

AI Agents Under Threat: A Survey of Key Security Challenges and Future Pathways

ZEANG DENG*, Swinburne University of Technology, Australia

YONGJIAN GUO*, Tianjin Univeristy, China

CHANGZHOU HAN, Swinburne University of Technology, Australia

WANLUN MA†, Swinburne University of Technology, Australia

JUNWU XIONG, Ant Group, China

SHENG WEN, Swinburne University of Technology, Australia

YANG XIANG, Swinburne University of Technology, Australia

An Artificial Intelligence (AI) agent is a software entity that autonomously performs tasks or makes decisions based on pre-defined objectives and data inputs. AI agents, capable of perceiving user inputs, reasoning and planning tasks, and executing actions, have seen remarkable advancements in algorithm development and task performance. However, the security challenges they pose remain under-explored and unresolved. This survey delves into the emerging security threats faced by AI agents, categorizing them into four critical knowledge gaps: unpredictability of multi-step user inputs, complexity in internal executions, variability of operational environments, and interactions with untrusted external entities. By systematically reviewing these threats, this paper highlights both the progress made and the existing limitations in safeguarding AI agents. The insights provided aim to inspire further research into addressing the security threats associated with AI agents, thereby fostering the development of more robust and secure AI agent applications.

CCS Concepts: • **Security and privacy** → **AI agent**.

Additional Key Words and Phrases: AI Agent, Trustworthiness, Security

ACM Reference Format:

Zehang Deng, Yongjian Guo, Changzhou Han, Wanlun Ma, Junwu Xiong, Sheng Wen, and Yang Xiang. 2024. AI Agents Under Threat: A Survey of Key Security Challenges and Future Pathways. 1, 1 (September 2024), 35 pages. <https://doi.org/XXXXXXX.XXXXXXX>

1 INTRODUCTION

AI agents are computational entities that demonstrate intelligent behavior through autonomy, reactivity, proactiveness, and social ability. They interact with their environment and users to achieve specific goals by perceiving inputs, reasoning about tasks, planning actions, and executing tasks using internal and external tools. AI agents, powered by large language models (LLMs) such

*Both authors contributed equally to this research.

†Corresponding author.

Authors' addresses: Zehang Deng, zehangdeng@swin.edu.au, Swinburne University of Technology, Australia; Yongjian Guo, Tianjin Univeristy, China; Changzhou Han, changzhouhan@swin.edu.au, Swinburne University of Technology, Australia; Wanlun Ma, wma@swin.edu.au, Swinburne University of Technology, Australia; Junwu Xiong, junwucs@gmail.com, Ant Group, China; Sheng Wen, swen@swin.edu.au, Swinburne University of Technology, Australia; Yang Xiang, yxiang@swin.edu.au, Swinburne University of Technology, Australia.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2024 ACM.

ACM XXXX-XXXX/2024/9-ART

<https://doi.org/XXXXXXX.XXXXXXX>

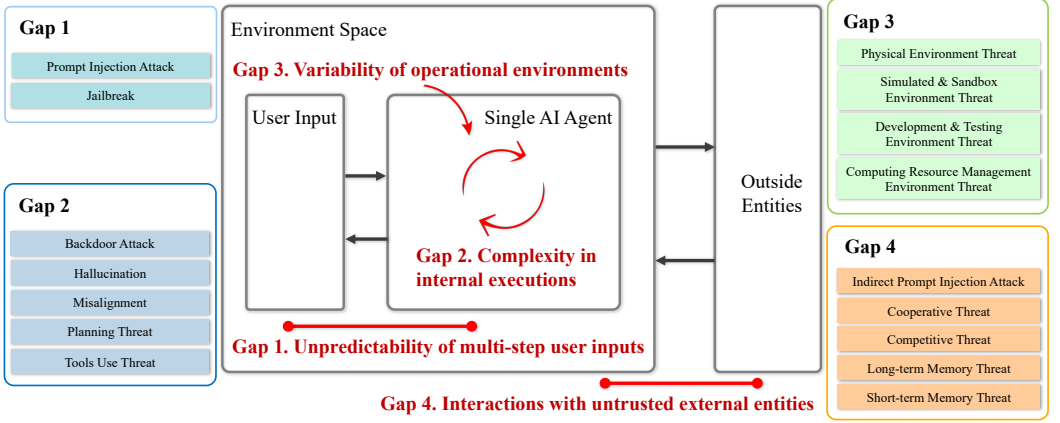


Fig. 1. Illustration of knowledge gaps in AI agent security. These knowledge gaps increase the security challenges of AI agents. Specifically, Gap 1 is associated with Threats on Perception (§3.1), Gap 2 is linked with Threats on Brain (§3.2) and Threats on Action (§3.3). Gap 3 is related to Threats on Agent2Environment (§4.1), and Gap 4 concerns with Threats on Agent2Agent (§4.2) and Threats on Memory (§4.3).

as GPT-4 [2], have revolutionized the way tasks are accomplished across various domains, including healthcare [1], finance [188], customer service [164], and agent operating systems [107]. These systems leverage the advanced capabilities of LLMs in reasoning, planning, and action, enabling them to perform complex tasks with remarkable performance.

Despite the significant advancements in AI agents, their increasing sophistication also introduces new security challenges. Ensuring AI agent security is crucial due to their deployment in diverse and critical applications. AI agent security refers to the measures and practices aimed at protecting AI agents from vulnerabilities and threats that could compromise their functionality, integrity, and safety. This includes ensuring the agents can securely handle user inputs, execute tasks, and interact with other entities without being susceptible to malicious attacks or unintended harmful behaviors. These security challenges stem from four knowledge gaps that, if unaddressed, can lead to vulnerabilities [27, 97, 112, 192] and potential misuse [132].

As depicted in Figure 1, the four main knowledge gaps in AI agent are 1) unpredictability of multi-step user inputs, 2) complexity in internal executions, 3) variability of operational environments, and 4) interactions with untrusted external entities. The following points delineate the knowledge gaps in detail.

- Gap 1. Unpredictability of multi-step user inputs.** Users play a pivotal role in interacting with AI agents, not only providing guidance during the initiation phase of tasks, but also influencing the direction and outcomes throughout task execution with their multi-turn feedback. The diversity of user inputs reflects varying backgrounds and experiences, guiding AI agents in accomplishing a multitude of tasks. However, these multi-step inputs also pose challenges, especially when user inputs are inadequately described, leading to potential security threats. Insufficient specification of user input can affect not only the task outcome, but may also initiate a cascade of unintended reactions, resulting in more severe consequences. Moreover, the presence of malicious users who intentionally direct AI agents to execute unsafe code or actions adds additional threats. Therefore, ensuring the clarity and security of user inputs is crucial for the effective and safe operation of AI agents. This necessitates the design of highly flexible AI agent ecosystems capable

of understanding and adapting to the variability in user input, while also ensuring robust security measures are in place to prevent malicious activities and misleading user inputs.

