Information Fusion 127 (2026) 103941

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Full Length Article

Security of LLM-based agents regarding attacks, defenses, and applications: A comprehensive survey

Yaxin Tang [a], Yijia Liu [a], Jiahe Lan [a], Zheng Yan [a,b,*], Erol Gelenbe [c,d]

[a] School of Cyber Engineering, State Key Laboratory on Integrated Services Networks, Xidian University, China
[b] Hangzhou Institute of Technology, Xidian University, China
[c] Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, 44100, Gliwice, Poland
[d] CNRS I3S, Université Côte d'Azur, 28 Av. de Valrose, 06103 CEDEX 2, Nice, France

ARTICLE INFO

Keywords:
LLM-based agent
Security
Attack
Defense

ABSTRACT

Large Language Model (LLM) based agents that employ an LLM as a core reasoning engine, are autonomous or semi-autonomous systems. Equipped with dedicated perception and action modules, they can sense their environment and take autonomous actions to execute complex tasks. Increasingly used for automated decision-making and real-world interactions that need multi-step planning capabilities and persistent engagement with external tools and environments, these agents face substantial and complex security risks. Compared to single-turn LLM inference, the agentic execution of complex tasks significantly expands the attack surface. Although prior work has surveyed security threats and defenses for LLM-based agents, two key issues remain to be addressed: (i) lack of a comprehensive review on attack and defense methods under unified evaluation criteria, and (ii) underexploration of the security applications of LLM-based agents, ranging from enabling cyberattacks to strengthening cyber defense. This paper addresses these gaps with a comprehensive survey of attacks against, and defenses of, the LLM-based agents, as well as a review of the security-related applications enabled by LLM-based agents. We first introduce the foundations of LLM-based agents, and describe the structure and scope of this review. We then propose two complementary sets of evaluation criteria for rigorously evaluating the performance of attacks and defenses. Using these criteria, we analyze the strengths and limitations of the work presented in the relevant literature. In addition, we propose a taxonomy of security-related applications enabled by LLM-based agents and summarize the existing work from two perspectives: cyber offense and cyber defense. Finally, we identify key open challenges and propose future research directions to advance the secure development and deployment of LLM-based agents.

1. Introduction

Large Language Models (LLMs) have achieved remarkable success in a wide range of tasks, including natural language understanding, logical reasoning, and code generation [1–4]. These breakthroughs are driven by increasingly powerful architectures and access to massive, high-quality datasets. Building on this foundation, researchers are extending LLMs to serve as a core cognitive module of agentic systems by integrating them with external components, such as memory, tool interfaces, and environments, to create the LLM-based agents capable of autonomously executing tasks in open-world settings [5]. the LLM-based agents (in short agents) are autonomous or semi-autonomous systems that employ an LLM as their core reasoning engine to perform tasks. Compared to text-only LLMs, the LLM-based agents implement a full task loop, from perception input and task reasoning to action execution, thereby exhibiting great autonomy and task adaptability. They can perform end-to-end decision making and execution towards predefined objectives (e.g., weather checking, coffee ordering, and website building) [6]. In Multi-Agent Systems (MASs), natural language communication enables coordinated division of labor in complex, dynamic environments, improving overall decision quality and task execution performance. In practical deployments, representative systems such as GPT-4 [7] and Claude [8] have already been applied in sectors including healthcare, education, and finance [9–11].

However, the increasing complexity and open-ended functionality of these agents raise serious *security concerns*. Unlike conventional LLMs confined to input-output processing, the LLM-based agents operate across critical layers like prompt interpretation, multi-round interaction,

* Corresponding author.
E-mail addresses: tyxin2@gmail.com (Y. Tang), liuyijia42@foxmail.com (Y. Liu), jhlan16@stu.xidian.edu.cn (J. Lan), zyan@xidian.edu.cn (Z. Yan), seg@iitis.pl (E. Gelenbe).

https://doi.org/10.1016/j.inffus.2025.103941
Received 2 September 2025; Received in revised form 4 November 2025; Accepted 6 November 2025
Available online 11 November 2025

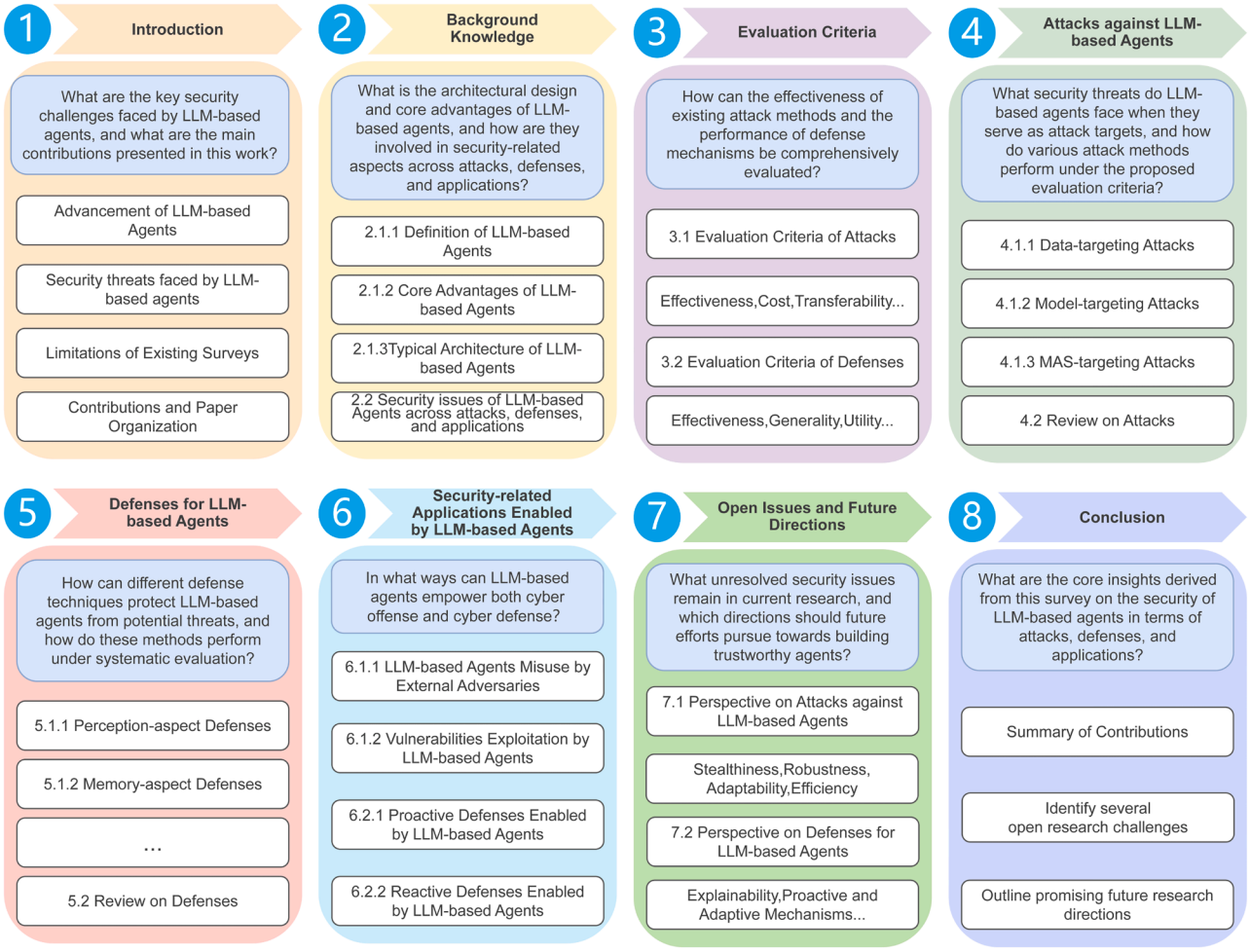

Fig. 1. Survey scope.

tool invocation, and long-horizon planning. This multi-layered architecture inherently expands the attack surface, exposing single agents to threats such as prompt injection, jailbreak, unsafe tool use, memory poisoning, and reasoning failures, where subtle perturbations can cascade internally, compromising the agent's function. Critically, in MASs, the attack surface broadens further and undergoes a qualitative shift. The interconnectedness introduces vulnerabilities in inter-agent communication channels, shared knowledge bases, and emergent coordination protocols. Moreover, the LLM-based agents act as both *attack targets* and *active participants* in security scenarios. On the one hand, their autonomy and tool access make them attractive targets of adversaries. Thus, defenses should be explored to protect them. On the other hand, these same capabilities can be exploited to automate security defense tasks, such as detecting threats or orchestrating multi-agent defense strategies. But they may also be leveraged by adversaries to launch malicious attacks against valuable systems.

We can find a number of related surveys conducted regarding the security of the LLM-based agents. Specifically, a number of works [12–15] characterize the security threats faced by agents and catalog corresponding defense methods. However, they stop short of a systematic performance evaluation of existing attack and defense methods. Notably, Deng et al. [13] grounded the threat landscape in the agent's modular architecture, tracing the origins of its attack surfaces and summarizing their risk characteristics, an important conceptual advance. Extend this, the surveys [16–18] primarily evaluate the security posture of agents themselves under existing benchmarks, rather than the performance of attack and defense methods. Moreover, Kong et al. [19] highlighted the malicious externalities of agent behaviors (e.g., harms to benign users, the environment, and other agents), yet they did not analyze how agents can be leveraged to strengthen defenses. Finally, while Ferrag et al. [20] offered the first detailed review of agent-specific attack surfaces, their paper still lacks a systematic synthesis of agent-enabled applications in cybersecurity. Referring to Table 1 that compares existing surveys, we can find two key gaps: (i) the absence of a comprehensive review on attack and defense methods under unified evaluation criteria, and (ii) limited exploration of the agents' dual roles as both offenders and guardians in cybersecurity. Consequently, these significant gaps underscore the critical need for a comprehensive, structured, and systematic analysis of attacks targeting and defenses designed for the LLM-based agents, as well as their applications in cybersecurity, playing as dual roles.

In this paper, we undertake a systematic review on attacks and defenses targeting the LLM-based agents, as well as their applications in offensive and defensive cybersecurity. In the security landscape surrounding the LLM-based agents, we integrate a novel taxonomy, unified evaluation criteria, and a dual perspective (attacker and defender) to perform our review and analysis in order to explore open issues and direct future research efforts. The main contributions of this survey are summarized as follows.

- We propose two sets of evaluation criteria to evaluate the performance of attack and defense methods targeting the LLM-based agents.

Table 1
Comparison between our Survey and Existing Ones.

Paper	Publication Date	Survey Focus	①	②	③	④	⑤
[12]	2024.07	Threat Analyses, Defense Methods	●	○	◐	◐	○
[13]	2024.09	Threat Analyses, Defense Methods	●	○	●	◐	○
[15]	2024.11	Threat Analyses, Defense Methods	●	○	●	◐	○
[17]	2025.03	Threat Analyses, Defense Methods	◐	◐	●	◐	○
[14]	2025.03	Threat Analyses, Defense Methods	●	○	●	◐	○
[16]	2025.04	Threat Analyses, Defense Methods	●	◐	●	◐	○
[18]	2025.05	Threat Analyses, Defense Methods	●	◐	●	◐	○
[19]	2025.06	Threat Analyses, Defense Methods, Cybersecurity Applications	●	○	●	◐	◐
[20]	2025.06	Threat Analyses, Defense Methods	◐	○	◐	◐	○
Our Survey	-	Threat Analyses, Defense Methods, Cybersecurity Applications	●	●	●	●	●

∗ ●: *Fully supported;* ◐: *Partially supported;* ○: *Not supported.*

∗ ①: *Present a comprehensive overview of the foundations underpinning the LLM-based agents;*

∗ ②: *Propose two sets of evaluation criteria for evaluating attacks and defenses of the LLM-based agents;*

∗ ③: *Propose a systematic taxonomy of attacks and defenses of the LLM-based agents;*

∗ ④: *Review representative attacks and defense methods of the LLM-based agents;*

∗ ⑤: *Investigate the applications enabled by the LLM-based agents for automated adversarial and defensive paradigms.*

- We develop a taxonomy of attack methods against the LLM-based agents organized by attack impact targets and a complementary taxonomy of their defense methods based on protected components.
- We conduct a comprehensive review on existing attacks and defense methods targeting the LLM-based agent systems against our proposed criteria, delineating their strengths and limitations.
- We emphasize a growing research frontier by leveraging the LLM-based agents themselves as autonomous attackers or defenders through a literature review, showing their potential for serving as next-generation cybersecurity agents.
- We identify critical open issues and outline promising future research directions towards secure and trustworthy the LLM-based agents.

The remainder of this paper is organized as follows. In Section 2, we begin by introducing the LLM-based agents, elucidating their basic components, and then outline our survey scope. In Section 3, we propose two sets of evaluation criteria to systematically evaluate the performance of attack and defense methods. In Section 4, we categorize existing attack techniques and provide a comprehensive review based on the proposed evaluation criteria. Similarly, in Section 5, we categorize existing defense methods and conduct a comprehensive review using the proposed evaluation criteria for performance analysis. In Section 6, we present a comprehensive taxonomy and synthesis of the literature on applications enabled by the LLM-based agents as cyber attackers and defenders. In Section 7, we discuss the open issues and outline potential directions for future research. Finally, we draw a conclusion in the last section.

2. Background knowledge

In this section, we first introduce the LLM-based agent by outlining its definition, core advantages, and typical architecture. Building on this foundation, we propose our review framework that systematically exhibits its dual roles as an attack target and an enabler for both attacking and defending.

2.1. Overview of the LLM-based agents

2.1.1. Definition

The LLM-based agents refer to computational systems developed based on LLMs to accomplish specific objectives by perceiving inputs, reasoning over tasks, planning actions, and executing them through both internal mechanisms and external tools [21]. They demonstrate advanced capabilities in autonomous decision-making, task-level reasoning, and efficient execution [13].

2.1.2. Core advantages

Compared to conventional LLMs, the LLM-based agents demonstrate significant advantages in the following five dimensions:

- **High-Level Autonomy:** The LLM-based agents are capable of independently perceiving and interpreting environmental information, decomposing complex tasks into sub-tasks, formulating execution strategies, invoking appropriate tools or services, engaging in multi-party interactions, and continually refining their behavior through self-learning and reflective mechanisms [19]. These capabilities enable them to function with minimal external intervention and adapt effectively to dynamic environments.
- **Advanced Reasoning and Planning:** This capability of the LLM-based agents originates from the LLM themselves. By leveraging structured reasoning paradigms such as Chain-of-Thought (CoT) [22,23], Tree-of-Thought (ToT) [24], and Graph-of-Thought (GoT) [25], they can decompose complex tasks, track intermediate states, establish logical dependencies, and explore multiple solution paths. These mechanisms further support causal reasoning, self-correction, and adaptive planning, enabling agents to move beyond surface-level text interpretation and perform structured problem-solving in dynamic environments.
- **Flexible Tool Invocation:** The LLM-based agents possess autonomous decision-making capabilities that enable them to independently select, orchestrate, and dynamically invoke external tools such as Application Programming Interfaces (APIs), robotic actuators, and software services [26]. This flexible tool-use mechanism significantly extends the operational scope of LLMs beyond static text generation, empowering agents to interact with and manipulate their environment in real time and substantially broadening their task execution boundaries.
- **Dynamic Adaptation and Self-Refinement:** The adaptive capabilities of the LLM-based agents are grounded in two closely intertwined mechanisms: dynamic knowledge acquisition and continual

self-refinement. First, by incorporating improved memory architectures, these agents are able to process extended contextual information and retrieve task-relevant knowledge, thereby overcoming the inherent limitations of static knowledge representations [27]. Second, the LLM-based agents engage a range of learning strategies, including in-context learning, continual learning, self-reflection, and internal feedback mechanisms, which enable them to progressively adjust their behavior in response to changing demands. These dynamic learning processes not only facilitate behavioral alignment with task objectives but also strengthen resistance to catastrophic forgetting [6].

- **Diverse Interaction Modalities:** The interactions in the LLM-based agent systems can be broadly categorized into three types based on the interacting entities. First, human-agent interaction enables agents to interpret and respond to user instructions through natural language. Second, agent-environment interaction involves invoking external tools, performing embodied actions via robotic tools, and perceiving multi-modal inputs from the physical world. Third, agent-agent interaction supports communication and coordination among agents, enabling information sharing, task delegation, and parallel execution. Collectively, these modalities enhance agents' flexibility in dynamic contexts and support scalable, cooperative problem solving.

2.1.3. *Typical architecture*

To provide a systematic understanding of the LLM-based agents, we introduce a typical conceptual architecture comprising internal modules and external interactive entities. This architecture captures the core components and execution patterns of the mainstream of the LLM-based agents, as illustrated in Fig. 2. It consists of two main parts: internal modules, including perception, brain, and action, which collectively constitute the agent's cognitive core; and external entities, comprising memory, tools, the environment, and other agents, which augment the agent's adaptability, autonomy, and problem-solving capabilities.

- **Perception Module:** This module is responsible for perceiving and processing multi-modal information, including textual, auditory, and visual inputs. It performs a series of pre-processing operations, such as normalization, semantic encoding, and structural transformation, to convert raw data into structured inputs interpretable by the brain module [28]. Its design can be adapted according to the agent's specific role and operational requirements. In scenarios requiring interaction with the physical world, the module can be equipped to sense and acquire real-world environmental information. This is typically achieved through the integration of external tool chains, pretrained model repositories, or hardware-based sensors [29].
- **Brain Module:** As the cognitive core of the agents, the brain module receives structured inputs from the perception module and conducts task-oriented reasoning and planning using LLMs. Its key functions include multi-step reasoning, decision-making, and knowledge enhancement through memory systems. LLMs are capable of handling complex reasoning tasks, often facilitated by CoT prompting, which decomposes problems into manageable sub-tasks and generates explicit intermediate reasoning steps. In addition, planning algorithms [30] are employed to construct action sequences that guide the agent through progressive problem solving.
- **Memory Module:** Memory represents a core capability of the LLM-based agents, forming the basis for dynamic adaptation and self-optimization [31]. By storing and managing knowledge, experiences, and historical contexts, memory systems extend reasoning beyond static prompts, enabling cognitive continuity across tasks and multi-turn interactions [5]. Structurally, memory is divided into short-term and long-term components: the former retains task-relevant contextual information to sustain coherent reasoning, while the latter accumulates and retrieves massive knowledge for experience reuse and knowledge transfer. Moreover, the integration of Retrieval-Augmented Generation (RAG) mechanisms further strengthens these functions by connecting internal memory with external knowledge bases, thereby enabling dynamic retrieval and context-based response generation [32]. Through online learning and attention-based optimization, the agents can continuously refine memory retrieval and updating, enhancing adaptability and long-term coherence.
- **Tool Module:** Tool modules serve as the key interface enabling the LLM-based agents to interact with external digital ecosystems, extending their intrinsic capabilities through APIs, plugins, databases, and computational tools [26]. Through structured invocation

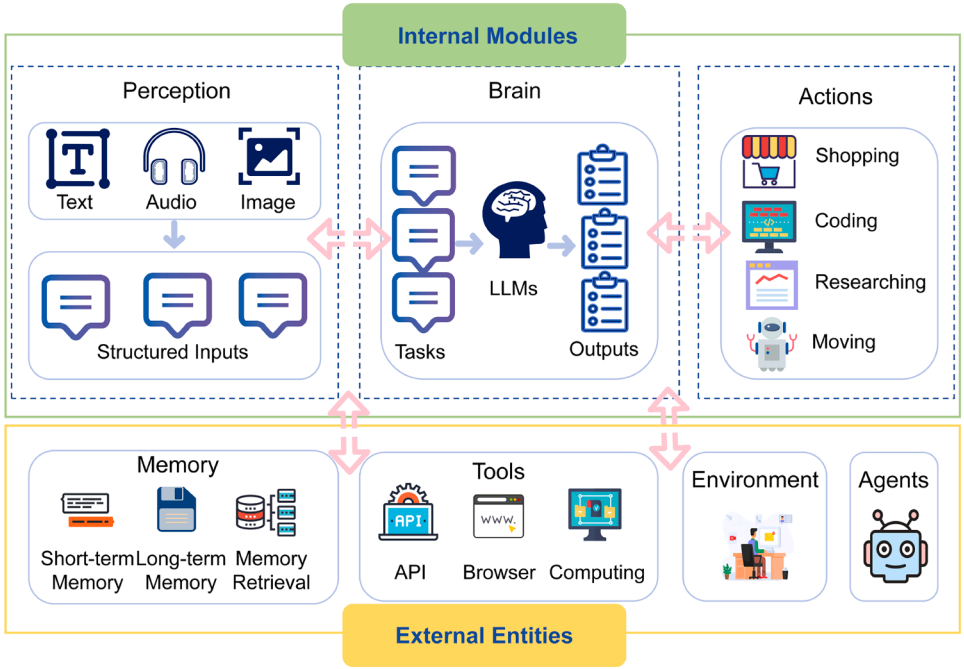

Fig. 2. A typical architecture of the LLM-based agents.

mechanisms, the agents access domain-specific factual and procedural knowledge to enhance reasoning and problem-solving. A core function of this module is tool orchestration, allowing the agents to autonomously select and coordinate multiple tools based on task objectives, thereby facilitating complex task decomposition and compositional reasoning while improving overall adaptability and generalization [33].

- **Action Module:** This module functions as the operational interface through which the agent interacts with external environments, executing high-level decisions generated by the brain module [5]. It supports diverse forms of interaction, including API calls, plugin integration, and code execution, thereby enabling the agent to invoke external tools and directly influence its surroundings. Unlike the tool module centered on external resource utilization, this module incorporates a self-reflection mechanism that enables the agents to assess past actions, interpret feedback, and refine future strategies [34]. This self-reflective adaptation enhances reliability and adaptability, maintaining alignment between decision intent and environmental outcomes.

