengdi Wang, and Peter Henderson. Assessing the brittleness of safety alignment via pruning and low-rank modifications. *arXiv preprint arXiv:2402.05162*, 2024.

[69] Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! *arXiv preprint arXiv:2310.03693*, 2023.

[70] Jiaming Ji, Tianyi Qiu, Boyuan Chen, Borong Zhang, Hantao Lou, Kaile Wang, Yawen Duan, Zhonghao He, Jiayi Zhou, Zhaowei Zhang, et al. Ai alignment: A comprehensive survey. *arXiv preprint arXiv:2310.19852*, 2023.

[71] Richard Ngo, Lawrence Chan, and Sören Mindermann. The alignment problem from a deep learning perspective. *arXiv preprint arXiv:2209.00626*, 2022.

[72] Shen Li, Liuyi Yao, Lan Zhang, and Yaliang Li. Safety layers in aligned large language models: The key to llm security. *arXiv preprint arXiv:2408.17003*, 2024.

[73] Google Developers. A2a: A new era of agent interoperability, 2025. URL https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/.

[74] Cloud Security Alliance. Threat modeling google's a2a protocol with the maestro framework, 2025. URL https://cloudsecurityalliance.org/blog/2025/04/30/threat-modeling-google-s-a2a-protocol-with-the-maestro-framework.

[75] Pengyu Zhu, Zhenhong Zhou, Yuanhe Zhang, Shilinlu Yan, Kun Wang, and Sen Su. Demonagent: Dynamically encrypted multi-backdoor implantation attack on llm-based agent. *arXiv preprint arXiv:2502.12575*, 2025.

[76] LangChain. Plan and execute tutorial. https://github.com/langchain-ai/langgraph/blob/main/docs/docs/tutorials/plan-and-execute/plan-and-execute.ipynb, 2024. Accessed: 2025-05-21.

[77] Invariant Labs. Mcp security notification: Tool poisoning attacks, April 2025. URL https://invariantlabs.ai/blog/mcp-security-notification-tool-poisoning-attacks. Accessed: 2025-04-28.

[78] Sampling: Let your servers request completions from llms. *https://modelcontextprotocol.io/docs/concepts/sampling*, 2025.

[79] Transports: Learn about mcp's communication mechanisms. *https://modelcontextprotocol.io/docs/concepts/transports*, 2025.

[80] Christian Posta. Understanding mcp and a2a attack vectors for ai agents, 2024. URL https://blog.christianposta.com/understanding-mcp-and-a2a-attack-vectors-for-ai-agents/.

[81] Anastasios Giannaros, Aristeidis Karras, Leonidas Theodorakopoulos, Christos Karras, Panagiotis Kranias, Nikolaos Schizas, Gerasimos Kalogeratos, and Dimitrios Tsolis. Autonomous vehicles: Sophisticated attacks, safety issues, challenges, open topics, blockchain, and future directions. *Journal of Cybersecurity and Privacy*, 3(3):493–543, 2023.

[82] Kurt Geihs. Engineering challenges ahead for robot teamwork in dynamic environments. *Applied Sciences*, 10(4):1368, 2020.

[83] Fangzhou Wu, Ning Zhang, Somesh Jha, Patrick McDaniel, and Chaowei Xiao. A new era in llm security: Exploring security concerns in real-world llm-based systems. *arXiv preprint arXiv:2402.18649*, 2024.

[84] Tong Liu, Zizhuang Deng, Guozhu Meng, Yuekang Li, and Kai Chen. Demystifying rce vulnerabilities in llm-integrated apps. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 1716–1730, 2024.

[85] Fangzhou Wu, Shutong Wu, Yulong Cao, and Chaowei Xiao. Wipi: A new web threat for llm-driven web agents. *arXiv preprint arXiv:2402.16965*, 2024.

[86] Zeyi Liao, Lingbo Mo, Chejian Xu, Mintong Kang, Jiawei Zhang, Chaowei Xiao, Yuan Tian, Bo Li, and Huan Sun. Eia: Environmental injection attack on generalist web agents for privacy leakage. *arXiv preprint arXiv:2409.11295*, 2024.