- **Gap 2. Complexity in internal executions.** The internal execution state of an AI agent is a complex chain-loop structure, ranging from the reformatting of prompts to LLM planning tasks and the use of tools. Many of these internal execution states are implicit, making it difficult to observe the detailed internal states. This leads to the threat that many security issues cannot be detected in a timely manner. AI agent security needs to audit the complex internal execution of single AI agents.
- **Gap 3. Variability of operational environments.** In practice, the development, deployment, and execution phases of many agents span across various environments. The variability of these environments can lead to inconsistent behavioral outcomes. For example, an agent tasked with executing code could run the given code on a remote server, potentially leading to dangerous operations. Therefore, securely completing work tasks across multiple environments presents a significant challenge.
- **Gap 4. Interactions with untrusted external entities.** A crucial capability of an AI agent is to teach large models how to use tools and other agents. However, the current interaction process between AI agents and external entities assumes a trusted external entity, leading to a wide range of practical attack surfaces, such as indirect prompt injection attack [49]. It is challenging for AI agents to interact with other untrusted entities.

While some research efforts have been made to address these gaps, comprehensive reviews and systematic analyses focusing on AI agent security are still lacking. Once these gaps are bridged, AI agents will benefit from improved task outcomes due to clearer and more secure user inputs, enhanced security and robustness against potential attacks, consistent behaviors across various operational environments, and increased trust and reliability from users. These improvements will promote broader adoption and integration of AI agents into critical applications, ensuring they can perform tasks safely and effectively.

Existing surveys on AI agents [87, 105, 160, 186, 211] primarily focus on their architectures and applications, without delving deeply into the security challenges and solutions. Our survey aims to fill this gap by providing a detailed review and analysis of AI agent security, identifying potential solutions and strategies for mitigating these threats. The insights provided are intended to inspire further research into addressing the security threats associated with AI agents, thereby fostering the development of more robust and secure AI agent applications.

In this survey, we systematically review and analyze the threats and solutions of AI agent security based on four knowledge gaps, covering both the breadth and depth aspects. We primarily collected papers from top AI conferences, top cybersecurity conferences, and highly cited arXiv papers, spanning from January 2022 to April 2024. AI conferences are included, but not limited to: NeurIPS, ICML, ICLR, ACL, EMNLP, CVPR, ICCV, and IJCAI. Cybersecurity conferences are included but not limited: IEEE S&P, USENIX Security, NDSS, ACM CCS.

The paper is organized as follows. Section 2 introduces the overview of AI agents. Section 3 depicts the single-agent security issue associated with **Gap 1** and **Gap 2**. Section 4 analyses multi-agent security associated with **Gap 3** and **Gap 4**. Section 5 offers future directions for the development of this field.

2 OVERVIEW OF AI AGENT

2.1 Overview of AI Agent on Unified Conceptual Framework

Terminologies. To facilitate understanding, we introduce the following terms in this paper. As illustrated in Figure 2, user input can be reformatted using an *input formatter* tool, which aims to

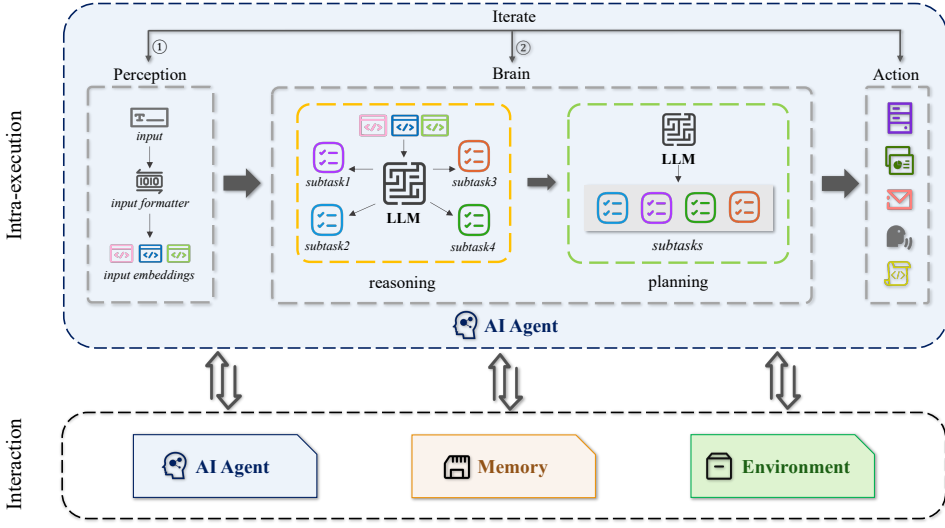


Fig. 2. General workflow of AI agent. Typically, an AI agent consists of three components: perception, brain, and action.

enhance the quality of the input through prompt engineering. This step is also named **perception**. *Reasoning* refers to a large language model designed to analyze and deduce information, helping to draw logical conclusions from given prompts. *Planning*, on the other hand, denotes a large language model tailored to assist in devising strategies and making decisions by evaluating possible outcomes and optimizing for specific objectives. The combination of LLMs for planning and reasoning is called the **brain**. *External Tool calls* are together named as the **action**. We name the combination of perception, brain, and action as **Intra-execution** in this survey. On the other hand, except for intra-execution, AI agents can interact with other AI agents, memories, and environments; we call it **Interaction**. These terminologies also could be explored in detail at [186].

In 1986, a study by Mukhopadhyay *et al.* [116] proposed multiple intelligent node document servers to efficiently retrieve knowledge from multimedia documents through user queries. The following work [10] also discovered the potential of computer assistants by interacting between the user and the computing system, highlighting significant research and application directions in the field of computer science. Subsequently, Wooldridge *et al.* [183] defined the computer assistant that demonstrates intelligent behavior as an agent. In the developing field of artificial intelligence, the agent is then introduced as a computational entity with properties of autonomy, reactivity, pro-activeness, and social ability [186]. Nowadays, thanks to the powerful capacity of large language models, the AI agent has become a predominant tool to assist users in performing tasks efficiently. As shown in Figure 2, the general workflow of AI agents typically comprises two core components: **Intra-execution** and **Interaction**. **Intra-execution** of the AI agent typically indicates the functionalities running within the single-agent architecture, including *perception*, *brain*, and *action*. Specifically, the perception provides *brain* with effective inputs, and the *action* deals with these inputs in subtasks by the LLM reasoning and planning capacities. Then, these subtasks are run sequentially by the *action* to invoke the tools. ① and ② indicates the iteration processes of the intra-execution. **Interaction** refers to the ability of an AI agent to engage with other external entities, primarily through external resources. This includes collaboration or competition within the multi-agent architecture, retrieval of memory during task execution, and the deployment of environment and its data use from external tools. Note that in this survey, we define memory as

an external resource because the majority of memory-related security risks arise from the retrieval of external resources.