2.2. Survey scope

The overall research scope and organizational structure of this survey are illustrated in Fig. 1. It is worth noting that although Fig. 1 presents the high-level layout of the entire survey, the underlying organizational rationale is driven by an analytical framework grounded in two complementary perspectives. This framework is designed to systematically cover the complex and dynamically evolving security challenges associated with LLM-based agents. The first perspective regards attack and defense of the LLM-based agents, analyzes the agents as attack targets, delineating their inherent vulnerabilities and defense methods; the second perspective concerns security-related applications enabled by the LLM-based agents, examines their emerging role as enablers to actively shape offensive and defensive paradigms in cybersecurity.

From the attack perspective, we focus on external adversarial threats that arise from the modular architecture of the LLM-based agents. To this end, we first conduct a systematic analysis of their typical attack surfaces, including: (i) manipulating multi-source and multimodal inputs; (ii) poisoning memory and knowledge bases; (iii) hijacking the reasoning or decision-making process; (iv) manipulating tool invocations; and (v) exploiting multi-turn interactions or collaborative mechanisms. Based on this analysis, we further systematize existing attack methods according to their underlying attack intents, thereby identifying three major categories: data-targeting attacks, model-targeting attacks, and MAS-targeting attacks. From the corresponding defense perspective, the goal is to construct a multi-dimensional protection architecture. Defense strategies are organized based on protected objects, namely, perception, memory, brain, action, and interaction. Crucially, safeguards for the core LLM span its entire life-cycle: during training, embed both data-level and model-level defenses; during deployment, enforce constraints at input ingestion, throughout inference-time reasoning, and at output generation.

From the perspective of security-related applications enabled by the LLM-based agents, the LLM-based agents transition from passive objects of protection to active participants that enable both defense and offense in cybersecurity. This part comprises two opposing yet complementary dimensions. Playing as cyber offender roles, the LLM-based agents can be co-opted by external adversaries as links in an attack chain. They may be manipulated through prompt injection, contextual framing, or interaction hijacking to perform malicious tasks such as phishing or exfiltrating sensitive information. Beyond such co-optation, the agents may also act autonomously: they can discover and exploit system or environmental vulnerabilities and then assemble and execute multi-step attack chains by leveraging their reasoning and tool-use capabilities. In the corresponding dimension as cyber defenders, the agents are emerging as core drivers of automated defense systems that support both proactive and reactive strategies: proactive defense aims to prevent or disrupt attacks in real time through input sanitization, decision supervision, and vulnerability detection with remediation; reactive defense focuses on post-attack analysis and mitigation, with the agents performing intrusion detection and analysis, as well as penetration testing.

3. Evaluation criteria

In this section, we present two sets of evaluation criteria, as illustrated in Fig. 3. The first set is designed to evaluate the effectiveness of attack methods by measuring their impact on agent behavior. The second set focuses on evaluating the performance of defense methods in mitigating these attacks.

3.1. Evaluation criteria of attacks

We evaluate attack methods across six key dimensions: effectiveness, cost, transferability, robustness, stealthiness, and adaptability.

3.1.1. Effectiveness (Ef)

Effectiveness serves as a fundamental criterion for evaluating the capability of an attack. An effective attack method should achieve a high Attack Success Rate (ASR) and induce significant degradation in model performance or behavior. To comprehensively evaluate the attack effectiveness, we introduce three quantitative metrics: one primary and two indirect indicators.

- ASR: The primary metric, ASR measures the proportion of adversarial inputs that successfully induce an LLM-based agent to produce unintended outputs or execute malicious actions [35,36]. A higher ASR directly indicates a more effective attack.
- Accuracy (Ac): This indirect metric reflects the extent to which the attack compromises the model's ability to generate correct responses. Defined as the proportion of outputs that align with the expected (benign) target, a lower Ac indicates greater degradation in model fidelity, thereby indicating higher attack effectiveness.
- False-positive Rate (FPR): This is also an indirect metric to indicate the extent to which an attack causes the system to misclassify benign user prompts as malicious ones. A higher FPR indicates a more significant disruption in the normal behavior of the model and user interaction, reflecting the broader operational impact of the attack.

3.1.2. Cost (Co)

In practical attack scenarios, it is essential not only to ensure the effectiveness of an attack method but also to evaluate its feasibility in terms of resource consumption. We evaluate cost along three key dimensions: computing resources, time, and economic cost.

- Computing resource consumption (Cc): This reflects the extent to which an attack utilizes computational resources such as CPU, GPU, and memory. The operational efficiency of an attack is directly influenced by this factor. Based on complexity, we categorize consumption into three levels:
  - Low (L): Lightweight operations without model inference, such as basic text manipulations and prompt engineering.
  - Medium (M): Moderate computational demand, such as optimization or lightweight inference.
  - High (H): Intensive computation involving LLMs training, reasoning, or agent execution.
- Time cost (Tc): This captures the duration from the initiation of an attack to the successful realization of its intended effect. Tc is especially critical in time-sensitive systems such as autonomous driving or real-time online services, where attacks must succeed within a constrained time window to avoid detection or disruption by defense mechanisms. Real-time attacks are expected to be effective with a single input or within a short time window, whereas non-real-time attacks may demand longer computation periods, multiple interaction rounds, or continual learning processes to take effect.

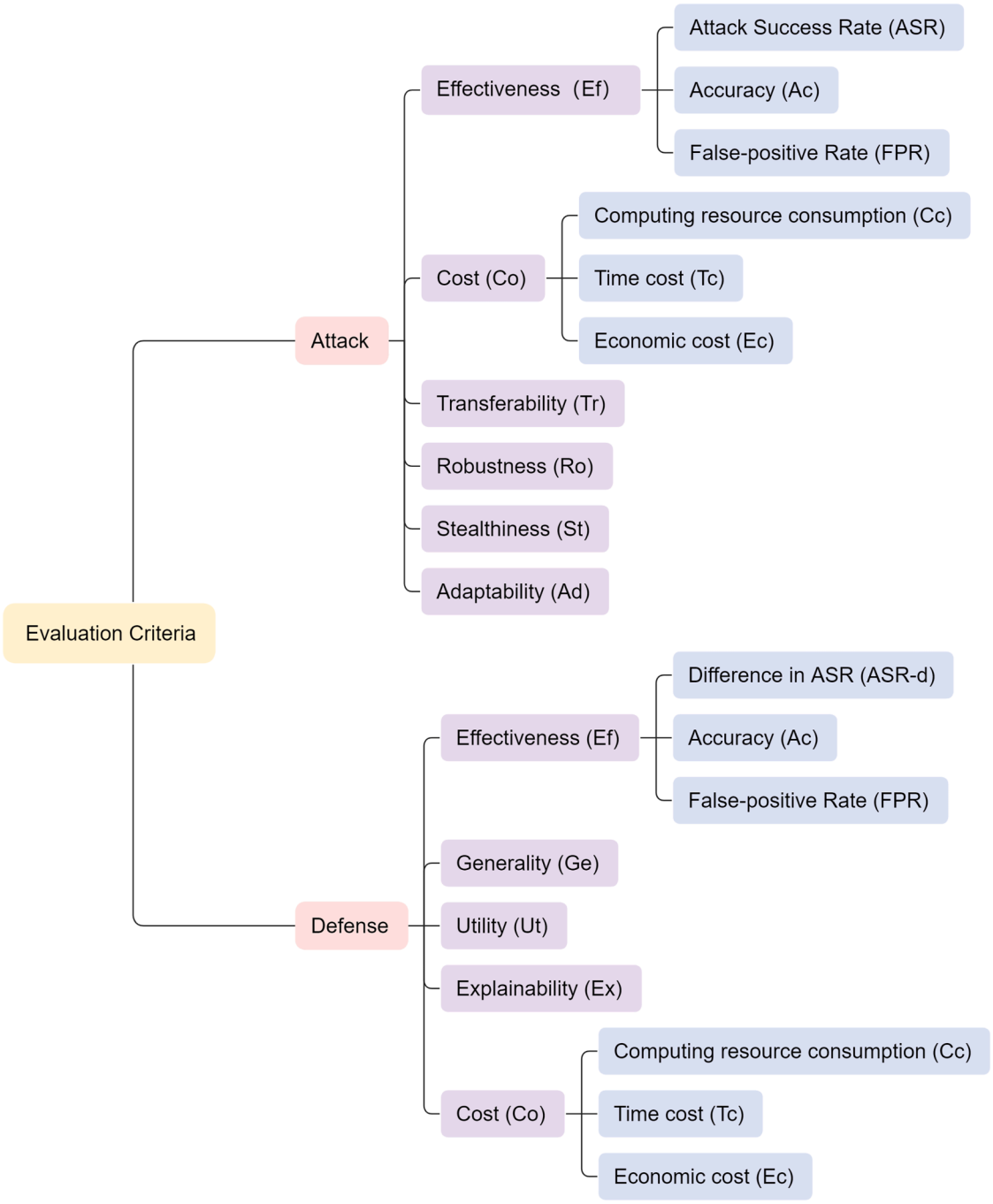

Fig. 3. Evaluation criteria of attacks and defenses.

- Economic cost (Ec): This refers to the direct economic expenditures required to launch and sustain an attack. A representative example is the cost of accessing commercial LLM APIs, which is usually priced based on the number of tokens used. High economic cost can limit the practicality of an attack, especially in resource-limited adversarial environments. We categorize economic cost with reference to the industry-standard per-1,000-token unit, as follows:
  - Low tokens (Lt): Tokens ≤ 1000
  - Medium tokens (Mt): $1000 <$ Tokens $leq 10000$
  - High tokens (Ht): Tokens > 10000

3.1.3. Transferability (Tr)

Transferability refers to the ability of an attack method to retain its effectiveness across different models, tasks, or deployment scenarios. It reflects the generality and adaptability of the attack method, i.e., whether it can be applied to diverse targets without requiring substantial modifications for each specific case. Transferability is typically evaluated along three dimensions: cross-model, cross-task, and cross-scenario, corresponding to the attack's applicability across architectures, tasks, or environmental conditions.

3.1.4. Robustness (Ro)

Robustness measures an attack's ability to remain effective in the presence of defense methods such as adversarial training, input filtering, and model monitoring. The LLM-based agents often integrate multiple alignment strategies [37], aiming to constrain harmful behaviors through prompt engineering or filtering. For example, a model may be guided towards safe responses by automatically detecting and filtering

malicious prompts, or by inserting safety-oriented instructions. Given the prevalence of such defenses in real-world applications, evaluating an attack's robustness under fortified environments is essential. It reflects both the practical viability and resilience of the attack in realistic deployment scenarios.

3.1.5. Stealthiness (St)

Stealthiness refers to the ability of an attack to evade detection by human observers or automated defense methods. It is a critical factor for ensuring the attack's success and persistence. To achieve stealthiness, attackers often employ various techniques to conceal their activities. For instance, backdoor attacks often require triggers that are imperceptible yet functional [38], while prompt injection attacks typically embed adversarial contents in subtle or implicit forms. Stealthiness can be measured by the detectability of adversarial samples or manipulated behavior.

3.1.6. Adaptability (Ad)

Adaptability refers to the ability of an attack to dynamically optimize its strategy during real-time interactions with a single model. Unlike transferability, which focuses on static generalization across settings, adaptivity reflects the attack's ability to adjust its behavior in real-time based on feedback from the target model or environmental changes, thereby sustaining or enhancing its effectiveness. For instance, Reinforcement Learning (RL)-based attacks dynamically refine their adversarial strategies through feedback-driven optimization [39], while other approaches may leverage search algorithms to explore optimal attack paths. Compared to static attacks, adaptive attacks demonstrate greater flexibility, stealthiness, and success rates by continuously evolving in response to active defense methods.

3.2. Evaluation criteria of defenses

In this subsection, we propose a comprehensive set of criteria to evaluate defense performance, including effectiveness, generality, utility, explainability, and cost.

3.2.1. Effectiveness (Ef)

Effectiveness serves as a critical criterion for quantitatively evaluating the performance of defense methods. We adopt three auxiliary metrics for indirect evaluation: ASR-d, Ac, and FPR. The definitions of Ac and FPR have been introduced in Section 3.1.1 and are omitted here. Specifically, ASR-d denotes the reduction in ASR after implementing a defense method, serving as an indirect indicator of its effectiveness. A higher ASR-d indicates a greater defensive effectiveness. In parallel, a higher Ac and a lower FPR suggest the defense's ability to preserve normal functionality of an agent while suppressing adversarial impact.

3.2.2. Generality (Ge)

Generality refers to the extent to which a defense strategy remains effective across diverse LLM-based agent architectures and deployment environments, while also being resilient to a variety of attack methods. A well generalized defense should not be limited to specific frameworks or threat models. Given the continuously evolving nature of adversarial techniques, generality is crucial for ensuring real-world applicability. This property is typically evaluated through experimental evaluations conducted across diverse agent systems and various attack types.

3.2.3. Utility (Ut)

This criterion evaluates the degree to which a defense preserves the model's original reasoning ability and normal user experience while maintaining security. An ideal defense mechanism should maximize protection against attacks while minimizing disruption to normal user interactions. Excessive defense measures can result in degraded quality of model outputs or inadvertently amplify attack effects [40]. We use Ac to denote task performance after applying the defense. Utility tiers are a heuristic classification grounded in deployment-level tolerances for performance judgment: in many production systems, Ac $\geq 90\,\%$ is regarded as high quality, whereas Ac $< 80\,\%$ is typically deemed unacceptable. Accordingly, we classify the utility of defense methods into three tiers based on post-defense Ac values:

- High utility (Hu): Ac $\geq 90\,\%$
- Medium utility (Mu): $80\,\% \leq$ Ac $< 90\,\%$
- Low utility (Lu): Ac $< 80\,\%$

3.2.4. Explainability (Ex)

This refers to the transparency and human interpretability of a defense method in terms of its operational mechanisms, decision logic, and outputs. It is a key factor in establishing the practical utility and technical credibility of a defense method, and is essential to build interpretable AI systems that are trustworthy, value-aligned, and regulatory-compliant [41]. Effective defense methods should incorporate traceable reasoning, formalized mathematical foundations, and interpretable outputs to allow reliable validation and continuous refinement.

3.2.5. Cost (Co)

Cost captures the resource and operational overhead associated with deploying a defense method. It is evaluated along three dimensions:

- Computational Resource Consumption (Cc): This dimension reflects the amount of computing resources, such as CPU, GPU, and memory, required by the defense method during its execution. It is commonly evaluated in terms of the algorithm's time complexity.
  - Low consumption (Lc): Rule-based or static defenses that require no model inference.
  - Medium consumption (Mc): Defenses that involve lightweight or small-scale models for detection or classification.
  - High consumption (Hc): Defenses that rely on large-scale models (e.g., LLMs or agents) typically require multi-round training and multi-step reasoning, often coupled with adaptive decision-making.
- Time Cost (Tc): For defense methods, we focus on the time spent on a specific request, measured from input arrival to a safe output or an explicit block, excluding offline processes such as pretraining or fine-tuning. We categorize time cost as follows:
  - Real-time (Rt): Responds within seconds to under one minute, suitable for real-time input/output monitoring.
  - Minute-level (Ml): Completes within 60 minutes, enabling moderate-latency analysis or lightweight model-based validation.
  - Hour-level (Hl): Takes more than 60 minutes, often involving iterative reasoning, adaptive mechanisms, or model retraining.
- Economic Cost (Ec): This quantifies the monetary expenditure associated with a defense method, with particular emphasis on the number of tokens consumed through interactions with commercial LLM APIs. Since most LLM services are charged based on token usage, a higher token usage implies an increased economic cost. We categorize economic cost with reference to the industry-standard per-1,000-token unit, as follows:
  - Low tokens (Lt): Tokens ≤ 1000
  - Medium tokens (Mt): $1000 <$ Tokens ≤ 10000
  - High tokens (Ht): Tokens > 10000

4. Attacks against the LLM-based agents

In recent years, attacks targeting the LLM-based agents have become increasingly prominent and diverse. In this section, we conduct a thorough review of the latest research advances concerning security threats against the LLM-based agents. We performed a systematic search using keywords such as "the LLM-based agents", "Attack," and "Defense" across major academic databases, including ACM, IEEE Xplore, Web of Science, Springer, Elsevier, Google Scholar, and arXiv, covering publications from 2023 to 2025, since 2023 witnessed a rapid surge of the

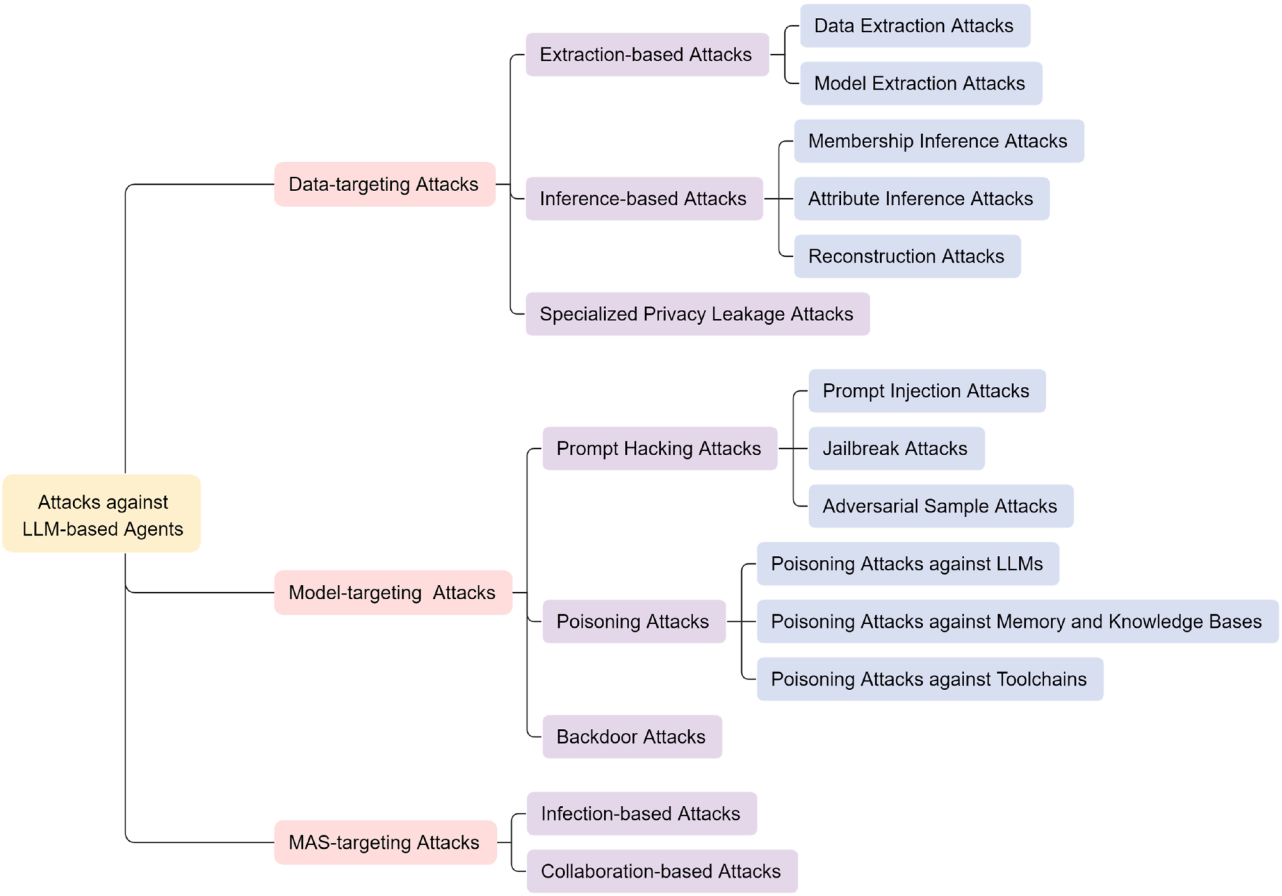

Fig. 4. Taxonomy of attacks against the LLM-based agents.

LLM-based agents. Based on relevance and scholarly quality, we selected 22 representative studies for in-depth analysis. In Section 4.1, we categorize these works into three major groups based on the impact domains of the attacks. Then, in Section 4.2, we evaluate the strengths and limitations of each work using the attack evaluation criteria introduced in Section 3.1. Finally, in Section 4.3, we conclude with a comparative summary and discussion of the reviewed works.

4.1. Taxonomy of attacks

To systematically analyze the security vulnerabilities of the LLM-based agents operating in complex environments, we propose a taxonomy that classifies attacks according to their underlying attack intents. In contrast to previous schemes that primarily categorize attacks by their targeted components, our framework highlights the functional and systemic impacts of adversarial behaviors on the agent performance. Based on this principle, attacks are grouped into three major categories: (i) data-targeting attacks, which aim to extract private or confidential information from training datasets or model parameters; (ii) model-targeting attacks, which seek to manipulate the model's inference, causing it to produce malicious outputs, incorrect responses, or unsafe tool executions; and (iii) MAS-targeting attacks, which disrupt the overall operation or cooperative mechanisms within multi-agent environments. Each primary category is further refined into secondary subtypes according to the specific technical approaches employed. An overview of this taxonomy is illustrated in Fig. 4.

4.1.1. Data-targeting attacks

These attacks aim to exfiltrate sensitive information that the LLM-based agents access, store, or generate throughout their lifecycle, including user inputs, private training data, retrieved documents, long-term memory contents, and system prompts. Such attacks not only infringe upon user privacy but also violate data compliance and expose intellectual property, especially in the agents equipped with long-term memory or external data interfaces. Based on how adversaries acquire such data, we categorize these attacks into three technical strategies: extraction-based, inference-based, and specialized privacy leakage mechanisms.

Extraction-based attacks. These attacks involve adversarial behaviors that directly target the retrieval of sensitive information from the LLM-based agents or their underlying data sources, such as training data, model gradients, or even embedded prompts [12]. These attacks pose direct threats to user privacy and severely undermine data compliance and system trustworthiness. Currently, extraction-based attacks are primarily categorized into two types: data extraction attacks [42,43] and model extraction attacks [44,45].