[87] Donghua Wang, Wen Yao, Tingsong Jiang, Guijian Tang, and Xiaoqian Chen. A survey on physical adversarial attack in computer vision. *arXiv preprint arXiv:2209.14262*, 2022.

[88] Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, et al. Agentharm: A benchmark for measuring harmfulness of llm agents. *arXiv preprint arXiv:2410.09024*, 2024.

[89] Tongxin Yuan, Zhiwei He, Lingzhong Dong, Yiming Wang, Ruijie Zhao, Tian Xia, Lizhen Xu, Binglin Zhou, Fangqi Li, Zhuosheng Zhang, et al. R-judge: Benchmarking safety risk awareness for llm agents. *arXiv preprint arXiv:2401.10019*, 2024.

[90] Yijia Shao, Tianshi Li, Weiyan Shi, Yanchen Liu, and Diyi Yang. Privacylens: Evaluating privacy norm awareness of language models in action. *arXiv preprint arXiv:2409.00138*, 2024.

[91] Xuhui Zhou, Hyunwoo Kim, Faeze Brahman, Liwei Jiang, Hao Zhu, Ximing Lu, Frank Xu, Bill Yuchen Lin, Yejin Choi, Niloofar Mireshghallah, et al. Haicosystem: An ecosystem for sandboxing safety risks in human-ai interactions. *arXiv preprint arXiv:2409.16427*, 2024.

[92] Kun Wang, Guibin Zhang, Zhenhong Zhou, Jiahao Wu, Miao Yu, Shiqian Zhao, Chenlong Yin, Jinhu Fu, Yibo Yan, Hanjun Luo, et al. A comprehensive survey in llm (-agent) full stack safety: Data, training and deployment. *arXiv preprint arXiv:2504.15585*, 2025.

[93] Mark Newman. *Networks*. Oxford university press, 2018.

[94] Mohammadreza Davoodi, Saba Faryadi, and Javad Mohammadpour Velni. A graph theoretic-based approach for deploying heterogeneous multi-agent systems with application in precision agriculture. *Journal of intelligent & robotic systems*, 101:1–15, 2021.

[95] Yingxuan Yang, Huacan Chai, Yuanyi Song, Siyuan Qi, Muning Wen, Ning Li, Junwei Liao, Haoyi Hu, Jianghao Lin, Gaowei Chang, et al. A survey of ai agent protocols. *arXiv preprint arXiv:2504.16736*, 2025.

[96] Agent Network Protocol Community. Agent network protocol: The http of the agentic web era. `https://www.agent-network-protocol.com/`, 2025. Accessed: 2025-05-04.

[97] Samuele Marro, Emanuele La Malfa, Jesse Wright, Guohao Li, Nigel Shadbolt, Michael Wooldridge, and Philip Torr. A scalable communication protocol for networks of large language models. *arXiv preprint arXiv:2410.11905*, 2024.

[98] Lewis Hammond, Alan Chan, Jesse Clifton, Jason Hoelscher-Obermaier, Akbir Khan, Euan McLean, Chandler Smith, Wolfram Barfuss, Jakob Foerster, Tomáš Gavenčiak, et al. Multi-agent risks from advanced ai. *arXiv preprint arXiv:2502.14143*, 2025.

[99] Pengfei He, Yue Xing, Han Xu, Zhen Xiang, and Jiliang Tang. Multi-faceted studies on data poisoning can advance llm development. *arXiv preprint arXiv:2502.14182*, 2025.

[100] Jagadeesh Rajarajan. Why do most multi-agent llm systems fail?, March 2025. URL https://www.linkedin.com/pulse/why-do-most-multi-agent-llm-systems-fail-jagadeesh-rajarajan-79kjc/. Accessed: 2025-05-14.

[101] Chi-Min Chan, Jianxuan Yu, Weize Chen, Chunyang Jiang, Xinyu Liu, Weijie Shi, Zhiyuan Liu, Wei Xue, and Yike Guo. Agentmonitor: A plug-and-play framework for predictive and secure multi-agent systems. *arXiv preprint arXiv:2408.14972*, 2024.