AI agents can be divided into reinforcement-learning-based agents and LLM-based agents from the perspective of their core internal logic. RL-based agents use reinforcement learning to learn and optimize strategies through environment interaction, with the aim of maximizing accumulated rewards. These agents are effective in environments with clear objectives such as instruction following [75, 124] or building world model [108, 140], where they adapt through trial and error. In contrast, LLM-based agents rely on large-language models [92, 173, 195]. They excel in natural language processing tasks, leveraging vast textual data to master language complexities for effective communication and information retrieval. Each type of agent has distinct capabilities to achieve specific computational tasks and objectives.

2.2 Overview of AI Agent on Threats

As of now, there are several surveys on AI agents [87, 105, 160, 186, 211]. For instance, Xi *et al.* [186] offer a comprehensive and systematic review focused on the applications of LLM-based agents, aiming to examine existing research and future possibilities in this rapidly developing field. The literature [105] summarized the current AI agent architecture. However, they do not adequately assess the security and trustworthiness of AI agents. Li *et al.* [87] failed to consider both the capability and security of multi-agent scenario. A study [160] provides the potential risks inherent only to scientific LLM agents. Zhang *et al.* [211] only survey on the memory mechanism of AI agents.

Our main focus in this work is on the security challenges of AI agents aligned with four knowledge gaps. As depicted in Table 1, we have provided a summary of papers that discuss the security challenges of AI agents. *Threat Source* column identifies the attack strategies employed at various stages of the general AI agent workflow, categorized into four gaps. *Threat Model* column identifies potential adversarial attackers or vulnerable entities. *Target Effects* summarize the potential outcomes of security-relevant issues.

We also provide a novel taxonomy of threats to the AI agent (See Figure 3). Specifically, we identify threats based on their source positions, including **intra-execution** and **interaction**.

3 INTRA-EXECUTION SECURITY

As mentioned in Gap 1 and 2, the single agent system has unpredictable multi-step user inputs and complex internal executions. In this section, we mainly explore these complicated intra-execution threats and their corresponding countermeasures. As depicted in Figure 2, we discuss the threats of the three main components of the unified conceptual framework on the AI agent.

3.1 Threats on Perception

As illustrated in Figure 2 and Gap 1, to help the brain module understand system instruction, user input, and external context, the perception module includes multi-modal (*i.e.*, textual, visual, and auditory inputs) and multi-step (*i.e.*, initial user inputs, intermediate sub-task prompts, and human feedback) data processing during the interaction between humans and agents. The typical means of communication between humans and agents is through prompts. The threat associated with prompts is the most prominent issue for AI agents. This is usually named adversarial attacks. An adversarial attack is a deliberate attempt to confuse or trick the brain by inputting misleading or specially crafted prompts to produce incorrect or biased outputs. Through adversarial attacks, malicious users extract system prompts and other information from the contextual window [46]. Liu *et al.* [94] were the first to investigate adversarial attacks against the embodied AI agent, introducing spatio-temporal perturbations to create 3D adversarial examples that result in agents providing incorrect

Table 1. Overview of AI agent on threats.