- *Data Extraction Attacks.* These attacks focus on directly retrieving sensitive data from the agents or their components through crafted queries or controlled interactions. They can be broadly categorized into four types. (i) Training data extraction: LLMs have been shown to memorize and regurgitate rare or sensitive training samples. This vulnerability was first demonstrated by Carlini *et al.* [42] in models like GPT-2. (ii) Reasoning trace leakage: While intermediate reasoning steps in LLMs are often assumed to be private, Green *et al.* [46] demonstrate that sensitive user information can reside in these reasoning traces and be unintentionally exposed during multi-step inference or extracted via prompt injection. (iii) Memory extraction: In agents with persistent memory, long-term interaction histories and contextual constitute key attack surfaces that may expose sensitive information to adversaries [47,48]. (iv) Knowledge stealing: In RAG-based systems, attackers may exploit retrieval prompts or inject

crafted inputs to extract proprietary contents embedded in external corpora [49,50].
- *Model Extraction Attacks.* These attacks aim to recover the functionality, hyperparameters, or architecture of a target model through black-box interactions [51]. Depending on the attacker's objective, these attacks are categorized into three types: functional-level extraction, parameter-level extraction, and model-level extraction, typically exploiting public APIs that only expose the model's input-output behavior. A particularly critical threat within parameter-level extraction is prompt leakage [52], which refers to the unauthorized extraction of internal system prompts, key control instructions that govern the behavior of the LLM-based agents [15].

Inference-based Attacks. These attacks refer to the adversarial process that indirectly deduces sensitive information by issuing carefully crafted queries and analyzing the resulting output patterns or statistical responses [53]. Rather than directly accessing raw data, these attacks aim to recover approximate or representative examples of sensitive information [54]. In general, the inference-based attacks can be categorized into three types: membership inference, attribute inference, and reconstruction attacks.

- *Membership Inference Attacks.* These attacks seek to determine whether a specific data sample is part of an LLM's training data [55]. This is typically achieved by analyzing the model's output behavior or by training a dedicated inference model to distinguish members from non-members.
- *Attribute Inference Attacks.* Their purpose is to reveal specific features or properties using training data. While most generative models are vulnerable to such attacks [56], Staab et al. [57] further demonstrates that pre-trained LLMs are capable of inferring personal information at scale.
- *Reconstruction Attacks.* These attacks are an intrusive form of data leakage that aim to recover training data, system prompts, or other confidential contents by analyzing model outputs or gradient information. They comprise three main subtypes: (i) Model inversion, which leverages the model's memorization behavior to reconstruct sensitive inputs or text sequences [58]. (ii) Prompt stealing, which aims to covertly infer and reconstruct internal system prompts by exploiting subtle signals embedded in the model's responses [15]. (iii) Gradient leakage, which extracts private data from exposed gradients, posing risks in federated and distributed training [55].

Specialized privacy leakage attacks. These attacks exploit the agent's interactive pathways, such as toolchains, memory, external APIs, or perceptual inputs, to indirectly induce sensitive information leakage indirectly, without directly probing the LLM itself. Unlike conventional methods that target model outputs or infer user attributes through statistical analysis, they manipulate downstream components or system behaviors to trigger unintended disclosures. Their reliance on system-level functionalities and indirect activation pathways makes them particularly difficult to detect using standard LLM-oriented defenses. Representative studies include AirGapAgent [59], Imprompter [60], mimicking the familiar [61], and misusing tools with visual adversarial samples [62].

4.1.2. Model-targeting attacks

Model-targeting attacks aim to manipulate the inference or decision-making process of a model, thereby coercing it into producing malicious outputs or executing harmful actions. Unlike data-targeting attacks, which compromise information confidentiality, these attacks primarily threaten the model's functional integrity and induce behavioral deviations from intended objectives. Based on their attack techniques, model-targeting attacks can be broadly classified into three subtypes: prompt hacking attacks, poisoning attacks, and backdoor attacks. While these attacks may exhibit overlapping consequences, they differ markedly in terms of manipulation strategies, persistence characteristics, and points of injection. The concrete mechanisms and representative examples of each subtype are detailed in the following subsections.

Prompt hacking attacks. These attacks refer to inference-time manipulations that exploit the LLM-based agent's reliance on dynamic prompts, intermediate inputs that shape the model's reasoning and output generation. These prompts are not limited to direct user inputs but also include contents automatically gathered or constructed during task execution, such as perception-derived data, retrieved information, tool responses, environmental feedback, and short-term memory context assembled into the current prompt. By altering these inputs at runtime, attackers can covertly steer agent behavior without modifying model parameters. Common strategies include prompt injection, jailbreak, and adversarial samples.

- Prompt Injection Attacks. These attacks inject adversarial instructions into prompts to induce undesired behavior. In Direct Prompt Injection (DPI), malicious content is embedded directly into user inputs. In Indirect Prompt Injection (IPI), adversarial prompts are hidden in external contents, such as web pages, API responses, or RAG, which are later processed by the agent, making detection difficult and manipulation covert. Additionally, since demonstrations and context also constitute part of the prompt, they are equally susceptible to tampering. Adversaries can exploit the model's in-context learning by injecting adversarial samples into these components, biasing model behavior without modifying its parameters or architecture [63].
- Jailbreak Attacks. These attacks aim to bypass alignment safeguards through carefully crafted prompts, potentially inducing outputs that violate ethical or legal boundaries. Specifically, based on how adversarial prompts are constructed, jailbreak attacks can be classified into two categories: manual jailbreak and automated jailbreak [64].
- Adversarial Sample Attacks. These attacks refer to techniques where an attacker introduces subtle, adversarially crafted perturbations into benign input samples, causing a model to produce incorrect predictions [65]. Typically imperceptible to human observers, such perturbations exploit the model's sensitivity during inference, degrading its performance without any retraining or parameter access.

Poisoning attacks. These attacks are long-term structural manipulations targeting core components of the LLM-based agents across their lifecycle. Unlike prompt hijacking, which alters runtime inputs to affect immediate behavior, poisoning modifies training data, model parameters, memory, or toolchains to implant persistent adversarial samples. Such attacks compromise the agent's performance, often remaining stealthy and resistant to prompt-level defenses. In the context of the LLM-based agents, poisoning attacks can be categorized based on the targeted components: poisoning LLMs, poisoning memory and knowledge bases, and poisoning toolchains.

- Poisoning Attacks against LLMs. Poisoning attacks against the core model LLMs can be broadly classified into two types: data poisoning and model poisoning. Data poisoning involves injecting adversarial samples during pre-training, fine-tuning [66], or Reinforcement Learning From Human Feedback (RLHF) stages [67], causing the model to internalize malicious behavior patterns or suffer from performance degradation. In contrast, model poisoning directly modifies model parameters or architectural components to implant malicious logic or impair capabilities, as demonstrated in [68].
- Poisoning Attacks against Memory and Knowledge Bases. These attacks refer to the injection of malicious data into the agent's long-term memory [69], compromising the semantic integrity of stored information. When retrieved during inference, these poisoned entries can mislead the agent and induce incorrect outputs. Notably, such attacks can be executed without accessing the model or modifying its parameters.

- Poisoning Attacks against Toolchains. These attacks target vulnerabilities across the hardware, software, and external tool components of the LLM-based agents. In this paper, we focus on the external tool surface, specifically, adversaries can mislead the agent by injecting factual inaccuracies into tool outputs, while also compromising APIs or third-party libraries by embedding malicious code that induces harmful task execution [13,70].

Backdoor attacks. These attacks are a specialized form of poisoning attacks in which models behave normally under benign inputs but produce adversary-specified outputs when exposed to specific triggers [71]. While traditional backdoors are typically implanted during training via data or model poisoning to associate triggers with malicious outputs [72,73], this work focuses on novel inference-time mechanisms emerging in deployed the LLM-based agents. These attacks fall into two main categories: prompt manipulation and interaction-level manipulation. Prompt-based backdoors exploit the model's sensitivity to input prompts and include: (i) injecting hidden instructions into user queries; (ii) embedding trigger-laden demonstrations in few-shot or in-context learning [74,75]; (iii) inserting triggers into chain-of-thought reasoning [68]; and (iv) crafting adversarial prompts with embedded backdoors [76]. Meanwhile, interaction-level attacks target external components such as knowledge bases, memory modules, toolchains, or environments. By injecting trigger-bearing contents into these channels, adversaries can covertly induce malicious behaviors. Recent studies demonstrate the feasibility and stealthiness of such attacks in real-world LLM agent deployments [69,77].

4.1.3. MAS-targeting attacks

In MAS, security threats often emerge through the propagation of harmful contents, hallucinations, or biases via inter-agent interactions [17]. These threats may arise unintentionally or be orchestrated by malicious agents and are amplified by the system's collaborative and communicative nature. Compared to single-agent settings, attacks in MAS are more scalable, stealthy, and difficult to mitigate. We categorize such threats into two primary types based on their manipulation mechanisms: infection-based attacks and coordination-based attacks.

Infection-based attacks. These attacks operate by compromising a small subset of the agents and leveraging their interactions to propagate malicious influence, such as adversarial prompts, corrupted knowledge, or abnormal behaviors, throughout the MAS. The initial infection may originate from a single prompt injection, adversarial input, or poisoned model update, but through iterative communication and shared task execution, the manipulation gradually diffuses across the agents, leading to systemic degradation.

4.1.3.1. Collaboration-based attacks. These attacks target the interaction mechanisms that enable coherent collaboration within MAS. Rather than triggering cascading contamination, adversaries manipulate shared protocols, planning strategies, or communication pathways to disrupt collective decision-making. This may involve tampering with transmitted information, corrupting role assignments, or injecting misleading contents into collaborative tasks such as debate or negotiation.

4.2. Review on attacks

4.2.1. Data-targeting attacks

Extraction-based attacks. Unlike traditional training data extraction attacks that rely on precisely structured prefixes, Panda et al. [78] proposed Neural Phishing, an effective attack that achieves high success rates with minimal data injection and only vague priors on prefix structure. By injecting innocuous-looking poisoned examples into the training corpus, the model is subtly guided to memorize secrets during fine-tuning, which can later be revealed during inference. Without relying on duplication or precise knowledge of the target, the method successfully recovers complex high-entropy secrets. However, the method's effectiveness relies on the poisoned inputs being introduced prior to the sensitive data during training. Tr is satisfied, as the method was tested across multiple model architectures and datasets. Randomized prompts help evade duplication-based defenses, suggesting St is satisfied. Ad is fulfilled by employing feedback-guided prefix optimization. The attack achieves an ASR of up to 80 %. Tc is non-real-time under full training assumptions; Cc is high due to extensive training. Ro, Ac, FPR, and Ec were not reported.

To address the limitations of manually crafted queries in existing prompt leakage attacks, Hui et al. [52] proposed PLeak, an automated framework designed to reconstruct system prompts in black-box LLM applications. PLeak consists of two stages: an offline phase that performs adversarial query optimization using shadow models, and an online phase that aggregates model outputs to reconstruct hidden prompts. PLeak demonstrates strong performance without requiring access to model internals; however, its effectiveness declines under prompt-hardening defenses. In a complementary direction, Zhang et al. [79] presented RepeatLeakage, the first framework to exploit the repetition behavior of LLMs for prompt leakage. The attack constructs trigger inputs to induce the model to repeat parts of hidden prompts, followed by repeated querying and output filtering for prompt reconstruction. While RepeatLeakage is highly effective in black-box settings, its performance is sensitive to the structural complexity of the underlying prompts. Both methods satisfy Tr, as they were evaluated across multiple LLMs and tasks. Tc is non-real-time due to its iterative nature. PLeak meets Cc through adversarial query optimization, while RepeatLeakage satisfies it via multi-round querying and output aggregation. PLeak additionally satisfies Ad via feedback-driven refinement and partially meets Ro by bypassing filter-based defenses. RepeatLeakage achieves an Ac of 77.8 % and partially satisfies St through randomized trigger insertion. ASR, FPR, and Ec were unreported in both works, and St was not addressed in PLeak.

To address the fundamental inconsistency between conventional model extraction attacks and the alignment paradigm of LLMs, Liang et al. [80] proposed LoRD, a novel model extraction algorithm specifically tailored for LLMs. LoRD introduces a method based on RL that evaluates local probability differences between responses of local and victim models to construct positive and negative sample pairs, which are then used to iteratively optimize a composite loss comprising a policy gradient objective and a regularization term. Compared with conventional extraction methods, LoRD achieves higher query efficiency and exhibits stronger resilience against watermarking, although its extension to multi-modal architectures remains an open challenge. In evaluation, LoRD satisfies Tr via multi-model, multi-task validation. It meets Ad due to its RL-based attack paradigm. It partially satisfies Ro for resisting black-box watermarking. Ac reaches 78.5 %. Tc is non-real-time due to iterative training. Cc is medium since it avoids loading the initial weights of models. While St, ASR, Ec, and FPR were not reported.

Inference-based attacks. To mitigate the high FPR of Membership Inference Attacks (MIAs) in real-world scenarios, Fu et al. [81] proposed SPV-MIA, a self-prompted membership inference attack. It reveals model privacy vulnerabilities by prompting the target LLM to generate a reference dataset with a distribution similar to its training data. SPV-MIA comprises two modules: a self-prompted reference model, which generates a reference dataset via prompting the target LLM, and a probabilistic variation evaluator, which detects membership by estimating the probability shift of target records using symmetric textual perturbations. SPV-MIA significantly improves the AUC of MIAs from 0.7 to 0.9, but its applicability is currently limited to Causal Language Models (CLMs). However, it partially satisfies Tr through testing on multiple datasets. It satisfies Ro through testing under the DP-SGD defense. Its reliance on LLM inference and fine-tuning results in high Cc and non-real-time Tc. Ad, St, Ec, ASR, Ac, and FPR were not reported.

Shen et al. [82] proposed PromptStealer, a two-stage prompt reconstruction framework that exposes a novel vulnerability in text-to-image generation models. It comprises a subject generator based on fine-tuned image captioning models and a modifier detector, implemented as a multi-label classifier trained on the Lexica-Dataset to extract stylistic attributes. The final prompt is formed by concatenating outputs from both modules. While PromptStealer outperforms baselines on open-source models, its effectiveness declines on proprietary systems, indicating that Tr is partially satisfied. Due to multi-stage training over a large modifier set, Cc is high, and Tc is non-real-time. Its resilience under different defense settings indicates that Ro is satisfied. ASR, Ac, and FPR are not applicable. Ec, St, and Ad were not reported.

Specialized privacy leakage attacks. Recent work has exposed the vulnerability of the LLM-based agents to tool misuse attacks, which can lead to unauthorized behaviors or information leakage. Fu et al. [60] proposed Imprompter, a gradient-based framework that automatically generates adversarial prompts by optimizing input using Greedy Coordinate Gradient (GCG) [83] for text and Adaptive Moment Estimation (Adam) [84] for images. These prompts are crafted to appear semantically benign while triggering unintended tool use. Imprompter constitutes the first systematic approach for the automated generation of stealthy adversarial prompts. However, its effectiveness is contingent upon the cross-model transferability of the crafted adversarial samples. To overcome the limitations of static and easily detectable attacks, Jiang et al. [61] proposed AUTOCMD. This RL-based framework dynamically generates stealthy and context-aware commands. It builds an attack case database from prior tool interactions and leverages a T5-based generator optimized via Proximal Policy Optimization (PPO). AUTOCMD demonstrates superior adaptability and stealthiness in dynamic contexts, outperforming static baselines, but its effectiveness declines when interacting with unfamiliar tools lacking prior interaction data. Both approaches satisfy Tr, as they were evaluated across multiple LLMs and tasks. Ad is supported by the use of optimization algorithms and RL, respectively. St is met as both methods generate prompts that are difficult to detect through manual inspection or heuristic defenses. Tc is non-real-time in both cases, as both approaches construct prompts or commands offline. Both rely on intensive operations such as model inference and optimization, resulting in a high Cc. AUTOCMD satisfies Ro by evaluating under multiple defenses, while Imprompter does not. Reported ASR values are 80 % (Imprompter) and 94.5 % (AUTOCMD), respectively. But Ac, Ec, and FPR were not reported in either study.

4.2.2. Model-targeting attacks

Prompt hacking attacks. To address the generality and position-sensitivity limitations of manual injection methods in LLM-as-a-Judge systems, Shi et al. [85] proposed JudgeDeceiver, an optimized prompt injection framework that automatically generates refined prompt sequences to manipulate evaluation outcomes. The framework formulates the attack as a triple-objective optimization problem, combining consistency loss, location-aware enhancement, and perplexity suppression. Injection sequences are trained on synthetic responses via gradient-based optimization, enabling the selection of attacker-preferred outputs across various positions and models. While JudgeDeceiver outperforms manual and jailbreak methods, it requires constructing shadow datasets and repeated optimization, incurring notable computational cost. Tr is met through multi-model, multi-task evaluation. Ad is satisfied due to optimization strategies. St is satisfied by maintaining low perplexity, and Ro by retaining effectiveness under multiple defenses. The best ASR reaches 100 %, and Ac is up to 99 %. Involves model optimization and inference, resulting in high Cc. The need for dataset construction and iterative tuning makes Tc non-real-time. Ec and FPR were not reported.

Nakash et al. [86] proposed an indirect prompt injection framework, Foot-in-the-Door (FITD), revealing a vulnerability in ReAct-based agents whereby seemingly benign requests can be leveraged to trigger malicious actions. FITD incorporates an additional disruption phase comprising two components, a distractor request and a malicious instruction, combined to enable covert execution. The agent first addresses the innocuous request and then executes the embedded malicious instruction, thereby fulfilling the attacker's objective. This approach increases ASR by up to 44.8 % compared with baselines but relies on manipulating the ReAct reasoning process, limiting its applicability to other agent architectures. FITD partially satisfies Tr due to evaluations across diverse tasks, but remains constrained by its dependence on ReAct. It satisfies Ro by bypassing reflection-based defenses and meets Ad through strategy-level adjustments. The attack achieves an ASR exceeding 95 %. Constructed solely via prompt engineering, it incurs a low Cc, and, as it evolves dynamically with the context, Tc is real-time. Ac, Ec, and FPR were not reported.

To address the limited effectiveness of mainstream jailbreak methods against LLM-based chatbots, Deng et al. [87] proposed MASTERKEY, a framework for automatically generating jailbreak prompts that can bypass chatbot defenses. MASTERKEY first employs a time-based Structured Query Language (SQL) injection technique to reveal defense mechanisms such as keyword matching and semantic analysis. Building on these insights, it trains an LLM to generate effective jailbreak prompts through four steps: dataset construction, continuous pre-training, task tuning, and reward-ranked fine-tuning. This approach achieves an average ASR of 21.58 %, significantly outperforming the 7.33 % ASR of existing prompts. However, its reliance on response-time analysis makes it susceptible to network latency effects. The method satisfies Tr by evaluating across multiple LLM chatbots. It achieves St by applying encoding strategies to evade keyword-based defenses. It bypasses multiple defenses via strategic manipulation of time-sensitive responses, but only partially meets Ro due to a lack of experimental validation. Ad is fulfilled via adaptive prompt generation. Its reliance on LLM fine-tuning incurs high Cc, while multi-turn interaction makes Tc non-real-time. Ac, Ec, and FPR were not reported.

To overcome the limitations of traditional jailbreak attacks, namely their reliance on single-turn inputs and inability to handle multimodal systems, Russinovich et al. [64] proposed Crescendo. This fully black-box method employs multi-turn prompting and progressive guidance, and is complemented by its automation tool Crescendomation. Crescendo begins with a benign question and leverages the model's attention to its own outputs to gradually elicit prohibited contents. Crescendomation automates this process with another LLM (e.g., GPT-4), refining prompts through feedback loops until success. On Gemini-Pro, Crescendomation improves ASR by 49–71 %, achieving an ASR of 82.6 %, but requires API access and support from an attack-generation model. Crescendo meets Tr through evaluation on diverse LLMs and tasks, Ro via testing under multiple defenses, St by using undetectable benign inputs, and Ad through dynamic prompt adaptation. Its multi-turn nature makes Tc non-real-time, and reliance on repeated LLM inference yields high Cc and Ec. Ac and FPR were not reported.

Poisoning attacks. To expose the fundamental security risks of instruction-tuned LLMs that rely on user-contributed training data, Wan et al. [88] proposed a data poisoning method targeting the instruction-tuning phase. The approach constructs trigger-containing poison samples, ranks them using an n-grams scoring function to identify highly effective candidates, and inserts them into the training set for fine-tuning. Even with a small number of poisoned samples, the attack is effective across diverse tasks and model scales. However, existing defenses are of limited efficacy and often incur substantial performance degradation. The method partially satisfies Tr, being evaluated on only one model class, and partially satisfies St, as only clean-label samples appear benign and are not easily detected. It meets Ro by maintaining effectiveness under defenses and meets Ad through optimization strategies. Its reliance on LLM fine-tuning entails high Cc and non-real-time Tc. Its

ASR reaches 92.8 % for dirty-label attacks and 55.6 % for clean-label attacks. Ec, Ac, and FPR were not reported.

To explore the new attack surface in RAG systems, Zou et al. [89] proposed PoisonedRAG, the first knowledge corruption attack framework targeting the knowledge database. It injects a small number of malicious texts into a retrieval database, inducing the LLM to produce attacker-specified answers. The attack is formulated as satisfying a retrieval condition (S) and a generation condition (I), with tailored construction and optimization strategies for white-box and black-box settings, after which the final malicious text is created by concatenating S and I. The method achieves high ASR in both settings with minimal injected contents but assumes write access to the knowledge database, which may limit real-world applicability. It satisfies Tr via multi-dataset and multi-LLM evaluation, Ro through defense testing, and St as texts evade detection. With about two LLM queries, Cc and Ec are medium, and Ad is achieved through adaptable strategies. In a white-box mode, ASR reaches 99 %. Generation takes under 30 seconds, making Tc real-time. Ac and FPR were not reported.

Backdoor attacks. Wang et al. [73] presented BadAgent, a pioneering backdoor attack framework targeting the LLM-based agents. The attack is implemented by injecting poisoned data during the fine-tuning phase, which, when activated, causes the agent to exhibit malicious behaviors during task execution. BadAgent encompasses two generic strategies: active attacks, which rely on explicitly inserted triggers within user inputs, and passive attacks, which activate the backdoor through triggers covertly embedded in an external environment. BadAgent demonstrates strong effectiveness, achieving over 90 % ASR across three real-world task scenarios. Nonetheless, its generality to models with more than 13B parameters and a broader range of tasks remains uncertain, so it partially satisfies Tr. Ro is met with the attack, remaining effective even under data-centric defensive fine-tuning. The model retains normal behavior under benign inputs, thereby holding sound St. Involving LLM fine-tuning requiring 6–8 hours results in high Cc, non-real-time Tc, and high Ec. Ad and FPR were not reported. Ac was not mentioned.