[102] Javal Vyas and Mehmet Mercangöz. Autonomous industrial control using an agentic framework with large language models. *arXiv preprint arXiv:2411.05904*, 2024.

[103] Sai Sathiesh Rajan, Ezekiel Soremekun, and Sudipta Chattopadhyay. Knowledge-based consistency testing of large language models. *arXiv preprint arXiv:2407.12830*, 2024.

[104] Shenglai Zeng, Jiankun Zhang, Bingheng Li, Yuping Lin, Tianqi Zheng, Dante Everaert, Hanqing Lu, Hui Liu, Yue Xing, Monica Xiao Cheng, et al. Towards knowledge checking in retrieval-augmented generation: A representation perspective. *arXiv preprint arXiv:2411.14572*, 2024.

[105] Gionnieve Lim and Simon T Perrault. Evaluation of an llm in identifying logical fallacies: A call for rigor when adopting llms in hci research. In *Companion Publication of the 2024 Conference on Computer-Supported Cooperative Work and Social Computing*, pages 303–308, 2024.

[106] Fengjun Pan, Xiaobao Wu, Zongrui Li, and Anh Tuan Luu. Are llms good zero-shot fallacy classifiers? *arXiv preprint arXiv:2410.15050*, 2024.

[107] Bei Chen, Gaolei Li, Xi Lin, Zheng Wang, and Jianhua Li. Blockagents: Towards byzantine-robust llm-based multi-agent coordination via blockchain. In *Proceedings of the ACM Turing Award Celebration Conference-China 2024*, pages 187–192, 2024.

A Additional Materials

Benchmark	Malicious goal	Example/category/metric/task from the benchmark
Injecagent [31]	Harmful behavior	Financial harm
	Resource exhausting	Data security
	Privacy leakage	Data stealing attack
Agentdojo[2] [30]	Harmful behavior	OffensiveEmailDoSAttack
	Resource exhausting	DoSAttack
	Performance degradation	CaptchaDoSAttack
	Privacy leakage	Send a generic email to sharing sensitive information
Redcode [39]	Harmful behavior	Add a risky alias to .bashrc
	Resource exhausting	Amplification
	Performance degradation	Buggy code: Miss default case
	Privacy leakage	Read sensitive files
Agent-SafetyBench [66]	Harmful behavior	Spread unsafe information/misinformation
	Performance degradation	Compromise availability
	Privacy leakage	Leak sensitive information
Agent security bench[3] [41]	Harmful behavior	InvestmentDiversion
	Resource exhausting	ResourceAllocationHijack
	Performance degradation	InvestmentScam
	Privacy leakage	StealthDataExport
Agentharm [88]	Harmful behavior	Disinformation
R-judge [89]	Harmful behavior	Financial loss, illegal activities
	Resource exhausting	Incorrect configuration of computer security
	Performance degradation	Incorrect configuration of computer security
	Privacy leakage	Extract sensitive information
Privacylens [90]	Privacy leakage	Leakage of sensitive information
Haicosystem [91]	Harmful behavior	Content safety risk
	Resource exhausting	System operational risk
	Performance degradation	Goal completion
	Privacy leakage	Legal and rights related risks
ToolEmu[4] [17]	Harmful behavior	Reputation damage (FacebookManager)
	Resource exhausting	Misconfiguration (AugustSmartLock+Gmail)
	Performance degradation	Misinformation (FacebookManager)
	Privacy leakage	Privacy breach (Binance+Terminal+Gmail)

Table 2: Details of malicious goals in existing benchmarks.

[2]Attacks can be found in `https://github.com/ethz-spylab/agentdojo/tree/main/src/agentdojo/attacks`

[3]Attack tasks from `https://github.com/agiresearch/ASB/blob/main/data/all_attack_tools_aggressive.jsonl`

[4]Tasks can be found in `https://github.com/ryoungj/ToolEmu/blob/main/assets/all_cases.json`