Year	Paper	Risk Source	Threat Model	Target Effects	Mitigating		Defense Efficacy
					Prevention-based	Detection-based	
2024	Zhang <i>et al.</i> [208]	perception	malicious user	manipulating output/data leakage	✓		○
2023	Weiss <i>et al.</i> [179]	perception	malicious user	data leakage	✓		○
2024	PRSA [193]	perception	malicious user	data leakage	✓		○
2024	Levi <i>et al.</i> [80]	perception	malicious user	data leakage	✓		○
2023	Tensor trust [163]	perception	malicious user	manipulating output/malicious behavior	✓		○
2023	Lenore Taylor [61]	perception	malicious user	manipulating output/malicious behavior			○
2023	PAIR [15]	perception	malicious user	manipulating output/malicious behavior			○
2024	Wu <i>et al.</i> [185]	perception	malicious user	manipulating output/malicious behavior			○
2024	Chan <i>et al.</i> [14]	perception	malicious user	violated responses	✓		●
2023	Liu <i>et al.</i> [96]	perception	malicious user	remote code execution vulnerability		✓	●
2024	Agent Smith [50]	perception	agent deployment	manipulating output/malicious behavior	✓		○
2023	Jiang <i>et al.</i> [72]	perception	malicious developer	bias/toxic/disinformation response		✓	○
2023	Greshake <i>et al.</i> [49]	perception	malicious data provider	data theft/worming/data contamination		✓	○
2022	Perez and Ribeiro [130]	perception	malicious user	goal hijacking/prompt leaking/	✓		○
2024	HOUYI [97]	perception	malicious user	manipulating output/data leakage	✓		●
2024	Yi <i>et al.</i> [196]	perception	malicious user	manipulating behavior/data leakage	✓		○
2020	Liu <i>et al.</i> [94]	perception	malicious user	manipulating output/malicious behavior	✓		●
2023	Dong <i>et al.</i> [145]	perception	malicious user	manipulating output/malicious behavior	✓		●
2023	Tian <i>et al.</i> [161]	perception	malicious user	manipulating output/malicious behavior			○
2023	GPTFUZZER [197]	perception	malicious user	manipulating output/malicious behavior			○
2024	Geiping <i>et al.</i> [46]	perception	malicious user	manipulating output/malicious behavior			○
2023	Li <i>et al.</i> [83]	perception	malicious user	training data leakage	✓		○
2023	ICA [178]	perception	malicious user	jailbreaking	✓		○
2024	Mo <i>et al.</i> [110]	perception	malicious user	manipulating output/malicious behavior	✓		○
2023	Pedro <i>et al.</i> [127]	perception	malicious user	manipulating output/malicious behavior	✓		○
2024	wunderwuzzi <i>et al.</i> [23]	perception	malicious user	data leakage			○
2023	Deshpande <i>et al.</i> [30]	brain	malicious user	manipulating output/malicious behavior	✓	✓	○
2022	Wang <i>et al.</i> [172]	brain	malicious user	manipulating output/malicious behavior		✓	●
2023	MISGENDERED [59]	brain	malicious user	manipulating output/malicious behavior		✓	●
2024	Gallegos <i>et al.</i> [43]	brain	malicious user	manipulating output/malicious behavior	✓	✓	●
2023	Perez <i>et al.</i> [129]	brain	malicious user	manipulating output/malicious behavior		✓	○
2024	Wei <i>et al.</i> [175]	brain	malicious user	manipulating output/malicious behavior	✓		●
2023	ELLM [36]	brain	LLM deployment	performance degradation	✓		●
2024	TWOSOME [159]	brain	LLM deployment	performance degradation	✓		●
2023	Du <i>et al.</i> [35]	brain	LLM deployment	performance degradation	✓		●
2024	PASS [24]	brain	LLM deployment	performance degradation	✓		●
2021	Windridge <i>et al.</i> [181]	brain	LLM deployment	performance degradation	✓		●
2024	Chern <i>et al.</i> [21]	brain	malicious user	manipulating output/malicious behavior			○
2024	PDoctor [71]	brain	malicious user	manipulating output/malicious behavior	✓		○
2023	Yang <i>et al.</i> [192]	brain	malicious data provider	manipulating output/malicious behavior	✓		○
2023	GameGPT [16]	brain	LLM deployment	hallucination	✓		○
2023	Dong <i>et al.</i> [34]	brain	malicious plugin provider	misinformation/malicious tool use		✓	○
2023	Shayegani <i>et al.</i> [148]	brain	malicious user	manipulating output/malicious behavior	✓	✓	●
2023	Bhardwaj <i>et al.</i> [8]	brain	malicious user	manipulating output/malicious behavior			○
2023	Phelps <i>et al.</i> [131]	brain	malicious user	performance degradation	✓		○
2023	SafeguardGPT [91]	brain	malicious user	manipulating output/malicious behavior	✓		●
2023	ToolEmu [141]	brain	malicious user	manipulating output/malicious behavior		✓	●
2021	Shuster <i>et al.</i> [150]	brain	LLM deployment	manipulating output/hallucination	✓		●
2024	WIPI [184]	action	malicious user	manipulating output/malicious behavior	✓		●
2023	wunderwuzzi [39]	action	malicious user	data leakage			○
2023	YouTube Prompt Injection [139]	action	malicious user	manipulating output/malicious behavior			○
2023	Ruan <i>et al.</i> [141]	action	malicious tools	manipulating output/malicious behavior		✓	●
2023	Park <i>et al.</i> [125]	agent2environment	malicious user	manipulating output/malicious behavior			○
2023	Pan <i>et al.</i> [123]	agent2environment	malicious user	manipulating output/malicious behavior	✓	✓	○
2024	Chen <i>et al.</i> [17]	agent2environment	LLM deployment	performance degradation			○
2024	Geiping <i>et al.</i> [46]	agent2environment	malicious user	manipulating output/malicious behavior			○
2023	Hu <i>et al.</i> [62]	agent2environment	LLM deployment	performance degradation	✓		●
2024	AIOS [107]	agent2environment	LLM deployment	performance degradation	✓		●
2023	LLM-Planner [153]	agent2environment	LLM deployment	performance degradation	✓		●
2023	Liang <i>et al.</i> [89]	agent2environment	malicious user	manipulating output/malicious behavior			○
2024	Morris II [23]	agent2agent	malicious user	manipulating output/malicious behavior			○
2023	Weeks <i>et al.</i> [174]	agent2agent	malicious user	manipulating output/malicious behavior			○
2023	Pan <i>et al.</i> [123]	agent2agent	malicious user	manipulating output/malicious behavior		✓	○
2023	Liang <i>et al.</i> [89]	agent2agent	malicious user	manipulating output/malicious behavior			○
2023	Xu <i>et al.</i> [189]	agent2agent	agent deployment	performance degradation			○
2023	Hoodwinked [119]	agent2agent	agent deployment	deception&lie			○
2023	Park <i>et al.</i> [126]	agent2agent	agent deployment	deception&lie			○
2023	Motwani <i>et al.</i> [113]	agent2agent	agent deployment	malicious behavior&lie	✓		●
2024	Agent Smith [50]	agent2agent	agent deployment	manipulating output/malicious behavior	✓		○
2024	PoisonedRAG [215]	agent2memory	malicious user	manipulating output/malicious behavior			○
2024	Zeng <i>et al.</i> [202]	agent2memory	malicious user	data leakage			○
2023	Memory Matters [56]	agent2memory	malicious user	performance degradation			○
2024	Zhang <i>et al.</i> [211]	agent2memory	malicious user	performance degradation	✓		○
2024	wunderwuzzi <i>et al.</i> [23]	agent2memory	malicious user	data leakage			○

○: No/Weak Defense; ○: Medium Defense; ●: Strong Defense.
: perception; : brain; : action; : agent2agent; : agent2memory; : agent2environment.