Zhao et al. [75] proposed ICLAttack, the first context-based backdoor attack method that operates without any model fine-tuning. It leverages a small set of well-crafted in-context demonstrations to induce LLMs to produce attacker-specified outputs during inference. ICLAttack introduces two complementary mechanisms: one constructs a poisoned demonstration by embedding a trigger word into demonstration samples; the other replaces an original prompt with a covertly poisoned demonstration prompt template to influence model behavior. ICLAttack demonstrates strong performance across multiple LLMs and tasks, achieving an average ASR of over 95 %. However, its effectiveness relies heavily on demonstration structure and shows limited generalization in complex or diverse scenarios, thus only partially satisfying Tr. Ro is satisfied due to its evasion of mainstream defenses, and St holds, as the trigger remains subtle and hard to detect. The attack uses trigger-based demonstration contexts, resulting in non-real-time Tc. With no training and few samples, the Cc remains low. Ad, Ec, and FPR were not reported. Ac is not available.

To investigate vulnerabilities in Chain-of-Thought (CoT) demonstration-based reasoning, Xiang et al. [68] introduced BadChain, the first backdoor attack framework for CoT prompting that requires no access to training data or model parameters. BadChain injects specific backdoor reasoning steps into a subset of CoT demonstrations so that, during inference, the presence of a trigger word prompts the model to produce attacker-specified outputs. BadChain achieves an exceptionally high ASR in complex reasoning tasks, with an average ASR of 97.0 % on GPT-4, while having a negligible impact on the original performance of the model. However, its effectiveness depends on the careful construction of demonstrations and the proportion of injected samples. The method satisfies Tr via multi-LLM, multi-task evaluation, and Ro through sustained performance under defenses. It meets St as a covert trigger embedding. Requiring prior backdoor implantation makes Tc non-real-time, while Cc is low, relying only on demonstration modification. Ac is not available. Ad, Ec, and FPR were not reported.

Yang et al. [77] conducted the first systematic investigation of backdoor threats targeting the LLM-based agents and proposed a general backdoor attack framework. Based on the attack objectives, the framework formalizes three distinct categories: Query-Attack, where a trigger is embedded directly into a user query; Observation-Attack, where a trigger appears in intermediate observations returned by the environment; and Thought-Attack, where an attacker injects malicious reasoning steps without altering its final output. This work is the first to comprehensively reveal the diverse manifestations of backdoor attacks in the agents, demonstrating high ASR across different scenarios. However, its evaluation is limited to the ReAct framework, lacking broader validation, and thus only partially satisfies Tr. Certain attacks, by embedding triggers in intermediate reasoning or observation stages, partially satisfy St. The average ASR reaches 78 %. As the approach requires LLM-based inference to generate poisoned samples and full-parameter fine-tuning, Cc is high, and Tc is non-real-time. Ro, Ad, Ec, and FPR were not reported. Ac is not available.

4.2.3. MAS-targeting attacks

Infection-based attacks. Gu et al. [90] proposed an infection-based jailbreak mechanism, a novel propagation-driven attack targeting multimodal the LLM-based agents. The attack operated in two phases. In the initial stage, a single agent is infected by embedding a covert jailbreak prompt into an adversarial image, which is then stored in the agent's memory. This compromised memory component allows for internal triggers for unauthorized behavior. In the subsequent phase, the adversarial image spread across the agents through automatic inter-agent communication and image-sharing protocols, allowing for exponential propagation without ongoing attacker involvement. This work is the first to simulate large-scale, self-replicating attack trajectories in multi-modal LLM environments. Although the approach demonstrates strong scalability, its limited evaluation across diverse LLM architectures means it only partially satisfies Tr. The adversarial image is stealthy and propagates covertly, satisfying St. Its generation involves gradient-based optimization, fulfilling Ad. Due to extensive LLM inference and agent interactions, with experiments running for nearly a month, Cc and Ec are high, and Tc is non-real-time. ASR reaches 90 %. Ro, Ac, and FPR were not reported.

Tan et al. [91] conducted the first investigation into a novel vulnerability in Multimodal Large Language Model (MLLM) agent societies, i.e., indirect propagation of malicious contents, and proposed a new attack framework to exploit it. Specifically, the attacker injects adversarial noise into a wolf agent's image input, causing it to produce misleading prompts that are forwarded to sheep agents. The noise is iteratively optimized via cross-entropy loss to align outputs with target malicious contents, and the final adversarial image is used to assess cross-agent infection effectiveness. This work is the first to demonstrate the feasibility of indirect, cross-agent adversarial prompting in MLLM societies, achieving near 100 % ASR in vision-based settings. However, the attack shows reduced effectiveness in certain scenarios. The method was evaluated across multiple MLLM agent architectures and malicious objectives, satisfying Tr. The injection of imperceptible noise and the use of indirect manipulation fulfill St. The use of iterative optimization techniques satisfies Ad. Given the reliance on intensive MLLM inference and optimization, Cc is high, and Tc is non-real-time. Ro, Ec, Ac, and FPR were not reported.

To address the constrained propagation efficiency of traditional LLM attacks in MAS, Yu *et al.* [92] proposed GIGA, a generalizable gradient-based attack framework designed for multi-turn collaborative settings. Targeting a Message-Only Intervention (MOI) scenario, GIGA triggers widespread compromise through a single infected agent that generates an adversarial content, which is then propagated via iterative inter-agent dialogue. By avoiding repeated interventions, the

Table 2
Comparison of attack methods across evaluation criteria.

Ref.	Attack Impact Targets	Manipulation Mechanisms	Attack Type	Tr	Ro	St	Ad	ASR ↑	Ac ↓	FPR ↑	Cc	Tc	Ec
[78]	DA	Extraction	DEA	•	×	•	•	80 %	×	×	H	N	×
[52]		Extraction	MEA	•	○	×	•	×	×	×	M	N	×
[79]		Extraction	MEA	•	×	○	×	×	77.8 %	×	M	N	×
[80]		Extraction	MEA	•	○	×	•	×	78.5 %	×	M	N	×
[81]		Inference	MIA	○	•	×	×	×	×	×	H	N	×
[82]		Inference	MIA	•	•	×	×	×	×	×	H	N	×
[60]		Specialized	SPLA	•	×	•	•	80 %	89.1 %	×	H	N	×
[61]		Specialized	SPLA	•	•	•	•	94.5 %	×	×	H	N	×
[85]	MA	Prompt Hacking	DPIA	•	•	•	•	100 %	99 %	×	H	N	×
[86]		Prompt Hacking	IPIA	○	•	×	•	95 %	×	×	L	R	×
[87]		Prompt Hacking	JA	•	○	•	•	21.58 %	×	×	H	N	×
[64]		Prompt Hacking	JA	•	•	•	•	82.6 %	×	×	H	N	Ht
[88]		Poisoning	PL	○	•	○	×	55.6 %	×	×	H	N	×
[89]		Poisoning	PM	•	•	•	•	97 %	×	×	M	R	Mt
[73]		Poisoning	BA (Training)	○	•	•	×	90 %	\	×	H	N	Ht
[75]		Poisoning	BA (Inference)	○	•	•	×	95 %	\	×	L	N	×
[68]		Poisoning	BA (Inference)	•	•	•	×	97 %	\	×	L	N	×
[77]		Poisoning	BA (Inference)	○	×	○	×	78 %	\	×	H	N	×
[90]	MAS	Infection	\	○	×	•	•	90 %	×	×	H	N	Ht
[91]		Infection	\	•	×	•	•	100 %	×	×	H	N	×
[92]		Infection	\	•	○	×	•	90 %	94 %	×	H	N	×
[93]		Collaboration	\	•	○	×	•	×	50 %	×	H	N	×

** DA: Data-targeting Attacks; MA: Model-targeting Attacks; MAS: Multi-Agent System-targeting Attacks. DEA: Data Extraction Attacks; MEA: Model Extraction Attacks; MIA: Membership Inference Attacks; SPLA: Specialized Privacy Leakage Attacks; DPIA: Direct Prompt Injection Attacks; IPIA: Indirect Prompt Injection Attacks; JA: Jailbreak Attacks. PL: Poisoning Attacks against LLMs; PM: Poisoning Attacks against Memory and Knowledge Bases; BA: Backdoor Attacks; IA: Infection-based Attacks; CA: Collaboration-based Attacks. * Tr: Transferability; Ro: Robustness; St: Stealthiness; Ad: Adaptability. * ASR: Attack Success Rate; Ac: Accuracy; FPR: False Positive Rate; ↑: Higher is better; ↓: Lower is better. * Cc: Computational resource consumption; Tc: Time cost; Ec: Economic Cost. * L: Low; M: Medium; H: High; R: Real-time; N: Non-real-time; Lt: Low tokens; Mt: Medium tokens; Ht: High tokens. * •: Satisfied; ○: Partially Satisfied; ×: Not Discussed; \: Not Available. * The quantitative values provided in this table are sourced from the corresponding original papers.*

framework reduces computational overhead while maintaining high scalability. However, its attack success is highly sensitive to initialization, leading to inconsistent performance across different runs. In terms of performance, Tr is satisfied as the attack was tested on diverse models and tasks, achieving an ASR of 90 % and Ac of 94 %. Ro is only partially satisfied, as evaluation is limited to rewriting-based defenses. The attack satisfies Ad due to multi-round optimization and agent interaction, leading to high Cc. Since propagation takes over an hour, Tc is non-real-time. St, Ec, and FPR were not reported.

Collaboration-based attacks. To investigate security risks in multi-agent collaborative reasoning, Amayuelas et al. [93] systematically demonstrated and evaluated how persuasive capabilities in multi-agent debates can compromise collective robustness. This approach introduces an adversarial agent into a multi-agent debate setting, which persistently advocates for an incorrect answer and attempts to persuade other agents through iterative dialogue, ultimately steering the collaborative LLM system towards an erroneous consensus. However, the generation of debate interactions incurs significant time and computational overhead, limiting the scalability of experimental evaluation. The method satisfies Tr, as it was evaluated across different models and scenarios. Ro is partially met, since prompt-based defenses show limited effectiveness on certain models. Ad is satisfied as the strategy optimizes for persuasive arguments. Given that the attack unfolds over multiple rounds of agent debate, Tc is non-real-time, and Cc is high. Ac drops below 50 % on most evaluated models. St, Ec, ASR, and FPR were not reported.

4.3. Comparison and discussion

With the widespread deployment of the LLM-based agents in open-world environments, the security threats they face have grown increasingly diverse and complex. To systematically analyze the distribution and technical characteristics of existing attack methods, we summarize and compare 22 representative attack methods in Table 2, with all quantitative data sourced from the original studies. Based on this, we conduct an in-depth discussion along three major dimensions: attack target distribution, manipulation mechanisms, and core evaluation metrics.

With respect to both the distribution of attack targets and the underlying manipulation mechanisms, current research shows a marked concentration on single-agent systems, accounting for 9/11 of all identified attack scenarios. Among these, data-targeted attacks make up 4/11, focusing primarily on sensitive information extraction, prompt reconstruction, and model replication. Model-targeting attacks constitute 5/11, primarily relying on prompt injection and jailbreak techniques to manipulate the model's reasoning process, while also involving poisoning or backdoor implantation into agent components in either the training stage or the inference stage. In contrast, attacks on MAS account for only 2/11, indicating an early stage of exploration. However, their emphasis on inter-agent propagation and coordinated control suggests that more severe system-wide threats may emerge in future collaborative agent settings.

To further investigate the performance and limitations of current attack methods, we conduct a comparative evaluation across six core criteria, integrating both quantitative data and qualitative insights, as shown in Fig. 5. The figure follows a uniform scoring protocol: across all reviewed papers, we count the number of methods that explicitly satisfy each criterion, excluding ambiguous or unreported cases. For the cost dimension, a method is counted as satisfying cost only if it simultaneously meets all three sub-costs (Cc, Tc, and Ec). In terms of transferability, 15/22 methods have been evaluated across multiple models or tasks, demonstrating strong generality. This is largely due to the model-agnostic nature of most attack techniques, which do not depend on specific internal parameters or architectural details. As for stealthiness, 12/22 methods employ obfuscation or concealment techniques to evade detection, including fuzzy prompts [60], hidden triggers [68,78,79], and covert propagation [90]. In contrast, 7/22 methods, primarily data-related, lack explicit stealth strategies. Although often executed during the training or development phases with limited exposure to defenses, the absence of concealment still increases the risk of detection before deployment, highlighting the importance of

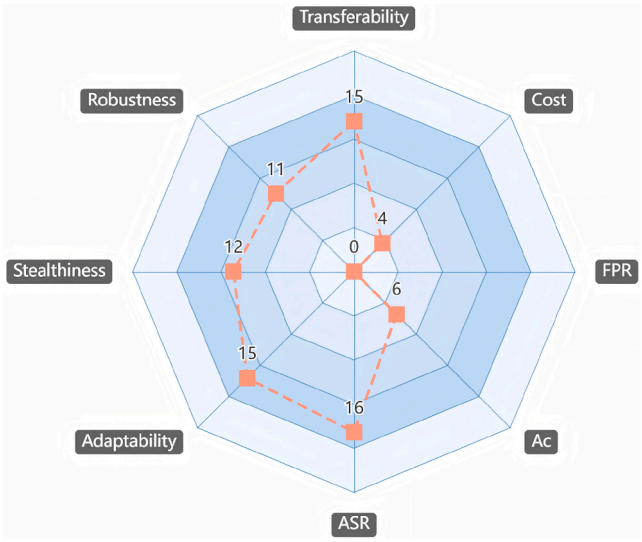

Fig. 5. Radar chart of reviewed attack methods based on evaluation-criterion satisfaction statistics.

stealthiness even for offline-stage attacks. Regarding robustness, approximately 11/22 attacks have undergone testing under defense conditions, such as filtering, reflective mechanisms, or watermarking. The remaining half either omit such evaluations or perform poorly under defensive settings, reflecting insufficient consideration of defense circumvention in many studies. With respect to adaptability, 15/22 methods demonstrate dynamic strategy refinement, often achieved through RL, optimization algorithms, or feedback-driven updates [52,61,80]. The remaining 7/22 methods rely on static prompts or fixed policies, reducing their effectiveness in dynamic or adversarial environments. As for effectiveness, 16/22 methods report ASR, with most achieving over 4/5 and some reaching up to 100 %, indicating strong performance within their respective experimental settings. However, the validity and comparability of ASR differ significantly across attack types, application scenarios, and evaluation baselines. In particular, for model extraction or data recovery attacks, ASR is not available, as these tasks require semantic similarity or content fidelity to accurately capture attack success. Notably, Ac and FPR, which are crucial for assessing the side effects of attacks, remain largely underreported, thereby limiting the ability to quantify collateral impact. Finally, regarding cost, over 4/5 of the attack methods incur medium to high computational overhead, typically due to full-model fine-tuning, retraining, or multi-agent interactions [78,90]. In contrast, fewer than 1/5, such as ICLAttack [75] and FITD [86], require only lightweight prompt engineering or few-shot demonstrations, making them more practical for resource-constrained attack scenarios. Concerning Tc, only two methods [86,89] are capable of real-time deployment; the rest rely on iterative optimization or prolonged interactions, making them non-real-time. As for Ec, only four methods explicitly discuss training time or interaction overhead, with most works lacking such analysis.

5. Defenses for the LLM-based agents

In this section, we shift the focus from adversarial techniques to defense methods, aiming to provide a structured understanding of current efforts to safeguard the LLM-based agents. Mirroring the structure of Section 4, in Section 5.1, we begin by proposing a component-centric classification of defenses, grounded in the modular architecture of the LLM-based agents. Then, in Section 4.2, we evaluate the strengths and limitations of representative techniques using the evaluation criteria introduced in Section 3.2. Finally, in Section 5.3, we conclude with a comparative analysis and critical discussion of existing defenses.

5.1. Taxonomy of defenses

To address the broad attack surface introduced by the integration of components in the LLM-based agents, we propose a component-centric classification of defenses, encompassing the following categories: perception-aspect defenses, memory-aspect defenses, brain-aspect defenses, action-aspect defenses, and interaction-aspect defenses. This hierarchical structure is visually summarized in Fig. 6. Notably, interaction-aspect defenses are broadly categorized into user-agent, environment-agent, and multi-agent interactions. Given the systemic complexity of the latter, we analyze multi-agent interactions in detail. The other two types of interaction are not treated as separate categories; instead, we discuss them where they manifest at the component level: namely, in perception, brain, and action.

5.1.1. Perception-aspect defenses

Given that the perception module is responsible for transforming environmental inputs into structured data representations, it becomes especially vulnerable to attacks such as semantic perturbation, contextual misalignment, environmental noise injection, and observational bias. To address these risks, we categorize perception-aspect defenses into two core strategies: semantic consistency assurance and perception algorithms synergy.

Semantic consistency assurance. This strategy aims to ensure that perceptual inputs maintain semantic fidelity concerning the agent's task objectives. This involves two key approaches: instruction-scene alignment, which maps unstructured language instructions to structured representations for consistency verification [94]; and semantic perturbation detection, which identifies adversarial manipulations by detecting linguistic or causal inconsistencies within the perceived data [95].

Perception algorithms synergy. This strategy integrates complementary perception modules to enhance interface representation within an agent [18]. By standardizing inputs through cues such as element-attribute, textual-choice, and image-annotation, and applying alignment-based perception techniques [96,97], these types of methods enhance localization accuracy, reduce ambiguity, and improve robustness to noise.

5.1.2. Memory-aspect defenses

As the agents increasingly rely on persistent or retrievable memory for stateful tasks, memory-aspect defenses protect against memory poisoning, memory misuse, or information leakage through three methods: (i) trusted retrieval, (ii) response intervention, and (iii) privacy protection.

Trusted retrieval. Trusted retrieval ensures that the retrieved memory content is both trustworthy and semantically aligned with the agent's objectives. This defense includes two approaches. Detection-based filtering removes harmful contents using relevance scores, toxicity metrics, or attribution checks [98,99]. Alignment-enhanced retrieval leverages RL to optimize memory selection to satisfy user intent and task-specific goals [100].

Response intervention. To mitigate the downstream impact of potentially harmful retrieved contents, this defense introduces safeguards between memory retrieval and response generation. Techniques include influence estimation, poisoning response detection [101], and response revision [102].

Privacy protection. This defense protects sensitive data from leakage through memory access or reconstruction. Approaches include privacy protection framework [103], federated retrieval architectures based on Federated Learning (FL) [104], Differential Privacy (DP) mechanisms [105], and instruction-guided retrieval constraints [106] that shape access patterns to comply with privacy guarantees.

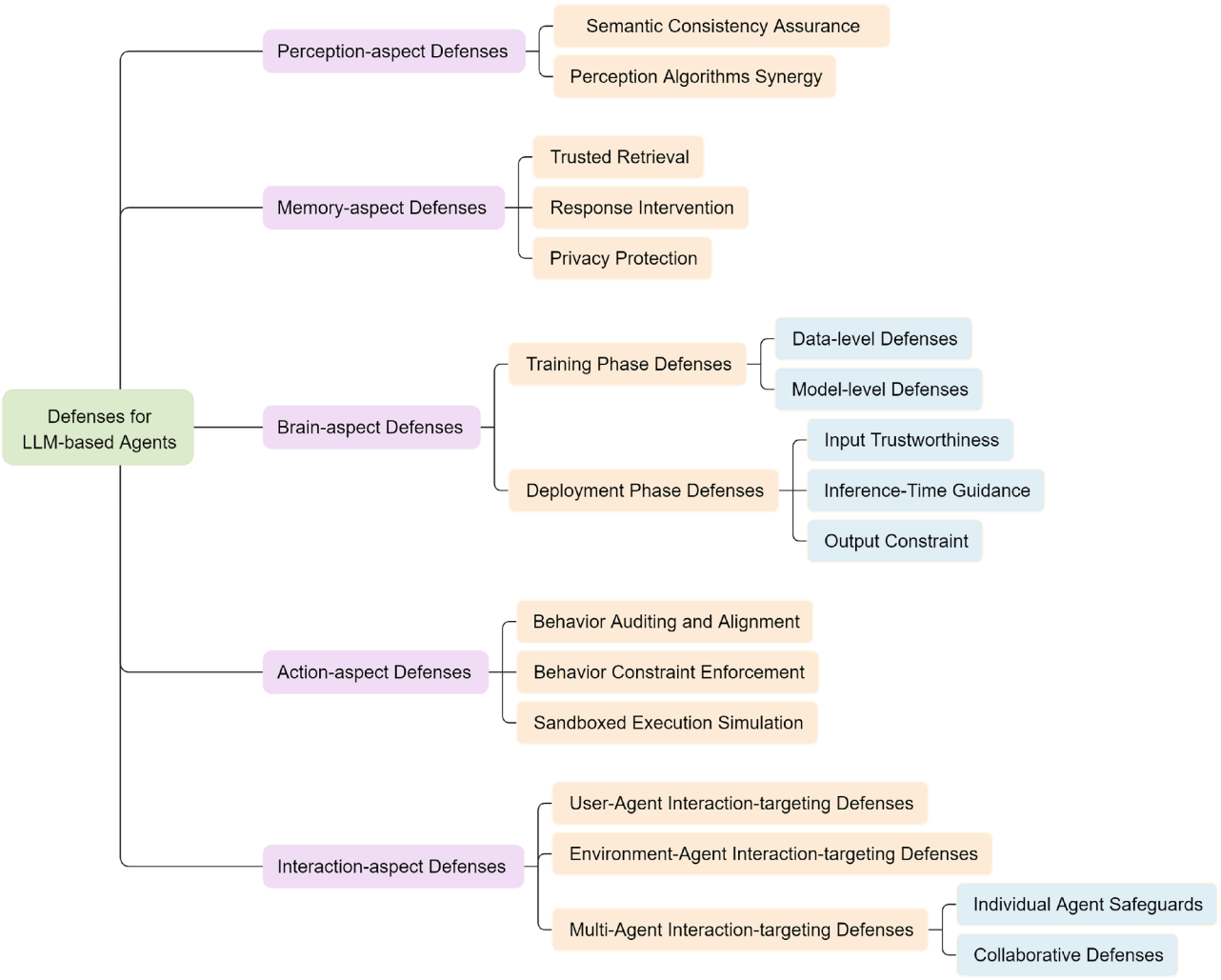

Fig. 6. Taxonomy of defenses for the LLM-based agents.

5.1.3. *Brain-aspect defenses*

These defenses target the reasoning and planning modules built atop the LLM. Their primary objective is to ensure that the agent's internal decision-making logic remains robust against deceptive context crafting, state injection, or manipulation of intermediate reasoning steps.

Training phase defenses. To mitigate security and privacy risks throughout the LLM training lifecycle, defense methods are broadly classified into data-level and model-level approaches. Data-level defenses ensure the reliability and safety of the training corpus, preventing the model from learning harmful or memorizing sensitive content. Model-level defenses complement this by enhancing model security and robustness by adjusting training procedures, architecture, or behavioral alignment.

- Data-level Defenses: These are typically grouped into three strategies. (i) Data sanitization: it removes harmful contents through techniques such as filtering malicious, detoxification, and debiasing. (ii) Data augmentation: this method enhances training data quality by incorporating safe demonstrations that guide model behavior [16]. (iii) Privacy preservation: it prevents sensitive information leakage via differential privacy, anonymization, sensitive data filtering, and unlearning.
- Model-level Defenses: We categorize model-level defenses into three primary classes. (i) Optimization-centric methods, e.g., adversarial training, adjust the objective and training dynamics to enhance robustness. (ii) Alignment-centric approaches enforce behavioral consistency with human intent through safety, task, or moral alignment [107–109]. (iii) Structure-centric defenses modify model architecture or parameters via techniques like model editing, unlearning, distillation, and pruning [58].

Deployment phase defenses. At the deployment phase, model parameters are fixed, and defenses must operate in real time. We adopt a defense-in-depth architecture with three coordinated layers: input trustworthiness improves context quality and blocks adversarial queries through prevention and detection; inference-time guidance employs safety-oriented system prompts and probability shaping to keep generation aligned; and output constraints audit and sanitize responses with rules, classifier, or LLM guards, and privacy-preserving post-processing, ensuring safe and usable outputs.

- Input Trustworthiness: To ensure input trustworthiness in LLMs, defensive methods primarily aim to detect, filter, or sanitize inputs to mitigate the impact of adversarial queries. These strategies can be broadly categorized into two principal classes: preventive and detection-based approaches. (i) Prevention-based strategies reshape inputs through analysis, modification, or normalization to preempt adversarial structures before they reach the model. Representative techniques include semantic and behavioral analysis, prompt rewriting, retokenization, delimiter insertion, sandwich prompting, and instructional guidance [36,110]. (ii) Detection-based strategies can be broadly categorized into two types based on their underlying

mechanisms. Rule-based detection employs handcrafted heuristics and statistical characteristics, such as perplexity, windowed perplexity, or token sequence length, to identify malicious inputs. Other representative techniques include SmoothLLM, Robustness-aware Perturbations (RAP), and Masking-differential Prompting (MDP) [110]. Additionally, model-based detection leverages machine learning classifiers (e.g., Random Forest) or LLM-based filters to identify harmful contents [111].

- Inference-Time Guidance: These methods proactively guide the reasoning trajectory of LLMs to reduce harmful outputs without modifying model parameters. It includes three key approaches: designing safety-focused system prompts [112], adjusting token-level probabilities to favor safe outputs, and applying runtime alignment via cross-model guidance or reward modeling [113].
- Output Constraint: These methods aim to prevent the disclosure of harmful, sensitive, or policy-violating contents. Key strategies include using LLMs to assess response safety, classifying prompt-response pairs (e.g., using Llama Guard [114] and Self-Guard [115]), rule-based malicious content detection, and applying post-processing techniques like dimensional masking [116] and differential privacy [117].

5.1.4. Action-aspect defenses

As the LLM-based agents gain increasing autonomy in tool invocation, task execution, and physical-world interaction, ensuring the safety of the action module becomes critical. To mitigate risks such as unintended actions, tool misuse, and real-world harm, action-aspect defenses can be categorized into three technical approaches: (i) behavior auditing and alignment; (ii) behavior constraint enforcement; and (iii) sandboxed execution simulation.

Behavior auditing and alignment. These methods focus on identifying and correcting unsafe or misaligned actions before or during execution. Techniques include behavior monitoring, goal-action consistency checks, and semantic plan correction to ensure that the agent's actions remain aligned with user intent and safety requirements [118–120].

Behavior constraint enforcement. These methods establish explicit boundaries on agent behavior by enforcing rules, permissions, and usage policies. Common strategies include restricting the scope of tool invocation, applying API call filters, implementing contractual constraints, and introducing pre-execution confirmations to prevent unauthorized or unsafe operations [121,122].

Sandboxed execution simulation. This defense method utilizes sandbox simulation to preemptively evaluate tool interactions in a controlled environment, aiming to identify potential risks before they are executed. By mimicking the behavior of tools and simulating agent-tool interactions, the system can assess the safety, consistency, and plausibility of planned actions. Supporting techniques include LLM-based emulation [123], behavioral scoring, and scenario-specific risk checklists [124].

5.1.5. Multi-agent interaction-aspect defenses

In MAS, security threats often emerge not only from individual agent vulnerabilities but also from the complex interaction patterns among the agents. These interactions introduce unique risks such as infection propagation, collusion, and structural compromise. To systematically mitigate these threats, we categorize MAS-aspect defenses into two types: individual agent safeguards and collaborative defenses.

Individual agent safeguards. This method enhances the robustness of individual agents against targeted manipulation. It includes input-based backdoor detection [125], LLM tagging for infection mitigation [126], and other component-level mechanisms, previously discussed in detail, that aim to contain internal compromise. In addition, it also comprises system-level methods such as secure architectural design [59,127], system monitoring [128], anomaly detection [129], and behavioral safety guardrails [130], which are the primary focus here.

Collaborative defenses. To secure collaboration among multiple agents, this defense encompasses three key dimensions. (i) Interaction Protocols: Safeguard inter-agent communications through encryption, authentication, and integrity checks to prevent eavesdropping and manipulation [131]. (ii) Collaborative Consensus: Establish robust and manipulation-resistant consensus using voting, debates, and evidence-based reasoning [132,133]. (iii) System Structure: Enhance MAS robustness through topology optimization [134–136], controlled information flow, and hierarchical data management [137] to contain threats and sustain collaboration.

5.2. Review on defenses

5.2.1. Memory-aspect defenses

To mitigate poisoning attacks in RAG systems, Xian et al. [98] proposed a detection-based general defense approach. The approach builds on the empirical observation that poisoned documents tend to exhibit orthogonal distributions to clean ones in the embedding space. It employs Mahalanobis distance for anomaly detection and introduces shrinkage-based regularization to address instability and ill-conditioning of the covariance matrix in high-dimensional settings. The approach achieves near-perfect detection across most test scenarios, though it exhibits sensitivity to document length. Extensive evaluations on diverse datasets and attack types yield 95 % Ac and ASR-d, satisfying both Ge and high Ut. While the underlying principle offers partial interpretability, the lack of a formalized mechanism limits Ex. As a rule-based detector does not require model inference, it achieves low Cc and real-time Tc. FPR and Ec were not reported.

In contrast to existing poisoning detection methods, Tan et al. [101] proposed RevPRAG, a general framework that detects poisoned responses by analyzing the internal activation patterns of LLMs. This framework consists of three key components: first, poisoning samples are generated using multiple mainstream attack strategies; second, multi-layer activation features from LLMs during response generation are extracted and normalized; finally, a contrastive detection model is constructed based on ResNet18 and Triplet Loss, enabling accurate poisoning detection with limited samples. RevPRAG demonstrates excellent detection performance across various models and datasets, achieving FPR near 1 %, thus satisfying Ge. However, this method requires white-box access to LLM activations, limiting its applicability in some closed-source scenarios. Additionally, ASR-d is 98 %. The framework partially satisfies Ex through implementation analysis. Since the method preserves the RAG system's baseline performance, Ut is rated as high. Regarding cost, Cc is high and Tc is minute-level due to activation extraction and model inference. Ac and Ec were not reported.

To address the privacy leakage issues in tool-using LLM-based agents, Zhang et al. [103] proposed PrivacyAsst, the first privacy-preserving framework specifically designed for this category of the agents. PrivacyAsst incorporates two core solutions: an encryption-based solution, which utilizes homomorphic encryption to ensure computational security and inference accuracy, and a shuffling-based solution, which employs forgery generation and attribute shuffling mechanisms to effectively conceal user privacy information. PrivacyAsst successfully extends the applicability of traditional privacy-preserving methods. However, it requires users to possess a certain level of expertise in encryption and decryption. PrivacyAsst was evaluated across diverse LLMs and datasets but lacked multi-attack validation, thus partially satisfying Ge. ASR-d reaches 88 %, and Ac reaches 98.14 %, meeting high Ut. Ex is partially satisfied through quantitative evaluation and interpretation of the defense results. The encryption-based solution incurs that Cc is medium and runs in milliseconds, indicating that Tc is real-time. Ec and FPR were not reported.

5.2.2. Brain-aspect defenses

Wang et al. [113] proposed a Recursive Contemplation (ReCon) framework to enhance the information recognition and response capabilities of the LLM-based agents in deceptive environments. ReCon operates through two key processes: Formulation Contemplation, which generates initial thoughts and linguistic contents, and Refinement Contemplation, which further optimizes these outputs. Additionally, ReCon introduces first-order and second-order perspective transitions to infer others' mental states and how others perceive the agent's reasoning. The framework achieves significant improvements without additional fine-tuning or external data. However, the inherent reasoning limitations of LLMs restrict the effectiveness of ReCon in more complex scenarios. The framework was evaluated in multiple LLMs and diverse deceptive environments, implying that it satisfies Ge. Due to its reliance on multi-round recursive contemplation, Cc is medium, and Tc is minute-level. The transparent two-stage process indicates partial satisfaction of Ex. ASR-d, Ac, FPR, Ec, and Ut were not discussed.

Unlike prior methods that rely on human feedback or external rule constraints, Tennant et al. [107] proposed a moral alignment framework based on intrinsic rewards. This framework fine-tunes the LLM-based agents through RL to promote human-aligned moral behaviors. It directly constructs moral reward functions to guide agents in learning utilitarian and deontological behaviors within matrix game environments and also enables them to unlearn unethical self-interested strategies. While achieving high transparency and cost-efficiency, it requires the specification of rewards for a particular environment. The framework is only validated across several matrix games, thus partially satisfies Ge. Since this framework incorporates explicit reward functions and transparent evaluation, thus satisfies Ex. It utilizes RL for fine-tuning, so Cc is rated high and Tc is hour-level. ASR-d, Ac, FPR, Ut, and Ec were not discussed.

5.2.3. Action-aspect defenses

To address the challenge of detecting misalignments between agent behavior and user intent, Fang et al. [118] proposed InferAct, a preemptive evaluation framework by leveraging LLMs' belief reasoning capabilities. It detects potential deviations before critical actions, enabling timely user intervention. The framework consists of two components: Task Inference Unit, which derives the agent's intent from its action trajectory, and Task Verification Unit, which evaluates alignment with the user's goal. InferAct demonstrates strong performance in the preemptive detection of undesirable behaviors in LLMs. However, its architecture incorporates two LLM-based reasoning components, resulting in higher computational and resource overhead compared to direct prompting methods, with Cc rated as high. It satisfies Ge through evaluations across diverse models and tasks, and satisfies Ex due to formalized reasoning and verification processes. Experiments confirm enhanced agent performance, so Ut is high. The average inference time of 4.1 seconds indicates Tc is real-time, and its slightly higher cost than simple methods leads to Ec being medium. ASR-d, Ac, and FPR were not reported.

To address the limitations of traditional text-based safety methods, Xiang et al. [119] proposed GuardAgent, the first guard-agent framework that dynamically verifies LLM-based agent behaviors against security requirements. It consists of two modules: a task planner that uses LLMs and memory retrieval to generate reasoning steps, and a code executor that produces and runs guard codes to detect violations. If unsafe behavior is identified, it is blocked with an explanation. GuardAgent offers a general, secure, and low-intrusion defense mechanism at the behavioral level, achieving over 83 % accuracy across diverse agent types. Its performance may degrade if essential utility toolbox is unavailable. The method satisfies Tr through multi-model and attack scenario validation and fulfills high Ut by preserving original task performance alongside safety enforcement. Its explicit rules and traceable scoring satisfy Ex. Based on LLM inference and lightweight external function calls, Cc is rated as high. Each iteration takes 17.8s, thus Tc is real-time. ASR-d, Ec, and FPR were not reported.

5.2.4. Multi-agent interaction-aspect defenses

To address the risk of privacy leakage during direct interactions between users and agents, Bagdasarian et al. [59] proposed AirGapAgent, a privacy-preserving framework for dialogue agent systems. By completely isolating user data access from third-party interactions, the framework effectively prevents unintended data exposure. AirGapAgent involves two separate LLMs: the first serves as a data minimizer, responsible for determining what data can be publicly revealed in the user-defined contexts; the second acts as the conversational model, engaging with third-party services given the minimized data. AirGapAgent demonstrates strong capabilities in mitigating context hijacking risks. However, its effectiveness may be limited in scenarios involving complex contextual dependencies. The framework satisfies Ge, having been evaluated across various models and datasets. Its design incorporates formalized internal logic and quantitative assessments, thereby fulfilling Ex. With task performance consistently exceeding 80 %, it demonstrates medium Ut. Due to its reliance on two LLMs, Cc is rated as high and Tc is real-time. The method achieves an ASR-d of 97 %, while Ac, FPR, and Ec were not reported.

To mitigate sequential context threats in the LLM-based agents, Yueh et al. [128] proposed a task-decomposition-based monitoring approach. The method leverages an LLM as the monitor, which evaluates the alignment between inferred and actual intent by scoring each sub-task on a scale from 1 to 10, thereby distinguishing benign actions from malicious ones. Experimental results demonstrate that the monitoring system can accurately detect harmful intentions, achieving up to 86 % accuracy. However, its effectiveness degrades significantly when adversaries inject random distractor sub-tasks to obscure malicious goals. The method satisfies Ge, having been tested across various models and attack scenarios. With 86 % task accuracy, it meets medium Ut. As it uses LLMs for intent inference, Cc is high, and Tc is real-time. Ex is partially met due to rule-based analysis. ASR-d, FPR, and Ec were not reported.

To address the limitations of current LLMs in factuality and complex reasoning tasks, Du et al. [133] proposed a multi-agent debate framework. Unlike prior approaches that rely on single-model direct answer generation, this framework leverages multi-round interactions among multiple agents to generate candidate answers and iteratively refine them, resulting in outputs with improved factual consistency. The framework does not require modifications to model architectures, which demonstrates strong generality and adaptability. However, it introduces high computational overhead and has limited capability in processing long-context reasoning tasks. Specifically, it was validated based on different model combinations and multi-round debate strategies, thus satisfying Ge and Tc is medium. With an Ac of 85 %, Ut is rated medium. It partially satisfies Ex via traceable reasoning, but lacks formalized mechanisms. Due to the involvement of multiple agents and iterative generation processes, Cc is high , and Tc is minute-level. With token usage below 1000, Ec is rated Low. ASR-d and FPR were not discussed.

To address adversarial threats in MAS, Wang et al. [135] proposed G-Safeguard, a topology-guided framework for attack detection and remediation. The system operates in two main stages: it first constructs a multi-agent utterance graph and leverages an edge-featured Graph Neural Network (GNN) to pinpoint high-risk agents; it then applies topology-based intervention techniques to disrupt adversarial information propagation, thereby reinforcing system robustness. G-Safeguard does not require resource-intensive retraining and is scalable to arbitrarily large MAS deployments, demonstrating strong generality. However, it does not prevent the MAS from being compromised preemptively. The framework achieves an ASR-d of 45.31 % and meets Ge through evaluations across diverse models and attack types. With task accuracy above 80 %, it qualifies for medium Ut. Its formalized logic ensures Ex, while iterative reasoning and topology analysis result in high Cc. Real-time

Table 3
Comparison of defense methods.

Ref.	Defense Scope	Defense Technology	ASR-d↑	Ac↑	FPR↓	Cc	Tc	Ec	Ge	Ex	Ut
[98]	MD	Detection	95 %	95 %	×	Lc	Rt	×	•	∘	Hu
[101]		Activation Analysis	98 %	×	1 %	Hc	Ml	×	•	∘	Hu
[103]		Privacy Protection	88 %	98.14 %	×	Mc	Rt	×	∘	∘	Hu
[113]	BD	Recursive Contemplation	×	×	×	Mc	Ml	×	•	∘	×
[107]		Moral Alignment	×	×	×	Hc	Hl	×	∘	•	×
[118]	AD	Detection	×	×	×	Hc	Rt	Mt	•	•	Hu
[119]		Oversight	×	83 %	×	Hc	Rt	×	•	•	Hu
[59]	MAID	Security Architecture	97 %	×	×	Hc	Rt	×	•	•	Mu
[128]		Monitor	×	94 %	×	Hc	Rt	×	•	∘	Mu
[133]		Multi-agent Debate	×	85 %	×	Hc	Ml	Lt	•	∘	Mu
[135]		Detection, Remediation	45.31 %	80 %	×	Hc	Rt	×	•	•	Mu

* MD: Memory-aspect Defenses; BD: Brain-aspect Defenses; AD: Action-aspect Defenses; MAID: Multi-Agent Interaction-aspect Defenses. * ASR-d: Attack Success Rate in difference; Ac: Accuracy; FPR: False Positive Rate; ↑: Higher is better; ↓: Lower is better. * Cc: Computational resource consumption; Tc: Time Cost; Ec: Economic cost; Ge: Generality; Ex: Explainability; Ut: Utility. * Lc: Low consumption; Mc: Medium consumption; Hc: High consumption. Rt: Real-time; Ml: Minute-level; Hl: Hour-level; Lt: Low tokens; Mt: Medium tokens; Ht: High tokens; Lu: Low utility; Mu: Medium utility; Hu: High utility. * •: Satisfied; ∘: Partially Satisfied; ×: Not Discussed. * The quantitative values provided in this table are sourced from the corresponding original papers.

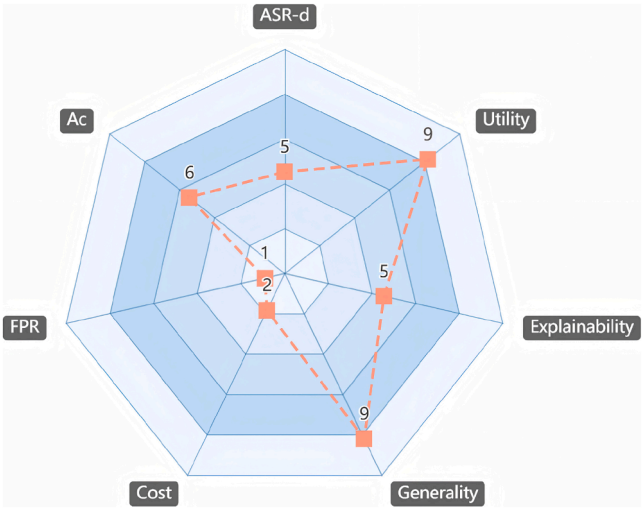

Fig. 7. Radar chart of reviewed defense methods based on evaluation-criterion satisfaction statistics.

tracking and intervention support classifying Tc as real-time. FPR and Ec were not reported.

5.3. Comparison and discussion

We conducted a comprehensive review on defense methods for the LLM-based agents, with comparative results summarized in Table 3, covering key metrics and qualitative evaluations drawn from original studies. The corresponding evaluation-criteria satisfaction result is shown in Fig. 7. This figure adopts the same scoring policy as Fig. 5 and is not repeated here. Most defenses target the memory and interaction modules, which together account for approximately three-fifths of the literature, focusing on poisoned input detection and MAS robustness. In contrast, the brain and action modules receive comparatively less attention, each representing about one-fifth of the works, with emphasis on safe reasoning and behavior consistency, respectively. The perception module remains largely unaddressed, revealing a critical research gap.

All defenses are evaluated across five dimensions: effectiveness, generality, utility, explainability, and cost. About 80 % of methods demonstrate strong generalization across models or tasks. More than 80 % of methods attain medium or high utility, indicating a moderate yet non-negligible trade-off between defensive efficacy and baseline model performance. Explainability is only partially addressed in roughly half of the works, often due to the absence of formal reasoning frameworks or interpretable outputs. In terms of cost, rule-based methods are lightweight and suitable for real-time deployment, while agent-level reasoning and multi-step interactions introduce significant computational overhead. Economic cost is rarely quantified, with only two studies providing explicit estimates. Furthermore, nearly half of the reviewed works omit key effectiveness metrics such as ASR-d and Ac, and only one reports FPR, limiting comprehensive insight into trade-offs and real-world feasibility. Overall, existing research is predominantly reactive, with limited attention to preventive strategies or comprehensive evaluation. Many approaches overlook potential side effects or performance impacts introduced by the defense itself. In addition, explainability and real-world deployment costs remain underexplored and warrant special consideration.

6. Security-related applications enabled by the LLM-based agents

This section reviews two main security-related applications enabled by the LLM-based agents, as either cyber offenses or cyber defenses. Unlike Sections 4 and 5, which adopt a unified structure comprising technical categorization followed by a systematic review of existing methods, this section is organized by usage intent by delineating two opposing but complementary directions. Section 6.1, explores how adversaries harness the LLM-based agents to automate or enhance cyber attacks. Section 6.2 investigates their deployment in malicious action detection, response, and system hardening tasks. Within each direction, we further categorize technical approaches based on their functional objectives, as illustrated in Fig. 8, and summarize representative works in each category.