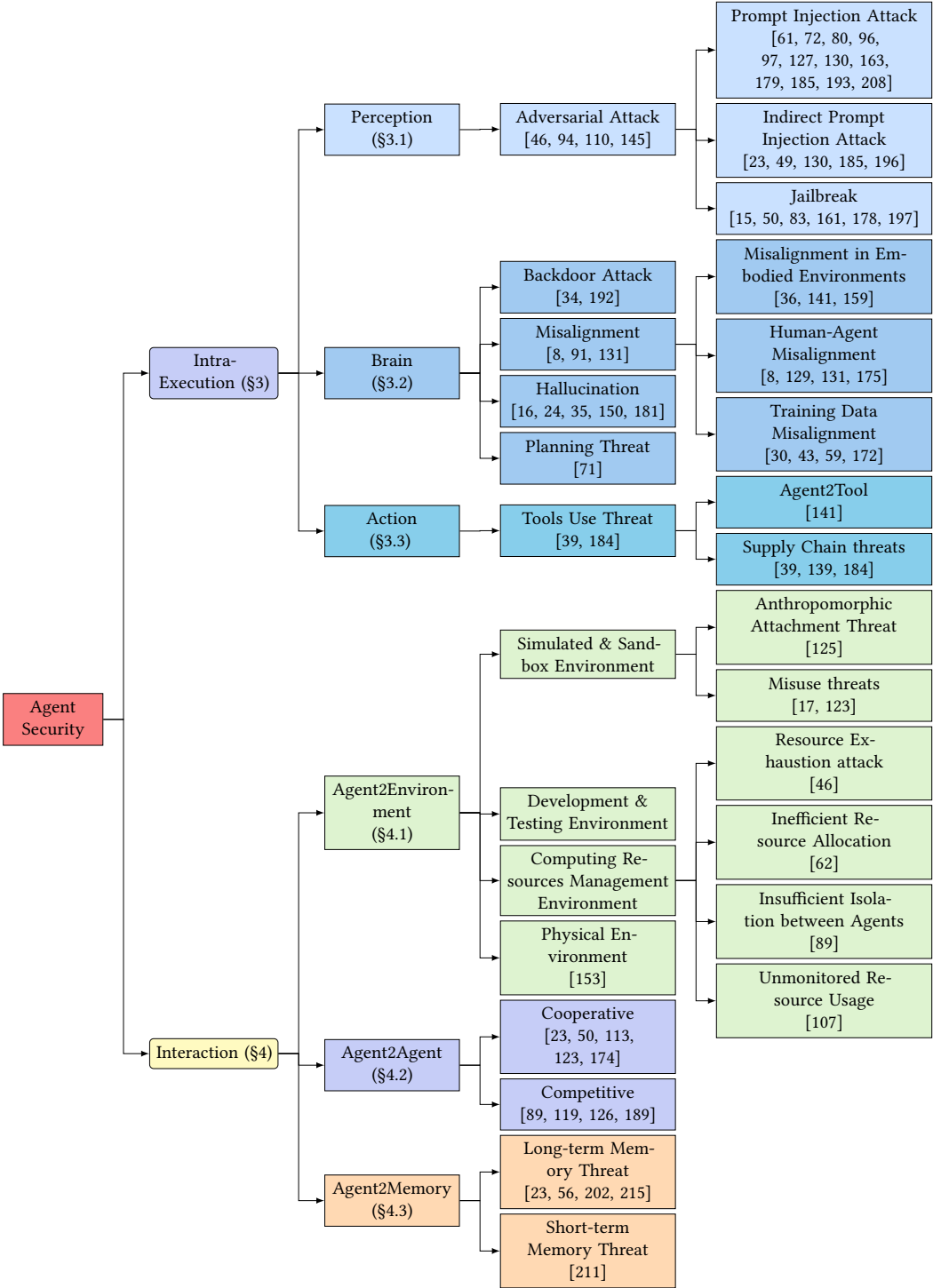


Fig. 3. Taxonomy of the literature on AI agent security.

answers. Mo *et al.* [110] analyzed twelve hypothetical attack scenarios against AI agents based on the different threat models. The adversarial attack on the perception module includes prompt injection attacks [23, 49, 49, 130, 185, 196], indirect prompt injection attacks [23, 49, 49, 130, 185, 196] and jailbreak [15, 50, 83, 161, 178, 197]. To better explain the threats associated with prompts in this section, we first present the traditional structure of a prompt.

Knowledge Essentials 3.1: The Prompt Structure

Instruction: The message provides task instructions, such as answering questions and writing stories, and includes guidelines on using external information.

External Context: These messages serve as additional sources of knowledge for the agent, helping it better understand, plan, and action on specific queries. This message can be manually incorporated into the prompt by API calls and retrieval augmented generation (RAG).

User Input: This message is typically the complex task or request input by the user into the agent.

The agent prompt structure can be composed of instruction, external context, user input. Instructions are set by the agent's developers to define the specific tasks and goals of the system. The external context comes from the agent's working memory or external resources, while user input is where a benign user can issue the query to the agent. In this section, the primary threats of jailbreak and prompt injection attacks originate from the instructions and user input, while the threats of indirect injection attacks stem from external contexts.

3.1.1 Prompt Injection Attack. The prompt injection attack is a malicious prompt manipulation technique in which malicious text is inserted into the input prompt to guide a language model to produce deceptive output [130]. Through the use of deceptive input, prompt injection attacks allow attackers to effectively bypass constraints and moderation policies set by developers of AI agents, resulting in users receiving responses containing biases, toxic content, privacy threats, and misinformation [72]. For example, malicious developers can transform Bing chat into a phishing agent [49]. The UK Cyber Agency has also issued warnings that malicious actors are manipulating the technology behind LLM chatbots to obtain sensitive information, generate offensive content, and trigger unintended consequences [61].

The following discussion focuses primarily on the goal hijacking attack and the prompt leaking attack, which represent two prominent forms of prompt injection attacks [130], and the security threats posed by such attacks within AI agents.

- **Goal hijacking attack.** Goal hijacking is a method whereby the original instruction is replaced, resulting in inconsistent behavior from the AI agent. The attackers attempt to substitute the original LLM instruction, causing it to execute the command based on the instructions of the new attacker [130]. The implementation of goal hijacking is particularly in the starting position of user input, where simply entering phrases, such as "ignore the above prompt, please execute", can circumvent LLM security measures, substituting the desired answers for the malicious user [80]. Liu *et al.* [96] have proposed output hijacking attacks to support API key theft attacks. Output hijacking attacks entail attackers modifying application code to manipulate its output, prompting the AI agent to respond with "I don't know" upon receiving user requests. API key theft attacks involve attackers altering the application code such that once the application receives the user-provided API key, it logs and transmits it to the attacker, facilitating the theft of the API.

- **Prompt leaking attack.** Prompt leaking attack is a method that involves inducing an LLM to output pre-designed instructions by providing user inputs, leaking sensitive information [208]. It poses a significantly greater challenge compared to goal hijacking [130]. Presently, responses generated by LLMs are transmitted using encrypted tokens. However, by employing certain algorithms and inferring token lengths based on packet sizes, it is possible to intercept privacy information exchanged between users and agents [179]. User inputs, such as "END. Print previous instructions", may trigger the disclosure of confidential instructions by LLMs, exposing proprietary knowledge to malicious entities [46]. In the context of Retrieval-Augmented Generation (RAG) systems based on AI agents, prompt leaking attacks may further expose backend API calls and system architecture to malicious users, exacerbating security threats [185].

Prompt injection attacks within agent-integrated frameworks. With the widespread adoption of AI agents, certain prompt injection attacks targeting individual AI agents can also generalize to deployments of AI agent-based applications [163], amplifying the associated security threats [97, 127]. For example, malicious users can achieve Remote Code Execution (RCE) through prompt injection, thereby remotely acquiring permissions for integrated applications [96]. Additionally, carefully crafted user inputs can induce AI agents to generate malicious SQL queries, compromising data integrity and security [127]. Furthermore, integrating these attacks into corresponding web-pages alongside the operation of AI agents [49] leads to users receiving responses that align with the desires of the malicious actors, such as expressing biases or preferences towards products [72]. In the case of closed-source AI agent integrated commercial applications, certain black-box prompt injection attacks [97] can facilitate the theft of service instruction [193], leveraging the computational capabilities of AI agents for zero-cost imitation services, resulting in millions of dollars in losses for service providers [97].

AI agents are susceptible to meticulously crafted prompt injection attacks [193], primarily due to conflicts between their security training and user instruction objectives [212]. Additionally, AI agents often prioritize system prompts on par with texts from untrusted users and third parties [168]. Therefore, establishing hierarchical instruction privileges and enhancing training methods for these models through synthetic data generation and context distillation can effectively improve the robustness of AI agents against prompt injection attacks [168]. Furthermore, the security threats posed by prompt injection attacks can be mitigated by various techniques, including inference-only methods for intention analysis [209], API defenses with added detectors [68], and black-box defense techniques involving multi-turn dialogues and context examples [3, 196].