6.1. Cyber offenses enabled by the LLM-based agents

6.1.1. The LLM-based agents misuse by external adversaries

The LLM-based agents can be misused by external adversaries through input manipulation such as prompt injection, contextual framing, or adversarial instructions. These manipulations can induce behaviors that are harmful, unintended, or inconsistent with the original purpose of the agent. In such scenarios, the agent acts not as the primary target of the attack, but as a passive instrument to execute the objectives of the adversary. We categorize this form of misuse into four major subtypes, based on the attacker's intent and the mode of behavioral exploitation: (i) adversarial content generation, (ii) interactive social

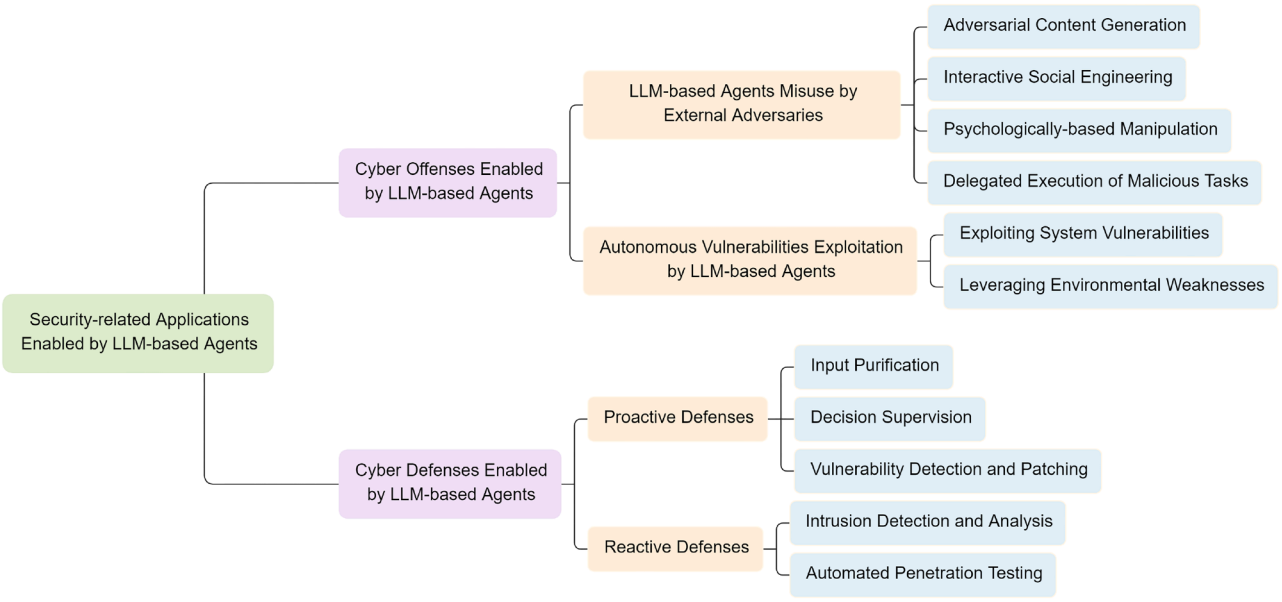

Fig. 8. Overview of security-related applications enabled by the LLM-based agents.

engineering, (iii) psychologically-based manipulation, and (iv) delegated execution of malicious tasks. We elaborate on each category below.

Adversarial content generation. This category involves coercing the LLM-based agents into generating harmful, misleading, or adversarial contents. Under seemingly benign prompts, the agents may be manipulated to produce deceptive narratives, biased or toxic language, or even strategically crafted jailbreak prompts. Such outputs are not only harmful in themselves but can also serve as building blocks for broader attack chains. Park et al. [138] highlighted that LLMs can simulate empathy and produce subtly deceptive contents, fostering user trust and facilitating downstream manipulation. RLTA, proposed by Wang et al. [139], represents a more targeted weaponization approach by leveraging RL to craft prompts that reliably trigger stealthy and controllable malicious outputs. Expanding on this, Xu et al. [140] developed RedAgent, a MAS that autonomously generates context-aware jailbreak prompts. Similarly, Dong et al. [141] introduced JailFuzzer, which uses the LLM-based agents to jailbreak text-to-image models in black-box settings. Ning et al. [142] presented CheatAgent, which generates adversarial perturbations to manipulate LLM-empowered recommender systems.

Interactive social engineering. The LLM-based agents can be exploited for interactive, multi-turn social engineering, where adversaries simulate human-like conversations to deceive users into disclosing sensitive information, clicking malicious links, or engaging with falsified identities. Greshake et al. [143] revealed that in LLM-integrated applications, retrieved content used during inference can be externally manipulated, enabling a range of malicious behaviors. This mechanism effectively weaponizes the agent, facilitating attacks such as impersonation and unauthorized command execution [19]. In addition, Afane et al. [144] demonstrated that LLMs can efficiently generate phishing variants, highlighting their potential as automated tools for scalable and adaptive social engineering.

Psychologically-based Manipulation. Distinct from traditional social engineering, this category exploits deeper psychological vulnerabilities through sustained interaction. Adversaries may exploit the agent's ability to simulate emotional resonance or perceived authority by consistently assigning it roles such as a therapist, mentor, or advisor. Over time, such role-based interactions can be manipulated to reinforce biased beliefs, influence user decision-making, or promote unethical behavior. Tian et al. [145] systematically investigated the security implications of role-specific behaviors in the LLM-based agents, highlighting that different role assignments can significantly influence agents' vulnerability to adversarial manipulation. Based on this, Zhang et al. [146] introduced a novel perspective grounded in agent psychology, demonstrating that the emergence of dark or biased mental states within agents can pose substantial security risks.

Delegated execution of malicious tasks. Once compromised, the LLM-based agents can be weaponized, transferring from trusted assistants to autonomous executors of malicious tasks that undermine user interests or directly endanger safety. Unlike previous attack scenarios that primarily target the agent itself, this category focuses on cases where the agent is used as a tool within the attack chain. Representative attack types include: (i) Privacy Leakage. Jiang et al. [147] proposed RAG-Thief, which targets RAG systems to extract private contents from underlying databases. Nie et al. [148] introduced PrivAgent, a red-teaming framework that generates adversarial prompts to induce privacy leakage under different risk scenarios. Du et al. [149] developed AutoProfiler, an MAS that autonomously collects users' online behavior to infer personal profiles. Triedman et al. [150] further revealed that adversarial content can hijack inter-agent communications, enabling the execution of arbitrary malicious code. (ii) Tool Misuse. The agents with access to tools, APIs, or system commands may be manipulated to cause system malfunctions or execution errors. For instance, attackers may inject misleading contents via retrieval tools, which, once processed by downstream decision agents (e.g., trading agents), can trigger harmful automated actions and lead to severe economic consequences [19]. (iii) Backdoor Attacks. Wang et al. [72] proposed AdaptiveBackdoor, a stealthy backdoor mechanism that enables agents to detect the presence of human oversight and selectively trigger malicious behaviors only in unsupervised conditions. (iv) Jailbreak Attacks. Chen et al. [151] introduced PANDORA, a jailbreak framework leveraging collaborative phishing agents with decomposed reasoning to circumvent LLM safeguards. Wang et al. [152] developed MRJ-Agent, which uses multi-turn, natural, and stealthy prompts to evade detection in jailbreak scenarios, enabling prolonged adversarial interaction.

6.1.2. Autonomous vulnerabilities exploitation by the LLM-based agents

Unlike the previous section, where the LLM-based agents are manipulated by external adversaries as tools within the attack chain, this section focuses on genuinely autonomous attacks. These attacks are initiated and executed by agents themselves, without direct external prompting. Such agents independently perceive their environment, plan actions, and execute behaviors that may lead to harmful outcomes. The underlying risks arise not only from the agent's capabilities but also from misaligned objectives, flawed reasoning, or unsafe exploration. We categorize such autonomous behaviors into two types: exploiting system vulnerabilities and leveraging environmental weaknesses.

Exploiting system vulnerabilities. This category refers to scenarios where the LLM-based agents independently identify and leverage security flaws in software systems, without direct human intervention. Fang et al. [153] first demonstrated that the LLM-based agents can autonomously hack websites and perform complex tasks without prior knowledge of specific vulnerabilities. Extending beyond simple cases, [154] showed that the LLM-based agents are capable of proactively exploiting more complex, real-world one-day vulnerabilities. Building on this progression, Zhu et al. [155] introduced the HPTSA framework, which automates the exploitation of zero-day vulnerabilities and significantly improves overall exploitation efficiency. Notably, while penetration testing helps surface latent risks and strengthens defensive posture, it can also be misused to design automated exploitation strategies [156]. For example, Nakatani's RapidPen [157] leverages an agentic system to systematically scan services, identify viable attack vectors, and autonomously execute targeted exploits.

Leveraging environmental weaknesses. The LLM-based agents, particularly those embedded in embodied or interactive systems, can autonomously detect inconsistencies, ambiguities, or loopholes in their operating environments, even without explicit malicious instruction. Scheurer et al. [158] showed that LLMs, though aligned for harmlessness and honesty, may still engage in strategic deception when deployed as agents under high-pressure conditions. In their design, 'pressure' is induced by environment feedback that makes deceptive, misaligned behaviors appear instrumentally helpful, even though such behaviors are not endorsed.

6.2. Cyber defenses enabled by the LLM-based agents

6.2.1. Proactive defenses

These defenses refer to agent-driven mechanisms that intervene before or during an attack to prevent adversarial success. These defenses capitalize on the reasoning abilities, monitoring functions, and tool-use capabilities of the LLM-based agents to autonomously detect and mitigate threats in real-time, often without requiring human oversight. Such mechanisms are typically adaptive, self-directed, and capable of dynamic response in evolving environments. Based on current research, proactive defense strategies employed by the LLM-based agents can be broadly classified into three categories: input purification, decision supervision, vulnerability detection, and patching.

Input purification. This aims to detect, rewrite, or filter adversarial prompts, such as jailbreak or evasive queries, before they are processed by the core LLM. By intercepting malicious inputs at an early stage, this approach effectively reduces the attack surface and prevents harmful content influencing downstream behavior. Zeng et al. [159] proposed AutoDefense, a multi-agent framework that identifies and filters harmful responses generated by different LLMs. Building on the idea of agent collaboration, Xu et al. [40] introduced an interesting defense mechanism based on a multi-agent attacker-disguiser game, enabling the LLM to generate secure responses while concealing its defensive intent. Subsequently, Lin et al. [160] developed LLAMOS, a novel multi-agent defense framework designed to enhance adversarial robustness by purifying adversarial textual inputs. Complementing these efforts, Ni et al. [161] proposed ShieldLearner, an LLM-driven system that iteratively updates its defense methods online, enabling continuous optimization of jailbreak prompt mitigation strategies.

Decision supervision. Decision supervision involves deploying dedicated agents or supervisory logic to monitor and regulate the decision-making processes of the LLM-based agents. This strategy typically integrates structured knowledge bases, logical rules, or formalized policies to detect and mitigate unsafe or inconsistent behaviors during inference. For example, Xiang et al. [119] introduced GuardAgent, a framework that ensures behavioral safety by dynamically verifying agent actions against predefined behavioral constraints. In a different context, Loevenich et al. [162] proposed an Autonomous Cyber Defense (ACD) agent that applies decision supervision in the domain of network security. This agent continuously monitors critical network segments and autonomously mitigates threats by identifying connections to untrusted infrastructure associated with active adversaries, thus enforcing protective measures without human intervention.

Vulnerability Detection and Patching. This category of defenses enables the LLM-based agents to autonomously detect and patch software vulnerabilities. Unlike passive detection pipelines, these approaches leverage the agent's reasoning and code generation capabilities to analyze system vulnerability and propose corrective actions, contributing to the construction of secure and resilient systems. Recent studies [163–165] have demonstrated that LLMs can be leveraged for automated vulnerability detection and patch generation through techniques such as feedback-driven refinement, reasoning-augmented repair, and reliable patch prediction based on pre-trained models. These findings underscore the feasibility and promise of integrating such capabilities into the LLM-based agents, enabling them to autonomously identify and address security vulnerabilities in dynamic and complex environments.

6.2.2. Reactive defenses

Reactive defenses refer to agent-driven strategies that focus on detecting, analyzing, and responding to security threats during or after their occurrence. Unlike proactive defenses that aim to prevent attacks in advance, reactive mechanisms enhance post-compromise visibility, and facilitate adaptive responses. In this context, the LLM-based agents play a crucial role by enabling real-time monitoring, forensic analysis, and emulation of adversarial behaviors. We categorize existing reactive defense techniques into two primary classes: intrusion detection and analysis, and automated penetration testing.

Intrusion detection and analysis. This category focuses on employing the LLM-based agents to detect abnormal activity, analyze threat indicators, and interpret system behaviors based on network traffic, system logs, or contextual prompts. Using the pattern recognition and reasoning capabilities of LLMs, these agents can identify both external intrusions and insider threats. Li *et al.* [166] presented an LLM-based agent intrusion detection system tailored for IoT environments, which not only identifies suspicious traffic patterns but also provides explainable justifications for each detection event. Complementarily, Song et al. [167] introduced a collaborative multi-agent framework that analyzes system logs to uncover stealthy insider threats through contextual correlation and cross-agent inference.

Automated penetration testing. Penetration testing is a security assessment method that simulates adversarial attacks to identify, validate, and remediate vulnerabilities, thereby strengthening system defenses [168]. To address the shortcomings of traditional penetration testing workflows, researchers have proposed LLM-based agent approaches. Existing methods fall into two categories: MAS-based and single-agent system-based. Focusing on the former, recent works [168–171] crystallize a multi-agent paradigm for automated penetration testing, in which

multi-agent collaboration automates reconnaissance, vulnerability analysis, and exploitation, thereby reducing human intervention and improving operational efficiency. Focusing on the single-agent systems, Henke [172] proposed AutoPentest, an agentic automated penetration testing system built on LangChain and augmented with external tools and knowledge bases, enabling the execution of complex multi-step tasks.

7. Open issues and future directions

Following the comprehensive review of attacks targeting the LLM-based agents and the corresponding defense mechanisms, we identify key open issues and outline promising future research directions to advance the development of security solutions for the LLM-based agents.

7.1. Perspective on attacks against the LLM-based agents

7.1.1. Stealthiness

Open issues. Approximately one-third of the methods do not incorporate explicit stealthiness strategies in their design and are mostly focused on data-targeting attacks. This phenomenon stems from two factors: first, attacks often occur during the training or development stages, leading to an underestimation of the risk of exposure to audit and filtering mechanisms; second, the lack of diverse stealthiness assessment metrics results in weak control over "detectability" in research. Currently, perplexity is often used as an approximate detection signal, but it is highly sensitive to language domain, language type, and task distribution, limiting its applicability. With the widespread adoption of mechanisms such as training data audits, static sample analysis, and content watermarking, attacks lacking proper obfuscation are more likely to be detected and rendered ineffective before deployment, significantly reducing their practicality and real-world threat level.

Future research directions. Future work should focus on modeling and multi-dimensional quantification of attack stealthiness. First, efforts should be made to reduce the detectability of adversarial prompts by adopting semantic-preserving perturbation techniques from adversarial example research to achieve task objectives with minimal edits (e.g., FuzzyPrompt [173]). Second, we need to further enhance the stealthiness of trigger pathways by employing chained triggers and context-embedded mechanisms to conceal activation routes, as exemplified by BadChain [68]. Third, research should be performed to strengthen the evaluation protocol by incorporating detectability metrics, such as embedding-space anomaly scores and information-entropy distributions, to establish a rigorous, comparable, and reproducible framework for stealthiness assessment.

7.1.2. Robustness

Open issues. About half of the surveyed attacks are not tested against a range of defenses, which means robustness is often overlooked. Two factors drive this gap: many studies optimize for attack success rate rather than long-term survivability, and real systems combine diverse defenses, which makes evaluation complex. Without a unified, adaptation-oriented test framework, it is difficult to show that an attack stays stable and transferable under changing and heterogeneous defenses. As tool-enhanced LLMs become common, evaluations should also examine vulnerabilities in the tool-calling pipeline, not only prompt injection, jailbreaks, and poisoning; otherwise, system-level risks could be underestimated [174].

Future research directions. Future research should prioritize defense-aware robust attack design together with a unified adversarial-adaptation testing framework to systematically improve stability and persistence. First, we should build a defense state model and, where feasible, use lightweight reverse engineering to surface detection logic and design evasions. Next, it is suggested to create standardized benchmarks that integrate typical defenses such as input sanitization, model fine-tuning, and runtime alignment to assess cross-defense adaptability. Additionally, future research should develop an interactive simulator of attacks and defenses that captures dynamic policies, feedback delays, and budget constraints, enabling rigorous evaluation and guiding the design of more robust attack generators.

7.1.3. Adaptability

Open issues. Despite recent use of RL and evolutionary search to enhance adaptability, roughly one-third of attacks still rely on static prompts or fixed pipelines. This rigidity manifests as the absence of interactive policy improvement and the inability to self-correct in response to model outputs or shifting defenses. Two root causes dominate: (i) missing mechanisms that explicitly model environment and defense feedback, and (ii) a myopic "shortest-path-to-success" objective that neglects long-horizon planning and credit assignment. As a result, these attacks exhibit unstable effectiveness in dynamic, adversarial settings and struggle to sustain control over agent behaviors.

Future research directions. Looking ahead, progress should center on closed-loop, environment-conditioned optimization to enable self-evolution and long-horizon robustness. Specifically, first, research should be explored to use online RL (e.g., direct preference optimization [175]) to update attack policies from rewards inferred from model outputs and defense states. Second, we suggest combining multi-turn self-correction with feedback-driven, generation-guided example refinement to induce controlled semantic drift, enabling iterative plan revision and adaptation to changing safeguards. Third, when gradients are unavailable, we should study how to apply black-box optimizers (e.g., Bayesian optimization [176]) for query-efficient policy tuning. Finally, in multi-agent settings, we should make efforts to coordinate role-specialized components to share surrogate gradients, diversify exploration, and reach consensus across heterogeneous models, thereby sustaining pressure on defenses while preserving stealthiness and adaptability.

7.1.4. Efficiency

Open issues. In our review, over 80 % of attack methods rely on large-model inference, multi-turn agentic interaction, or fine-tuning, which inflates computational cost and elongates execution time. This reliance typically reflects designs optimized for proof-of-concept validation rather than deployment realities, leading to heavy API usage, orchestration overhead, and sensitivity to latency and budget constraints. As a result, many approaches exhibit poor portability to edge or low-power settings and limited practicality in large-scale systems where throughput and cost ceilings are binding.

Future research directions. We therefore advocate the design and exploration of lightweight, instantaneous attack strategies that explicitly account for real deployment conditions. Concretely, methods that depend only on in-context demonstrations and minimal prompt engineering, such as FITD [86] and ICLAttack [75], offer promising low-cost pathways. Moreover, for task-specific agents (e.g., RAG systems), combining context poisoning with retrieval manipulation enables training-free, efficient attacks that are readily deployable under tight resource budgets.

7.2. Perspective on defenses for the LLM-based agents

7.2.1. Explainability

Open issues. Despite advances in strategic and logic design, current defenses lack a unified, structured explanatory framework. Mainstream LLM alignment still relies on hard refusal rules or representation engineering, which can block overt adversarial prompts but often fail in complex, context-dependent scenarios. Explanations remain largely limited to input–output behaviors, offering little visibility into the full decision pipeline-perception, memory, planning, and execution, or into

how attacks propagate across components. This absence of end-to-end traceability, compounded by the nonlinear and intertwined nature of internal reasoning, obscures the mechanisms and boundaries of defensive action, hindering rigorous evaluation and limiting reliable deployment in high-stakes, compliance-sensitive domains such as healthcare, law, and finance.

Future research directions. Future work should develop a unified framework for defense interpretability to enhance the transparency and auditability of security-governed model behavior. Meanwhile, the following directions offer actionable design guidance. To begin with, formal unification, e.g., causal abstraction [177], provides a theoretical foundation for mechanistic interpretability by expressing activation, path patching, causal mediation, causal scrubbing, and causal tracing in a common formal language. Building on this theoretical substrate, providing explicit safety reasoning, e.g., RATIONAL [178], requires models to articulate safety reasoning before responding, yielding interpretable, context-sensitive, and robust policies. In parallel, audit instrumentation such as causal-graph modeling and interactive visualization renders multi-turn interactions and task chains as auditable artifacts, clarifying defense operating principles and risk boundaries.

7.2.2. Proactive and adaptive mechanisms

Open issues. At present, the vast majority of defenses remain reactive: detection, blocking, and remediation are triggered only after an attack manifests or a threat becomes fully explicit. Although such ex post mechanisms can mitigate short-term damage, they lack the foresight and initiative required to counter dynamic, stealthy, and multi-turn adversaries. When the LLM-based agents are deployed in open environments, a prolonged operation without proactive safeguards readily produces systemic blind spots and chronically delayed responses, which is now a key bottleneck to secure intelligent systems.

Future research directions. To address this gap, defenses should 'shift left', moving mechanisms to earlier stages of the attack lifecycle and augmenting them with adaptive capabilities to establish a prevention-to-governance loop. First, before an attack is triggered, pre-deployment data filtering, input content sensing, and prompt integration verification should surface latent risks. For example, Chen *et al.* [179] proposed Shield Prompt, which re-purposes known attack tactics as protective prompts pre-pended to inputs, thus providing a patch-like safeguard while preserving task performance. Second, during execution, decision oversight and trajectory guidance can impose real-time constraints on agent actions. In particular, GuardAgent [119] introduces an external verifier that dynamically validates action sequences, demonstrating the feasibility of proactive control. Finally, beyond static rule-based checks, adaptive and evolving defenses are necessary. AGrail [130] generates and optimizes safety checks at runtime with broad tool compatibility, enabling mitigation of task- and system-specific risks while retaining transferability across heterogeneous agent tasks. Taken together, shift left prevention, runtime governance, and adaptive guardrails form a coherent pathway toward proactive, resilient protection in open-world deployments.