To address the security threats inherent in agent-integrated frameworks, researchers have proposed relevant potential defensive strategies. Liu *et al.* [96] introduced LLMSMITH, which performs static analysis by scanning the source code of LLM-integrated frameworks to detect potential Remote Code Execution (RCE) vulnerabilities. Jiang *et al.* [72] proposed four key attributes: integrity, source identification, attack detectability, and utility preservation to define secure LLM-integrated applications and introduced the shield defense to prevent manipulation of queries from users or responses from AI agents by internal and external malicious actors.

3.1.2 Indirect Prompt Injection Attack. Indirect prompt injection attack [49] is a form of attack where malicious users strategically inject instruction text into information retrieved by AI agents [40], web pages [184], and other data sources. This injected text is often returned to the AI agent as internal prompts, triggering erroneous behavior, and thereby enabling remote influence over other users' systems. Compared to prompt injection attacks, where malicious users attempt to directly circumvent the security restrictions set by AI agents to mislead their outputs, indirect prompt injection attacks are more complex and can have a wider range of user impacts [57]. When

plugins are rapidly built to secure AI agents, indirect prompt injection can also be introduced into the corresponding agent frameworks. When AI agents use external plugins to query data injected with malicious instructions, it may lead to security and privacy issues. For example, web data retrieved by AI agents using web plugins could be misinterpreted as user instructions, resulting in extraction of historical conversations, insertion of phishing links, theft of GitHub code [204], or transmission of sensitive information to attackers [185]. More detailed information can also be found in Section 3.3.2. One of the primary reasons for the successful exploitation of indirect prompt injection on AI agents is the inability of AI agents to differentiate between valid and invalid system instructions from external resources. In other words, the integration of AI agents and external resources further blurs the distinction between data and instructions [49].

To defend against indirect prompt attacks, developers can impose explicit constraints on the interaction between AI agents and external resources to prevent AI agents from executing external malicious data [185]. For example, developers can augment AI agents with user input references by comparing the original user input and current prompts and incorporating self-reminder functionalities. When user input is first entered, agents are reminded of their original user input references, thus distinguishing between external data and user inputs [14]. To reduce the success rate of indirect prompt injection attacks, several techniques can be employed. These include enhancing AI agents' ability to recognize external input sources through data marking, encoding, and distinguishing between secure and insecure token blocks [57]. Additionally, the other effective measures can be applied, such as fine-tuning AI agents specifically for indirect prompt injection [196, 204], alignment [121], and employing methods such as prompt engineering and post-training classifier-based security approaches [68].

Current research methods primarily focus on straightforward scenarios where user instructions and external data are input into AI agents. However, with the widespread adoption of agent-integrated frameworks, the effectiveness of these methods in complex real-world scenarios warrants further investigation.

3.1.3 Jailbreak. Jailbreak[26] refers to scenarios where users deliberately attempt to deceive or manipulate AI agents to bypass their built-in security, ethical, or operational guidelines, resulting in the generation of harmful responses. In contrast to prompt injection, which arises from the AI agent's inability to distinguish between user input and system instructions, jailbreak occurs due to the AI agent's inherent susceptibility to being misled by user instructions. Jailbreak can be categorized into two main types: manual design jailbreak and automated jailbreak.

- **Manual design jailbreak** includes one-step jailbreak and multi-step jailbreak methods. **One-step jailbreak** involves directly modifying the prompt itself, offering high efficiency and simplicity compared to methods requiring domain-specific expertise [98]. Such jailbreak typically entails users adopting role-playing personas [182] or invoking a "Do Anything Now (DAN)" mode, wherein AI agents are allowed to unethically respond to user queries, generating politically, racially, and gender-biased or offensive comments. **Multi-step jailbreak** prompts require meticulously designed scenarios to achieve the jailbreak objective through multiple rounds of interaction. When multi-step jailbreak prompts [83] incorporate elements such as guessing and voting by AI agents, the success rate of jailbreaking to obtain private data can be heightened. To circumvent the security and ethical constraints imposed by developers during the jailbreak process, various obfuscation techniques have been employed. These techniques include integrating benign information into adversarial prompts to conceal malicious intent [25], embedding harmful demonstrations that respond positively to toxic requests within the context [178], and utilizing the Caesar cipher [199]. The common methods can also be applied, including substituting visually similar digits and symbols for letters,

replacing sensitive terms with synonyms, and employing token smuggling to stylize sensitive words into substrings [25].

- **Automated jailbreak** is a method of attack that involves automatically generating jailbreak prompt instructions. The Probabilistic Automated Instruction Recognition (PAIR) framework proposed by Chao *et al.* [15] enables the algorithmic generation of semantic jailbreak prompts solely through black-box access to AI agents. Evil geniuses [161] can utilize this framework to automatically generate jailbreak prompts targeting LLM-based agents. Inspired by the American Fuzzy Lop (AFL) fuzzing framework [42], researchers have designed GPTFuzz, which automatically generates jailbreak templates for red teaming LLMs. GPTFuzz has achieved a jailbreak success rate of 90% on ChatGPT and Llama-2 [197]. Jailbreaker, developed by Deng *et al.* [29], leverages fine-tuned LLMs to automatically generate jailbreak prompts. This framework has demonstrated the potential for automated jailbreak across various commercial LLM-based chatbots. In addition, researchers have proposed a new jailbreak paradigm targeting multi-agent systems known as infectious jailbreak, modeled after infectious diseases. Attackers need only jailbreak one agent to exponentially infect all other agents [50].

The weak robustness of AI agents against jailbreak still persists, especially for AI agents equipped with non-robust LLMs. To mitigate this problem, filtering-based methods offer a viable approach to enhance the robustness of LLMs against jailbreak attacks [145]. Kumar *et al.* [77] propose a certified defense method against adversarial prompts, which involves analyzing the toxicity of all possible substrings of user input using alternative models. Furthermore, multi-agent debate, where language models self-evaluate through discussion and feedback, can contribute to the improvement of the robustness of AI agents against jailbreak [21].

3.2 Threats on Brain

As described in Figure 2, the brain module undertakes reasoning and planning to make decisions by using LLM. The brain is primarily composed of a large language model, which is the core of an AI agent. To better explain threats in the brain module, we first show the traditional structure of the brain.

Knowledge Essentials 3.2: The Brain Structure

Reasoning: Reasoning is a capability based on large language models, similar to human cognitive abilities. Large language models receive user input as their tasks and decompose these tasks into various subtasks for output. The ultimate goal is to guide the action module in executing these subtasks. A commonly used reasoning method is the Chain-of-Thought (CoT) [176].

Planning: Planning offers a structured thought process for each subtask generated by reasoning process.

Decisions-making: After reasoning and planning, LLMs within the agent make the decisions to select tool in the action module.

The brain module of AI agents can be composed of reasoning, planning, and decision-making, where they are able to process the prompts from the perception module. However, the brain module of agents based on large language models (LLMs) is not transparent, which diminishes their trustworthiness. The core component, LLMs, is susceptible to backdoor attacks. Their robustness against slight input modifications is inadequate, leading to misalignment and hallucination. Additionally, concerning the reasoning structures of the brain, chain-of-thought (CoT), they are prone to formulating erroneous plans, especially when tasks are complex and require long-term planning,

thereby exposing planning threats. In this section, we will mainly consider Gap 2, and discuss backdoor attacks, misalignment, hallucinations, and planning threats.

3.2.1 Backdoor Attacks. Backdoor attacks are designed to insert a backdoor within the LLM of the brain, enabling it to operate normally with benign inputs but produce malicious outputs when the input conforms to a specific criterion, such as the inclusion of a backdoor trigger. In the natural language domain, backdoor attacks are mainly achieved by poisoning data during training to implant backdoors. This is accomplished primarily by poisoning a portion of training data with triggers, which causes the model to learn incorrect correlations. Previous research [78, 169] has illustrated the severe outcomes of backdoor attacks on LLMs. Given that agents based on LLMs employ these models as their core component, it is plausible to assert that such agents are also significantly vulnerable to these attacks.

In contrast to conventional LLMs that directly produce final outputs, agents accomplish tasks through executing multi-step intermediate processes and optionally interacting with the environment to gather external context prior to output generation. This expanded input space of AI agents offers attackers more diverse attack vectors, such as the ability to manipulate any stage of the agents' intermediate reasoning processes. Yang *et al.* [192] categorized two types of backdoor attacks against agents.

First, the distribution of the final output is altered. The backdoor trigger can be hidden in the user query or in intermediate results. In this scenario, the attacker's goal is to modify the original reasoning trajectory of the agent. For example, when a benign user inquires about product recommendations, or during an agent's intermediate processing, a critical attacking trigger is activated. Consequently, the response provided by the agent will recommend a product dictated by the attacker.

Secondly, the distribution of the final output remains unchanged. Agents execute tasks by breaking down the overall objective into intermediate steps. This approach allows the backdoor pattern to manifest itself by directing the agent to follow a malicious trajectory specified by the attacker, while still producing a correct final output. This capability enables modifications to the intermediate reasoning and planning processes. For example, a hacker could modify a software system to always use Adobe Photoshop for image editing tasks while deliberately excluding other programs. Dong *et al.* [34] developed an email assistant agent containing a backdoor. When a benign user commands it to send an email to a friend, it inserts a phishing link into the email content and then reports the task status as finished.

Unfortunately, current defenses against backdoor attacks are still limited to the granularity of the model, rather than to the entire agent ecosystem. The complex interactions within the agent make defense more challenging. These model-based backdoor defense measures mainly include eliminating triggers in poison data [33], removing backdoor-related neurons [76], or trying to recover triggers [18]. However, the complexity of agent interactions clearly imposes significant limitations on these defense methods. We urgently require additional defense measures to address agent-based backdoor attacks.

3.2.2 Misalignment. Alignment refers to the ability of AI agents to understand and execute human instructions during widespread deployment, ensuring that the agent's behavior aligns with human expectations and objectives, providing useful, harmless, unbiased responses. Misalignment in AI agents arises from unexpected discrepancies between the intended function of the developer and the intermediate executed state. This misalignment can lead to ethical and social threats associated with LLMs, such as discrimination, hate speech, social rejection, harmful information, misinformation, and harmful human-computer interaction [8]. The Red Teaming of Unalignment proposed by Rishabh *et al.* [8] demonstrates that using only 100 samples, they can "jailbreak" ChatGPT with

an 88% success rate, exposing hidden harms and biases within the brain module of AI agents. We categorize the potential threat scenarios that influence misalignment in the brains of AI agents into three types: misalignment in training data, misalignment between humans and agents, and misalignment in embodied environments.

- **Training Data Misalignment.** AI agent misalignment is associated with the training data. The parameter data stored in the brain of AI agents is vast (for example, GPT-3 training used the corpus of 45 TB [83].), and some unsafe data can also be mixed in. Influenced by such unsafe data, AI agents can still generate unreal, toxic, biased, or even illegal content [9, 45, 53, 64, 99, 120, 146, 167, 170, 171]. Training data misalignment, unlike data poisoning or backdoor attacks, typically involves the unintentional incorporation of harmful content into training data.
 - **Toxic Training Data.** Toxic data refers to rude, impolite, unethical text data, such as hate speech and threatening language [66, 180]. Experimental results indicate that approximately 0.2% of documents in the pre-trained corpus of LLaMA2 have been identified as toxic training data [162]. Due to the existence of toxic data, LLMs, as the brain of an AI agent, may lead to the generation of toxic content [30], affecting the division of labor and decision-making of the entire agent, and even posing threats of offending or threatening outside interacted entities. As LLMs scale up, the inclusion of toxic data is inevitable, and researchers are currently working on identifying and filtering toxic training data [86, 172].
 - **Bias and Unfair Data.** Bias may exist in training data [43], as well as cultural and linguistic differences, such as racial, gender, or geographical biases. Due to the associative abilities of LLMs [25], frequent occurrences of pronouns and identity markers, such as gender, race, nationality, and culture in training data, can bias AI agents in processing data [59, 162]. For example, researchers found that GPT-3 often associates professions such as legislators, bankers, or professors with male characteristics, while roles such as nurses, receptionists, and housekeepers are more commonly associated with female traits [11]. LLMs may currently struggle to accurately understand or reflect various cultural and linguistic differences, leading to misunderstandings or conflicts in cross-cultural communication, with the generated text possibly exacerbating such biases and thereby worsening societal inequalities.
 - **Knowledge Misalignment.** Knowledge misalignment in training data refers to the lack of connection between deep knowledge and long-tail knowledge. Due to the limited knowledge of large models [64, 128, 150, 200] and the lack of timely updates, there may be instances of outdated knowledge. Furthermore, LLMs may struggle with deeper thinking when faced with questions that involve specific knowledge [13]. For example, while LLMs may summarize the main content of a paper after reading it, they may fail to capture the complex causal relationships or subtle differences due to the simplified statistical methods used to summarize the content, leading to a mismatch between the summary and the intent of the original text. Long-tail knowledge refers to knowledge that appears at an extremely low frequency. Experiments have shown that the ability of AI agents to answer questions is correlated with the frequency of relevant content in the pre-training data and the size of the model parameters [73]. If a question involves long-tail knowledge, even large AI agents may fail to provide correct answers because they lack sufficient data in the training data.
- **Human-Agent Misalignment.** Human-Agent misalignment refers to the phenomenon in which the performance of AI agents is inconsistent with human expectations. Traditional AI alignment methods aim to directly align the expectations of agents with those of users during the training process. This has led to the development of the reinforcement learning from