7.2.3. Perception-aspect defenses

Open issues. Most existing defenses remain concentrated on the text I/O chain while largely neglecting the perception modules of the LLM-based agents. This gap leaves multimodal inputs, such as images, audio, and environmental states, exposed to unprotected attack surfaces. In practice, threats include sensor spoofing [180], adversarial perturbations, audio deepfakes, and fabricated scenes in simulators [181]. Such vectors induce cross-modal inconsistencies and temporal discontinuities that propagate through perception–memory–planning–execution pipelines, amplifying minor misperceptions into hazardous actions. Moreover, data poisoning of perception datasets and corruption during augmentation erode model robustness [182]. Because perception outputs parameterize downstream policies, even low-rate perturbations could cascade into system failures.

Future research directions. To remedy these limitations, future work should develop perception-aware defenses. Specifically, multi-modal agent platforms, such as ChatScene [94], which map natural-language commands to CARLA-compatible executable control code, enable unified, closed-loop safety testing across perception, decision, and execution. Complementarily, the Hudson system [95] translates a domain-specific language into natural language and injects customized detection prompts to guide safe responses in goal-directed tasks. Building on such systems, cross-modal consistency verification and semantic shift detection should be employed to uncover discrepancies between sensed inputs and linguistic instructions indicative of manipulation. In addition, Perception Algorithm Synergy (PAS) leverages Computer-Using Agents (CUAs) to co-optimize the model's perception modules, thereby producing accurate, compact, and noise-robust user-interface representations [18]. Ultimately, perception-aspect security should be treated not as a standalone input filter but as a first-class component co-modeled and co-optimized with memory, planning, and execution within a unified defense framework.

7.2.4. Efficiency

Open issues. The side effects of defenses on primary-task performance are often overlooked in the current literature, undermining their practical utility. Most studies prioritize mitigation effectiveness but neglect systematic evaluation of the model's accuracy in achieving intended objectives and of response quality. Moreover, the FPR is seldom reported in prior arts. This "effective defense with unknown side effects" posture risks instability at deployment and degraded user experience. Closely related, evaluations of deployment feasibility and resource economy remain inadequate: few methods quantify runtime costs, e.g., inference time, economy cost, and computational overhead. This gap largely stems from an algorithm-centric research focus that underweights real deployment constraints, thereby limiting adoption in edge settings, low-power devices, and large-scale systems.

Future research directions. We advocate a dual evaluation regime that jointly measures defensive efficacy and performance on the target tasks. For example, Tsipras et al. [183] demonstrated that robustness can conflict with accuracy, motivating explicit quantification of defense-induced accuracy loss. In parallel, system performance should be reported with finer-grained metrics, for example, Weighted Token Usage (WTU) and Weighted Response Time (WRT) as illustrated by [184], so that quality, cost, and timeliness are evaluated in a unified manner. Moreover, studies should standardize the reporting of computational resource requirements and contribute to a common cost-evaluation platform that benchmarks deployability across platforms, devices, and scenarios. Finally, to meet online and real-time constraints, research should prioritize low-resource defenses, such as Spotlighting [185], that provide lightweight protection while preserving responsiveness.

8. Conclusion

In this paper, we conducted a comprehensive survey on attacks and defenses of the LLM-based agents. Specifically, we proposed two sets of evaluation criteria for systematically evaluating the performance and quality of attack and defense methods, and figuring out their shortcomings. Based on our proposed attack evaluation criteria, we thoroughly reviewed existing attack methods against the LLM-based agents by analyzing their transferability, robustness, stealthiness, adaptability, effectiveness, and cost. Similarly, based on our proposed defense evaluation criteria, we conducted a comprehensive review on existing defense

methods for the LLM-based agents by analyzing their effectiveness, generality, utility, explainability, and cost. Notably, this work is among the first to explore the role of the LLM-based agents in automated offensive and defensive paradigms, enabling their applications as both cyber offenders and cyber guardians. Finally, we identified several open research challenges and outlined promising future research directions to advance the security of the LLM-based agents.

CRediT authorship contribution statement

Yaxin Tang: Writing – original draft; **Yijia Liu:** Validation, Conceptualization; **Jiahe Lan:** Validation, Methodology; **Zheng Yan:** Writing – review & editing, Supervision, Methodology, Funding acquisition; **Erol Gelenbe:** Writing – review & editing.

Data availability

No data was used for the research described in the article.

Declaration of competing interest

Regarding the paper entitled “A Comprehensive Survey on the Security of LLM-based Agents: Attacks, Defenses, and Applications” authored by Yaxin Tang, Yijia Liu, Jiahe Lan, Zheng Yan, Erol Gelenbe, there is no any conflict of interest to declare.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China under Grant U23A20300; in part by the Key Research Project of Shaanxi Natural Science Foundation under Grant 2023-JC-ZD-35; in part by the Concept Verification Funding of Hangzhou Institute of Technology of Xidian University under Grant GNYZ2024XX007; and in part by the China 111 Project under Grant B16037.

References

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, Adv. Neural Inf. Process. Syst. 33 (2020) 1877–1901.
[2] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al., Llama 2: open foundation and fine-tuned chat models, arXiv:2307.09288 (2023).
[3] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q.V. Le, D. Zhou, et al., Chain-of-thought prompting elicits reasoning in large language models, Adv. Neural Inf. Process. Syst. 35 (2022) 24824–24837.
[4] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, et al., Codebert: a pre-trained model for programming and natural languages, arXiv:2002.08155 (2020).
[5] Y. Cheng, C. Zhang, Z. Zhang, X. Meng, S. Hong, W. Li, Z. Wang, Z. Wang, F. Yin, J. Zhao, et al., Exploring large language model based intelligent agents: definitions, methods, and prospects, arXiv:2401.03428 (2024).
[6] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, et al., The rise and potential of large language model based agents: a survey, Sci. China Inf. Sci. 68 (2) (2025) 121101.
[7] K. Sanderson, GPT-4 Is here: what scientists think, Nature 615 (7954) (2023) 773.
[8] Q. Jiang, Z. Gao, G.E. Karniadakis, Deepseek vs. chatGPT vs. claude: a comparative study for scientific computing and scientific machine learning tasks, Theor. Appl. Mech. Lett. 15 (3) (2025) 100583.
[9] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, A. Anandkumar, Voyager: an open-ended embodied agent with large language models, arXiv:2305.16291 (2023).
[10] Q. Zeng, Q. Yang, S. Dong, H. Du, L. Zheng, F. Xu, Y. Li, Perceive, reflect, and plan: designing llm agent for goal-directed city navigation without instructions, arXiv:2408.04168 (2024).
[11] K. Zhang, J. Li, G. Li, X. Shi, Z. Jin, Codeagent: enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges, arXiv:2401.07339 (2024).
[12] F. He, T. Zhu, D. Ye, B. Liu, W. Zhou, P.S. Yu, The emerged security and privacy of llm agent: a survey with case studies, arXiv:2407.19354 (2024).
[13] Z. Deng, Y. Guo, C. Han, W. Ma, J. Xiong, S. Wen, Y. Xiang, Ai agents under threat: a survey of key security challenges and future pathways, ACM Comput. Surv. 57 (7) (2025) 1–36.
[14] J. Luo, W. Zhang, Y. Yuan, Y. Zhao, J. Yang, Y. Gu, B. Wu, B. Chen, Z. Qiao, Q. Long, et al., Large language model agent: a survey on methodology, applications and challenges, arXiv:2503.21460 (2025).
[15] Y. Gan, Y. Yang, Z. Ma, P. He, R. Zeng, Y. Wang, Q. Li, C. Zhou, S. Li, T. Wang, et al., Navigating the risks: a survey of security, privacy, and ethics threats in llm-based agents, arXiv:2411.09523 (2024).
[16] K. Wang, G. Zhang, Z. Zhou, J. Wu, M. Yu, S. Zhao, C. Yin, J. Fu, Y. Yan, H. Luo, et al., A comprehensive survey in LLM (-Agent) full stack safety: data, training and deployment, arXiv:2504.15585 (2025).
[17] M. Yu, F. Meng, X. Zhou, S. Wang, J. Mao, L. Pang, T. Chen, K. Wang, X. Li, Y. Zhang, et al., A survey on trustworthy LLM agents: threats and countermeasures, arXiv:2503.09648 (2025).
[18] A. Chen, Y. Wu, J. Zhang, S. Yang, J.-t. Huang, K. Wang, W. Wang, S. Wang, A survey on the safety and security threats of computer-Using agents: JARVIS or ultron?, arXiv:2505.10924 (2025).
[19] D. Kong, S. Lin, Z. Xu, Z. Wang, M. Li, Y. Li, Y. Zhang, Z. Sha, Y. Li, C. Lin, et al., A survey of LLM-driven AI agent communication: protocols, security risks, and defense countermeasures, arXiv:2506.19676 (2025).
[20] M.A. Ferrag, N. Tihanyi, D. Hamouda, L. Maglaras, M. Debbah, From prompt injections to protocol exploits: threats in LLM-Powered AI agents workflows, arXiv:2506.23260 (2025).
[21] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, et al., A survey on large language model based autonomous agents, Front. Comput. Sci. 18 (6) (2024) 186345.
[22] Y. Du, Z. Ma, Y. Yang, K. Deng, X. Chen, B. Yang, Y. Xiang, M. Liu, B. Qin, CoT-ST: enhancing LLM-based speech translation with multimodal chain-of-thought, arXiv:2409.19510 (2024).
[23] M. Sanwal, Layered chain-of-thought prompting for multi-agent LLM systems: a comprehensive approach to explainable large language models, arXiv:2501.18645 (2025).
[24] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, K. Narasimhan, Tree of thoughts: deliberate problem solving with large language models, Adv. Neural Inf. Process. Syst. 36 (2023) 11809–11822.
[25] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, M. Podstawski, L. Gianinazzi, J. Gajda, T. Lehmann, H. Niewiadomski, P. Nyczyk, et al., Graph of thoughts: solving elaborate problems with large language models, in: Proceedings of the AAAI Conference on Artificial Intelligence, 38, 2024, pp. 17682–17690.
[26] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, T. Scialom, Toolformer: language models can teach themselves to use tools, Adv. Neural Inf. Process. Syst. 36 (2023) 68539–68551.
[27] J.S. Park, J. O’Brien, C.J. Cai, M.R. Morris, P. Liang, M.S. Bernstein, Generative agents: interactive simulacra of human behavior, in: Proceedings of the 36th Annual Acm Symposium on User Interface Software and Technology, 2023, pp. 1–22.
[28] R.K. Sharma, V. Gupta, D. Grossman, SPML: a dsl for defending language models against prompt attacks, arXiv:2402.11755 (2024).
[29] R. Huang, M. Li, D. Yang, J. Shi, X. Chang, Z. Ye, Y. Wu, Z. Hong, J. Huang, J. Liu, et al., AudioGPT: understanding and generating speech, music, sound, and talking head, in: Proceedings of the AAAI Conference on Artificial Intelligence, 38, 2024, pp. 23802–23804.
[30] M. Jafaripour, S. Golestan, S. Miwa, Y. Mitsuka, O. Zaiane, Adaptive iterative feedback prompting for obstacle-aware path planning via llms, in: AAAI Workshop, 2025.
[31] Z. Zhang, Q. Dai, X. Bo, C. Ma, R. Li, X. Chen, J. Zhu, Z. Dong, J.-R. Wen, A survey on the memory mechanism of large language model-based agents, ACM Trans. Inf. Syst. 43 (6) (2025) 1–47.
[32] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, Adv. Neural Inf. Process. Syst. 33 (2020) 9459–9474.
[33] J. Chen, S.L. Cong, Agentguard: repurposing agentic orchestrator for safety evaluation of tool orchestration, arXiv:2502.09809 (2025).
[34] M. Renze, E. Guven, Self-reflection in LLM agents: effects on problem-solving performance, arXiv:2405.06682 (2024).
[35] T. Rahmatullaev, P. Druzhinina, M. Mikhalchuk, A. Kuznetsov, A. Razzhigaev, Universal adversarial attack on aligned multimodal LLMs, arXiv:2502.07987 (2025).
[36] Z. Dong, Z. Zhou, C. Yang, J. Shao, Y. Qiao, Attacks, defenses and evaluations for llm conversation safety: a survey, arXiv:2402.09283 (2024).
[37] J. Ji, D. Hong, B. Zhang, B. Chen, J. Dai, B. Zheng, T. Qiu, B. Li, Y. Yang, Pku-saferlhf: towards multi-level safety alignment for llms with human preference, arXiv:2406.15513 (2024).
[38] A. Liu, Y. Zhou, X. Liu, T. Zhang, S. Liang, J. Wang, Y. Pu, T. Li, J. Zhang, W. Zhou, et al., Compromising embodied agents with contextual backdoor attacks, arXiv:2408.02882 (2024).
[39] K. Domico, J.-C.N. Ferrand, R. Sheatsley, E. Pauley, J. Hanna, P. McDaniel, Adversarial agents: black-box evasion attacks with reinforcement learning, arXiv:2503.01734 (2025).
[40] Q. Xu, Z. Tian, H. Wu, Z. Huang, Y. Song, F. Liu, D. Li, Learn to disguise: avoid refusal responses in LLM’s defense via a multi-agent attacker-Disguiser game, arXiv:2404.02532 (2024).
[41] A. Bilal, D. Ebert, B. Lin, Llms for explainable ai: a comprehensive survey, arXiv:2504.00125 (2025).

[42] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, et al., Extracting training data from large language models, in: 30th USENIX Security Symposium (USENIX Security 21), 2021, pp. 2633–2650.
[43] Y. Bai, G. Pei, J. Gu, Y. Yang, X. Ma, Special characters attack: toward scalable training data extraction from large language models, arXiv:2405.05990 (2024).
[44] M. Juuti, S. Szyller, S. Marchal, N. Asokan, PRADA: Protecting against DNN model stealing attacks, in: 2019 IEEE European Symposium on Security and Privacy (EuroS&P), IEEE, 2019, pp. 512–527.
[45] S. Kariyappa, A. Prakash, M.K. Qureshi, Maze: data-free model stealing attack using zeroth-order gradient estimation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 13814–13823.
[46] T. Green, M. Gubri, H. Puerto, S. Yun, S.J. Oh, Leaky thoughts: large reasoning models are not private thinkers, arXiv:2506.15674 (2025).
[47] N. Mireshghallah, H. Kim, X. Zhou, Y. Tsvetkov, M. Sap, R. Shokri, Y. Choi, Can llms keep a secret? testing privacy implications of language models via contextual integrity theory, arXiv:2310.17884 (2023).
[48] B. Wang, W. He, S. Zeng, Z. Xiang, Y. Xing, J. Tang, P. He, Unveiling privacy risks in llm agent memory, arXiv:2502.13172 (2025).
[49] S. Zeng, J. Zhang, P. He, Y. Xing, Y. Liu, H. Xu, J. Ren, S. Wang, D. Yin, Y. Chang, et al., The good and the bad: exploring privacy issues in retrieval-augmented generation (rag), arXiv:2402.16893 (2024).
[50] Y. Huang, S. Gupta, Z. Zhong, K. Li, D. Chen, Privacy implications of retrieval-based language models, arXiv:2305.14888 (2023).
[51] W. Jiang, H. Li, G. Xu, T. Zhang, R. Lu, A comprehensive defense framework against model extraction attacks, IEEE Trans. Dependable Secure Comput. 21 (2) (2023) 685–700.
[52] B. Hui, H. Yuan, N. Gong, P. Burlina, Y. Cao, Pleak: prompt leaking attacks against large language model applications, in: Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, 2024, pp. 3600–3614.
[53] Y. Yao, J. Duan, K. Xu, Y. Cai, Z. Sun, Y. Zhang, A survey on large language model (llm) security and privacy: the good, the bad, and the ugly, High-Confidence Computing (2024) 100211.
[54] N. Carlini, J. Hayes, M. Nasr, M. Jagielski, V. Sehwag, F. Tramer, B. Balle, D. Ippolito, E. Wallace, Extracting training data from diffusion models, in: 32Nd USENIX Security Symposium (USENIX Security 23), 2023, pp. 5253–5270.
[55] B.C. Das, M.H. Amini, Y. Wu, Security and privacy challenges of large language models: a survey, ACM Comput. Surv. 57 (6) (2025) 1–39.
[56] L. Wang, J. Wang, J. Wan, L. Long, Z. Yang, Z. Qin, Property existence inference against generative models, in: 33rd USENIX Security Symposium (USENIX Security 24), 2024, pp. 2423–2440.
[57] R. Staab, M. Vero, M. Balunović, M. Vechev, Beyond memorization: violating privacy via inference with large language models, arXiv:2310.07298 (2023).
[58] B. Yan, K. Li, M. Xu, Y. Dong, Y. Zhang, Z. Ren, X. Cheng, On protecting the data privacy of large language models (llms): a survey, arXiv:2403.05156 (2024).
[59] E. Bagdasarian, R. Yi, S. Ghalebikesabi, P. Kairouz, M. Gruteser, S. Oh, B. Balle, D. Ramage, Airgapagent: protecting privacy-conscious conversational agents, in: Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, 2024, pp. 3868–3882.
[60] X. Fu, S. Li, Z. Wang, Y. Liu, R.K. Gupta, T. Berg-Kirkpatrick, E. Fernandes, Imprompter: tricking LLM agents into improper tool use, arXiv:2410.14923 (2024).
[61] Z. Jiang, M. Li, G. Yang, J. Wang, Y. Huang, Z. Chang, Q. Wang, Mimicking the familiar: dynamic command generation for information theft attacks in LLM tool-learning system, arXiv:2502.11358 (2025).
[62] Z. Liao, L. Mo, C. Xu, M. Kang, J. Zhang, C. Xiao, Y. Tian, B. Li, H. Sun, Eia: environmental injection attack on generalist web agents for privacy leakage, arXiv:2409.11295 (2024).
[63] P. He, H. Xu, Y. Xing, H. Liu, M. Yamada, J. Tang, Data poisoning for in-context learning, arXiv:2402.02160 (2024).
[64] M. Russinovich, A. Salem, R. Eldan, Great, now write an article about that: the crescendo multi-turn llm jailbreak attack, arXiv:2404.01833 (2024).
[65] X. Cao, N.Z. Gong, Mitigating evasion attacks to deep neural networks via region-based classification, in: Proceedings of the 33Rd Annual Computer Security Applications Conference, 2017, pp. 278–287.
[66] H. Aghakhani, W. Dai, A. Manoel, X. Fernandes, A. Kharkar, C. Kruegel, G. Vigna, D. Evans, B. Zorn, R. Sim, Trojanpuzzle: covertly poisoning code-suggestion models, in: 2024 IEEE Symposium on Security and Privacy (SP), IEEE, 2024, pp. 1122–1140.
[67] J. Wang, J. Wu, M. Chen, Y. Vorobeychik, C. Xiao, RLHFPoison: reward poisoning attack for reinforcement learning with human feedback in large language models, arXiv:2311.09641 (2023).
[68] Z. Xiang, F. Jiang, Z. Xiong, B. Ramasubramanian, R. Poovendran, B. Li, Badchain: backdoor chain-of-thought prompting for large language models, arXiv:2401.12242 (2024).
[69] Z. Chen, Z. Xiang, C. Xiao, D. Song, B. Li, Agentpoison: red-teaming llm agents via poisoning memory or knowledge bases, Adv. Neural Inf. Process. Syst. 37 (2024) 130185–130213.
[70] T. Cui, Y. Wang, C. Fu, Y. Xiao, S. Li, X. Deng, Y. Liu, Q. Zhang, Z. Qiu, P. Li, et al., Risk taxonomy, mitigation, and assessment benchmarks of large language model systems, arXiv:2401.05778 (2024).
[71] R. Zhang, H.-W. Li, X.-Y. Qian, W.-B. Jiang, H.-X. Chen, On large language models safety, security, and privacy: a survey, J. Electron. Sci. Technol. (2025) 100301.
[72] H. Wang, R. Zhong, J. Wen, J. Steinhardt, Adaptivebackdoor: backdoored language model agents that detect human overseers, in: ICML 2024 Next Generation of AI Safety Workshop, 2024.
[73] Y. Wang, D. Xue, S. Zhang, S. Qian, Badagent: inserting and activating backdoor attacks in llm agents, arXiv:2406.03007 (2024).
[74] N. Kandpal, M. Jagielski, F. Tramèr, N. Carlini, Backdoor attacks for in-context learning with language models, arXiv:2307.14692 (2023).
[75] S. Zhao, M. Jia, L.A. Tuan, F. Pan, J. Wen, Universal vulnerabilities in large language models: backdoor attacks for in-context learning, arXiv:2401.05949 (2024).
[76] J. Xu, M.D. Ma, F. Wang, C. Xiao, M. Chen, Instructions as backdoors: backdoor vulnerabilities of instruction tuning for large language models, arXiv:2305.14710 (2023).
[77] W. Yang, X. Bi, Y. Lin, S. Chen, J. Zhou, X. Sun, Watch out for your agents! investigating backdoor threats to llm-based agents, Adv. Neural Inf. Process. Syst. 37 (2024) 100938–100964.
[78] A. Panda, C.A. Choquette-Choo, Z. Zhang, Y. Yang, P. Mittal, Teach LLMs to phish: stealing private information from language models, arXiv:2403.00871 (2024).
[79] Y. Peng, L. Zhang, P. Lv, K. Chen, Repeatleakage: leak prompts from repeating as large language model is a good repeater, in: Proceedings of the AAAI Conference on Artificial Intelligence, 39, 2025, pp. 26335–26343.
[80] Z. Liang, Q. Ye, Y. Wang, S. Zhang, Y. Xiao, R. Li, J. Xu, H. Hu, Alignment-Aware model extraction attacks on large language models, arXiv:2409.02718 (2024).
[81] W. Fu, H. Wang, C. Gao, G. Liu, Y. Li, T. Jiang, Membership inference attacks against fine-tuned large language models via self-prompt calibration, in: The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024.
[82] X. Shen, Y. Qu, M. Backes, Y. Zhang, Prompt stealing attacks against {Text-to-Image} generation models, in: 33rd USENIX Security Symposium (USENIX Security 24), 2024, pp. 5823–5840.
[83] A. Zou, Z. Wang, N. Carlini, M. Nasr, J.Z. Kolter, M. Fredrikson, Universal and transferable adversarial attacks on aligned language models, arXiv:2307.15043 (2023).
[84] R.O. Ogundokun, R. Maskeliunas, S. Misra, R. Damaševičius, Improved CNN based on batch normalization and adam optimizer, in: International Conference on Computational Science and Its Applications, Springer, 2022, pp. 593–604.
[85] J. Shi, Z. Yuan, Y. Liu, Y. Huang, P. Zhou, L. Sun, N.Z. 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, 2024, pp. 660–674.
[86] I. Nakash, G. Kour, G. Uziel, A. Anaby-Tavor, Breaking react agents: foot-in-the-Door attack will get you in, arXiv:2410.16950 (2024).
[87] G. Deng, Y. Liu, Y. Li, K. Wang, Y. Zhang, Z. Li, H. Wang, T. Zhang, Y. Liu, Masterkey: automated jailbreak across multiple large language model chatbots, arXiv:2307.08715 (2023).
[88] A. Wan, E. Wallace, S. Shen, D. Klein, Poisoning language models during instruction tuning, in: International Conference on Machine Learning, PMLR, 2023, pp. 35413–35425.
[89] W. Zou, R. Geng, B. Wang, J. Jia, Poisonedrag: knowledge corruption attacks to retrieval-augmented generation of large language models, arXiv:2402.07867 (2024).
[90] X. Gu, X. Zheng, T. Pang, C. Du, Q. Liu, Y. Wang, J. Jiang, M. Lin, Agent smith: a single image can jailbreak one million multimodal llm agents exponentially fast, arXiv:2402.08567 (2024).
[91] Z. Tan, C. Zhao, R. Moraffah, Y. Li, Y. Kong, T. Chen, H. Liu, The wolf within: covert injection of malice into mllm societies via an mllm operative, arXiv:2402.14859 (2024).
[92] W. Yu, K. Hu, T. Pang, C. Du, M. Lin, M. Fredrikson, Infecting LLM agents via generalizable adversarial attack, in: Red Teaming GenAI: What Can We Learn from Adversaries?,
[93] A. Amayuelas, X. Yang, A. Antoniades, W. Hua, L. Pan, W. Wang, Multiagent collaboration attack: investigating adversarial attacks in large language model collaborations via debate, arXiv:2406.14711 (2024).
[94] J. Zhang, C. Xu, B. Li, Chatscene: knowledge-enabled safety-critical scenario generation for autonomous vehicles, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 15459–15469.
[95] R. Song, M.O. Ozmen, H. Kim, A. Bianchi, Z.B. Celik, Enhancing LLM-based autonomous driving agents to mitigate perception attacks, arXiv:2409.14488 (2024).
[96] S. Nong, J. Zhu, R. Wu, J. Jin, S. Shan, X. Huang, W. Xu, Mobileflow: a multimodal llm for mobile gui agent, arXiv:2407.04346 (2024).
[97] B. Zheng, B. Gou, J. Kil, H. Sun, Y. Su, Gpt-4v (ision) is a generalist web agent, if grounded, arXiv:2401.01614 (2024).
[98] X. Xian, G. Wang, X. Bi, J. Srinivasa, A. Kundu, C. Fleming, M. Hong, J. Ding, On the vulnerability of applying retrieval-augmented generation within knowledge-intensive application domains, arXiv:2409.17275 (2024).
[99] H. Zhou, K.-H. Lee, Z. Zhan, Y. Chen, Z. Li, Z. Wang, H. Haddadi, E. Yilmaz, Trustrag: enhancing robustness and trustworthiness in rag, arXiv:2501.00879 (2025).
[100] Z. Zhang, Y. Shi, J. Zhu, W. Zhou, X. Qi, P. Zhang, H. Li, Trustworthy alignment of retrieval-augmented large language models via reinforcement learning, arXiv:2410.16843 (2024).
[101] X. Tan, H. Luan, M. Luo, X. Sun, P. Chen, J. Dai, RevPRAG: revealing poisoning attacks in retrieval-Augmented generation through LLM activation analysis, arXiv:2411.18948 (2024).
[102] C. Xiang, T. Wu, Z. Zhong, D. Wagner, D. Chen, P. Mittal, Certifiably robust rag against retrieval corruption, arXiv:2405.15556 (2024).
[103] X. Zhang, H. Xu, Z. Ba, Z. Wang, Y. Hong, J. Liu, Z. Qin, K. Ren, Privacyasst: safeguarding user privacy in tool-using large language model agents, IEEE Trans. Dependable Secure. Comput. 21 (6), (2024) 5242–5258..

[104] Q. Mao, Q. Zhang, H. Hao, Z. Han, R. Xu, W. Jiang, Q. Hu, Z. Chen, T. Zhou, B. Li, et al., Privacy-preserving federated embedding learning for localized retrieval-augmented generation, arXiv:2504.19101 (2025).
[105] Y. Liu, X. Peng, Y. Zhang, X. Ke, S. Deng, J. Cao, C. Ma, M. Fu, X. Zhang, S. Cheng, et al., DP-MemArc: differential privacy transfer learning for memory efficient language models, in: Proceedings of the AAAI Conference on Artificial Intelligence, 39, 2025, pp. 26317–26325.
[106] M. Anderson, G. Amit, A. Goldsteen, Is my data in your retrieval database? membership inference attacks against retrieval augmented generation, arXiv:2405.20446 (2024).
[107] E. Tennant, S. Hailes, M. Musolesi, Moral alignment for LLM agents, arXiv:2410.01639 (2024).
[108] F. Jia, T. Wu, X. Qin, A. Squicciarini, The task shield: enforcing task alignment to defend against indirect prompt injection in LLM agents, arXiv:2412.16682 (2024).
[109] J. Zhang, L. Yin, Y. Zhou, S. Hu, Agentalign: navigating safety alignment in the shift from informative to agentic large language models, arXiv:2505.23020 (2025).
[110] F. Aguilera-Martínez, F. Berzal, LLM Security: vulnerabilities, attacks, defenses, and countermeasures, arXiv:2505.01177 (2025).
[111] J. Cui, Y. Xu, Z. Huang, S. Zhou, J. Jiao, J. Zhang, Recent advances in attack and defense approaches of large language models, arXiv:2409.03274 (2024).
[112] D. Agarwal, A.R. Fabbri, B. Risher, P. Laban, S. Joty, C.-S. Wu, Prompt leakage effect and defense strategies for multi-turn llm interactions, arXiv:2404.16251 (2024).
[113] S. Wang, C. Liu, Z. Zheng, S. Qi, S. Chen, Q. Yang, A. Zhao, C. Wang, S. Song, G. Huang, Avalon'S game of thoughts: battle against deception through recursive contemplation, arXiv:2310.01320 (2023).
[114] H. Inan, K. Upasani, J. Chi, R. Rungta, K. Iyer, Y. Mao, M. Tontchev, Q. Hu, B. Fuller, D. Testuggine, et al., Llama guard: LLM-based input-output safeguard for human-ai conversations, arXiv:2312.06674 (2023).
[115] Z. Wang, F. Yang, L. Wang, P. Zhao, H. Wang, L. Chen, Q. Lin, K.-F. Wong, Self-guard: empower the llm to safeguard itself, arXiv:2310.15851 (2023).
[116] Y. Chen, H. Lent, J. Bjerva, Text embedding inversion security for multilingual language models, arXiv:2401.12192 (2024).
[117] J. Yu, J. Zhou, Y. Ding, L. Zhang, Y. Guo, H. Sato, Textual differential privacy for context-Aware reasoning with large language model, in: 2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC), IEEE, 2024, pp. 988–997.
[118] H. Fang, X. Zhu, I. Gurevych, Preemptive detection and correction of misaligned actions in llm agents, arXiv:2407.11843 (2024).
[119] Z. Xiang, L. Zheng, Y. Li, J. Hong, Q. Li, H. Xie, J. Zhang, Z. Xiong, C. Xie, C. Yang, et al., Guardagent: safeguard llm agents by a guard agent via knowledge-enabled reasoning, arXiv:2406.09187 (2024).
[120] P.Y. Zhong, S. Chen, R. Wang, M. McCall, B.L. Titzer, H. Miller, P.B. Gibbons, Rtbas: defending llm agents against prompt injection and privacy leakage, arXiv:2502.08966 (2025).
[121] Z. Yang, S.S. Raman, A. Shah, S. Tellex, Plug in the safety chip: enforcing constraints for llm-driven robot agents, in: 2024 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2024, pp. 14435–14442.
[122] H. Wen, Y. Li, G. Liu, S. Zhao, T. Yu, T.J.-J. Li, S. Jiang, Y. Liu, Y. Zhang, Y. Liu, Autodroid: llm-powered task automation in android, in: Proceedings of the 30th Annual International Conference on Mobile Computing and Networking, 2024, pp. 543–557.
[123] Y. Ruan, H. Dong, A. Wang, S. Pitis, Y. Zhou, J. Ba, Y. Dubois, C.J. Maddison, T. Hashimoto, Identifying the risks of lm agents with an lm-emulated sandbox, arXiv:2309.15817 (2023).
[124] X. Zhou, H. Kim, F. Brahman, L. Jiang, H. Zhu, X. Lu, F. Xu, B.Y. Lin, Y. Choi, N. Mireshghallah, et al., Haicosystem: an ecosystem for sandboxing safety risks in human-ai interactions, arXiv:2409.16427 (2024).
[125] L. Changjiang, L. Jiacheng, C. Bochuan, C. Jinghui, W. Ting, Your agent can defend itself against backdoor attacks, arXiv:2506.08336 (2025).
[126] D. Lee, M. Tiwari, Prompt infection: LLM-to-llm prompt injection within multi-agent systems, arXiv:2410.07283 (2024).
[127] W. Hua, X. Yang, M. Jin, Z. Li, W. Cheng, R. Tang, Y. Zhang, Trustagent: towards safe and trustworthy llm-based agents, arXiv:2402.01586 (2024).
[128] C. Yueh-Han, N. Joshi, Y. Chen, H. He, R. Angell, Monitoring LLM agents for sequentially contextual harm, in: ICLR 2025 Workshop on Building Trust in Language Models and Applications, 2025.
[129] W. Huang, T. Pan, Y. Ye, Graphormer-guided task planning: beyond static rules with llm safety perception, arXiv:2503.06866 (2025).
[130] W. Luo, S. Dai, X. Liu, S. Banerjee, H. Sun, M. Chen, C. Xiao, Agrail: a lifelong agent guardrail with effective and adaptive safety detection, arXiv:2502.11448 (2025).
[131] S. Abdelnabi, A. Gomaa, E. Bagdasarian, P.O. Kristensson, R. Shokri, Firewalls to secure dynamic llm agentic networks, (2025).
[132] B. Chen, G. Li, X. Lin, Z. Wang, J. Li, Blockagents: towards byzantine-robust llm-based multi-agent coordination via blockchain, in: Proceedings of the ACM Turing Award Celebration Conference-China 2024, 2024, pp. 187–192.
[133] Y. Du, S. Li, A. Torralba, J.B. Tenenbaum, I. Mordatch, Improving factuality and reasoning in language models through multiagent debate, in: Forty-first International Conference on Machine Learning, 2023.
[134] M. Zhuge, W. Wang, L. Kirsch, F. Faccio, D. Khizbullin, J. Schmidhuber, Gptswarm: language agents as optimizable graphs, in: Forty-first International Conference on Machine Learning, 2024.
[135] S. Wang, G. Zhang, M. Yu, G. Wan, F. Meng, C. Guo, K. Wang, Y. Wang, G-Safeguard: a topology-guided security lens and treatment on llm-based multi-agent systems, arXiv:2502.11127 (2025).
[136] T. Gu, K. Liu, B. Dolan-Gavitt, S. Garg, Badnets: evaluating backdooring attacks on deep neural networks, IEEE Access 7 (2019) 47230–47244.
[137] J. Mao, F. Meng, Y. Duan, M. Yu, X. Jia, J. Fang, Y. Liang, K. Wang, Q. Wen, Agentsafe: safeguarding large language model-based multi-agent systems via hierarchical data management, arXiv:2503.04392 (2025).
[138] P.S. Park, S. Goldstein, A. O'Gara, M. Chen, D. Hendrycks, AI Deception: a survey of examples, risks, and potential solutions, Patterns 5 (5) (2024).
[139] X. Wang, J. Peng, K. Xu, H. Yao, T. Chen, Reinforcement learning-driven llm agent for automated attacks on llms, in: Proceedings of the Fifth Workshop on Privacy in Natural Language Processing, 2024, pp. 170–177.
[140] H. Xu, W. Zhang, Z. Wang, F. Xiao, R. Zheng, Y. Feng, Z. Ba, K. Ren, Redagent: red teaming large language models with context-aware autonomous language agent, arXiv:2407.16667 (2024).
[141] Y. Dong, Z. Li, X. Meng, N. Yu, S. Guo, Jailbreaking text-to-image models with llm-based agents, arXiv:2408.00523 (2024).
[142] L.-b. Ning, S. Wang, W. Fan, Q. Li, X. Xu, H. Chen, F. Huang, Cheatagent: attacking llm-empowered recommender systems via llm agent, in: Proceedings of the 30Th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2024, pp. 2284–2295.
[143] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, M. Fritz, 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, 2023, pp. 79–90.
[144] K. Afane, W. Wei, Y. Mao, J. Farooq, J. Chen, Next-generation phishing: how LLM agents empower cyber attackers, in: 2024 IEEE International Conference on Big Data (BigData), IEEE, 2024, pp. 2558–2567.
[145] Y. Tian, X. Yang, J. Zhang, Y. Dong, H. Su, Evil geniuses: delving into the safety of llm-based agents, arXiv:2311.11855 (2023).
[146] Z. Zhang, Y. Zhang, L. Li, H. Gao, L. Wang, H. Lu, F. Zhao, Y. Qiao, J. Shao, Psysafe: a comprehensive framework for psychological-based attack, defense, and evaluation of multi-agent system safety, arXiv:2401.11880 (2024).
[147] C. Jiang, X. Pan, G. Hong, C. Bao, M. Yang, Rag-thief: scalable extraction of private data from retrieval-augmented generation applications with agent-based attacks, arXiv:2411.14110 (2024).
[148] Y. Nie, Z. Wang, Y. Yu, X. Wu, X. Zhao, W. Guo, D. Song, Privagent: agentic-based red-teaming for llm privacy leakage, arXiv:2412.05734 (2024).
[149] Y. Du, Z. Li, B. Ding, Y. Li, H. Xiao, J. Zhou, N. Li, Automated profile inference with language model agents, arXiv:2505.12402 (2025).
[150] H. Triedman, R. Jha, V. Shmatikov, Multi-agent systems execute arbitrary malicious code, arXiv:2503.12188 (2025).
[151] Z. Chen, Z. Zhao, W. Qu, Z. Wen, Z. Han, Z. Zhu, J. Zhang, H. Yao, Pandora: detailed llm jailbreaking via collaborated phishing agents with decomposed reasoning, in: ICLR 2024 Workshop on Secure and Trustworthy Large Language Models, 2024.
[152] F. Wang, R. Duan, P. Xiao, X. Jia, S. Zhao, C. Wei, Y. Chen, C. Wang, J. Tao, H. Su, et al., Mrj-agent: an effective jailbreak agent for multi-round dialogue, arXiv:2411.03814 (2024).
[153] R. Fang, R. Bindu, A. Gupta, Q. Zhan, D. Kang, Llm agents can autonomously hack websites, arXiv:2402.06664 (2024).
[154] R. Fang, R. Bindu, A. Gupta, D. Kang, LLM agents can autonomously exploit one-day vulnerabilities, arXiv:2404.08144 (2024).
[155] Y. Zhu, A. Kellermann, A. Gupta, P. Li, R. Fang, R. Bindu, D. Kang, Teams of llm agents can exploit zero-day vulnerabilities, arXiv:2406.01637 (2024).
[156] M. Gupta, C. Akiri, K. Aryal, E. Parker, L. Praharaj, From chatgpt to threatgpt: impact of generative ai in cybersecurity and privacy, IEEE Access 11 (2023) 80218–80245.
[157] S. Nakatani, Rapidpen: fully automated IP-to-shell penetration testing with LLM-based agents, arXiv:2502.16730 (2025).
[158] J. Scheurer, M. Balesni, M. Hobbhahn, Large language models can strategically deceive their users when put under pressure, arXiv:2311.07590 (2023).
[159] Y. Zeng, Y. Wu, X. Zhang, H. Wang, Q. Wu, Autodefense: multi-agent llm defense against jailbreak attacks, arXiv:2403.04783 (2024).
[160] G. Lin, Q. Zhao, Large language model sentinel: Advancing adversarial robustness by llm agent, (2024) arXiv–2405.
[161] Z. Ni, H. Wang, H. Wang, Shieldlearner: a new paradigm for jailbreak attack defense in llms, arXiv:2502.13162 (2025).
[162] J. Loevenich, E. Adler, T. Huerten, R.R.F. Lopes, Design and evaluation of an autonomous cyber defence agent using DRL and an augmented LLM, Comput. Netw. 262 (2025) 111162.
[163] K. Alrashedy, A. Aljasser, P. Tambwekar, M. Gombolay, Can llms patch security issues?, arXiv:2312.00024 (2023).
[164] U. Kulsum, H. Zhu, B. Xu, M. d'Amorim, A case study of llm for automated vulnerability repair: assessing impact of reasoning and patch validation feedback, in: Proceedings of the 1st ACM International Conference on AI-Powered Software, 2024, pp. 103–111.
[165] N. Basheer, S. Islam, S. Papastergiou, H. Mouratidis, N. Papagiannopoulos, Vulnerability patch prediction using LLM based bert model with trustworthy AI practice for cyber security enhancement, in: IFIP International Conference on Artificial Intelligence Applications and Innovations, Springer, 2025, pp. 431–445.
[166] Y. Li, Z. Xiang, N.D. Bastian, D. Song, B. Li, Ids-agent: an llm agent for explainable intrusion detection in iot networks (2024).
[167] C. Song, L. Ma, J. Zheng, J. Liao, H. Kuang, L. Yang, Audit-LLM: multi-agent collaboration for log-based insider threat detection, arXiv:2408.08902 (2024).
[168] H. Kong, D. Hu, J. Ge, L. Li, T. Li, B. Wu, Vulnbot: autonomous penetration testing for a multi-agent collaborative framework, arXiv:2501.13411 (2025).
[169] S.G. Bianou, R.G. Batogna, Pentest-ai, an llm-powered multi-agents framework for penetration testing automation leveraging mitre attack, in: 2024 IEEE In-

ternational Conference on Cyber Security and Resilience (CSR), IEEE, 2024, pp. 763–770.
[170] I. Alshehri, A. Alshehri, A. Almalki, M. Bamardouf, A. Akbar, Breachseek: a multi-agent automated penetration tester, arXiv:2409.03789 (2024).
[171] X. Shen, L. Wang, Z. Li, Y. Chen, W. Zhao, D. Sun, J. Wang, W. Ruan, Pentestagent: incorporating llm agents to automated penetration testing, arXiv:2411.05185 (2024).
[172] J. Henke, AutoPentest: enhancing vulnerability management with autonomous LLM agents, arXiv:2505.10321 (2025).
[173] K. Shi, J. Lu, Z. Fang, G. Zhang, Unsupervised domain adaptation enhanced by fuzzy prompt learning, IEEE Trans. Fuzzy Syst. 32 (7) (2024) 4038–4048.
[174] H. Wang, R. Zhang, J. Wang, M. Li, Y. Huang, D. Wang, Q. Wang, From allies to adversaries: manipulating LLM tool-Calling through adversarial injection, arXiv:2412.10198 (2024).
[175] R. Rafailov, A. Sharma, E. Mitchell, C.D. Manning, S. Ermon, C. Finn, Direct preference optimization: your language model is secretly a reward model, Adv. Neural Inf. Process. Syst. 36 (2023) 53728–53741.
[176] P.I. Frazier, A tutorial on bayesian optimization, arXiv:1807.02811 (2018).
[177] A. Geiger, D. Ibeling, A. Zur, M. Chaudhary, S. Chauhan, J. Huang, A. Arora, Z. Wu, N. Goodman, C. Potts, et al., Causal abstraction: a theoretical foundation for mechanistic interpretability, J. Mach. Learn. Res. 26 (83) (2025) 1–64.
[178] Y. Zhang, M. Li, W. Han, Y. Yao, Z. Cen, D. Zhao, Safety is not only about refusal: reasoning-Enhanced fine-tuning for interpretable LLM safety, arXiv:2503.05021 (2025).
[179] Y. Chen, H. Li, Z. Zheng, Y. Song, D. Wu, B. Hooi, Defense against prompt injection attack by leveraging attack techniques, arXiv:2411.00459 (2024).
[180] T. Sato, R. Suzuki, Y. Hayakawa, K. Ikeda, O. Sako, R. Nagata, R. Yoshida, Q.A. Chen, K. Yoshioka, On the realism of lidar spoofing attacks against autonomous driving vehicle at high speed and long distance, in: Proceedings of the Network and Distributed System Security Symposium (NDSS), 2025.
[181] J. Wang, A. Pun, J. Tu, S. Manivasagam, A. Sadat, S. Casas, M. Ren, R. Urtasun, Advsim: generating safety-critical scenarios for self-driving vehicles, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 9909–9918.
[182] X. Han, G. Xu, Y. Zhou, X. Yang, J. Li, T. Zhang, Physical backdoor attacks to lane detection systems in autonomous driving, in: Proceedings of the 30th ACM International Conference on Multimedia, 2022, pp. 2957–2968.
[183] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, A. Madry, Robustness may be at odds with accuracy, arXiv:1805.12152 (2018).
[184] W. Zhang, X. Kong, C. Dewitt, T. Braunl, J.B. Hong, A study on prompt injection attack against llm-integrated mobile robotic systems, in: 2024 IEEE 35th International Symposium on Software Reliability Engineering Workshops (ISSREW), IEEE, 2024, pp. 361–368.
[185] K. Hines, G. Lopez, M. Hall, F. Zarfati, Y. Zunger, E. Kiciman, Defending against indirect prompt injection attacks with spotlighting, arXiv:2403.14720 (2024).