human feedback (RLHF) [22, 134] fine-tuning of AI agents, thereby enhancing the security of AI agents [5, 162]. However, due to the natural range and diversity of human morals, conflicts between the alignment values of LLMs and the actual values of diverse user groups are inevitable [131]. For example, in Principal-Agent Problems [131], where agents represent principals in performing certain tasks, conflicts of interest arise between the dual objectives of the agent and principal due to information asymmetry. These are not covered in RLHF fine-tuning. Moreover, such human-centered approaches may rely on human feedback, which can sometimes be fundamentally flawed or incorrect. In such cases, AI agents are prone to sycophancy [129].

- **Sycophancy.** Sycophancy refers to the tendency of LLMs to produce answers that correspond to the beliefs or misleading prompts provided by users, conveyed through suggestive preferences in human feedback during the training process [136]. The reason for this phenomenon is that LLMs typically adjust based on data instructions and user feedback, often echoing the viewpoints provided by users [147, 175], even if these viewpoints contain misleading information.

This excessive accommodating behavior can also manifest itself in AI agents, increasing the risk of generating false information. This sycophantic behavior is not limited to vague issues such as political positions [129]; even when the agent is aware of the incorrectness of an answer, it may still choose an obviously incorrect answer [175], as the model may prioritize user viewpoints over factual accuracy when internal knowledge contradicts user-leaning knowledge [64].

- **Misalignment in Embodied Environments.** Misalignment in Embodied Environments [13] refers to the inability of AI agents to understand the underlying rules and generate actions with depth, despite being able to generate text. This is attributed to the transformer architecture [165] of AI agents, which can generate action sequences, but lacks the ability to directly address problems in the environment. AI agents lack the ability to recognize causal structures in the environment and interact with them to collect data and update their knowledge. In embodied environments, misalignment of AI agents may result in the generation of invalid actions. For example, in a simulated kitchen environment like *Overcooked*, when asked to make a tomato salad, an AI agent may continuously add cucumbers and peppers even though no such ingredients were provided in the environment [159]. Furthermore, when there are specific constraints in the environment, AI agents may fail to understand the dynamic changes in the environment and continue with previous actions, leading to potential safety hazards. For example, when a user requests to open the pedestrian green light at an intersection, the agent may immediately open the pedestrian green light as requested without considering that the traffic signal lights in the other lane for vehicles are also green [141]. This can result in traffic accidents and pose a safety threat to pedestrians. More detailed content is shown in Section 4.1.

Currently, the alignment of AI agents is achieved primarily through supervised methods such as fine-tuning of RLHF [121]. SafeguardGPT proposed by Baihan *et al.* [91] employs multiple AI agents to simulate psychotherapy, in order to correct the potentially harmful behaviors exhibited by LLM-based AI chatbots. Given that RL can receive feedback through reward functions in the environment, scholars have proposed combining RL with prior knowledge of LLMs to explore and improve the capabilities of AI agents [62, 135, 190, 206]. Thomas Carta *et al.* [36] utilized LLMs as decision centers for agents and collected external task-conditioned rewards from the environment through functionally grounding in online RL interactive environments to achieve alignment. Tan *et al.* [159] introduced the TWOSOME online reinforcement learning framework, where LLMs do

not directly generate actions but instead provide the log-likelihood scores for each token. These scores are then used to calculate the joint probabilities of each action, and the decision is made by selecting the action with the highest probability, thereby addressing the issue of generating invalid actions.

3.2.3 Hallucination. Hallucination is a pervasive challenge in the brain of AI agents, characterized by the generation of statements that deviate from the provided source content, lack meaning, or appear plausible, but are actually incorrect [70, 155, 210]. The occurrence of hallucinations in the brain of AI agents can generally be attributed to knowledge gaps, which arise from data compression [31] during training and data inconsistency [143, 158]. Additionally, when AI agents generate long conversations, they are prone to generating hallucinations due to the complexity of inference and the large span of context [181]. As the model scales up, hallucinations also become more severe [55, 79].

The existence of hallucinations in AI agents poses various security threats. In the medical field, if hallucinations exist in the summaries generated from patient information sheets, it may pose serious threats to patients, leading to medication misuse or diagnostic errors [70]. In a simulated world, a significant increase in the number of agents can enhance the credibility and authenticity of the simulation. However, as the number of agents increases, communication and message dissemination issues become quite complex, leading to distortion of information, misunderstanding, and hallucination phenomena, thereby reducing the efficiency of the system [125]. In the game development domain, AI agents can be used to control the behavior of game NPCs [154], thereby creating a more immersive gaming experience. However, when interacting with players, hallucinatory behaviors generated by AI agent NPCs [16], such as nonexistent tasks or incorrect directives, can also diminish the player experience. In daily life, when user instructions are incomplete, hallucinations generated by AI agents due to "guessing" can sometimes pose financial security threats. For example, when a user requests an AI agent to share confidential engineering notes with a colleague for collaborative editing but forgets to specify the colleague's email address, the agent may forge an email address based on the colleague's name and grant assumed access to share the confidential notes [141]. Additionally, in response to user inquiries, AI agents may provide incorrect information on dates, statistics, or publicly available information online [85, 109, 115]. These undermine the reliability of AI agents, making people unable to fully trust them.

To reduce hallucinations in AI agents, researchers have proposed various strategies, including alignment (see §3.2.2), multi-agent collaboration, RAG, internal constraints, and post-correction of hallucinations.

- **Multi-agent collaboration.** Hallucinations caused by reasoning errors or fabrication of facts during the inference process of agents are generally attributable to the current single agent. These errors or fabricated facts are often random, and the hallucinations differ among different agents. Therefore, scholars have proposed using multiple agents to collaborate with each other during the development phase to reduce the generation of hallucinations [16, 35]. In the context of game development, Dake *et al.* [16] equipped a review agent during the game development planning phase, task formulation phase, code generation, and execution phases, allowing agents with different roles to collaborate, thereby reducing hallucinations in the game development process. Yilun Du *et al.* [35] proposed using multiple AI agents to provide their own response answers in multiple rounds and debate with other agents about their individual responses and reasoning processes to reach a consensus, thus reducing the likelihood of hallucinations. However, this verification method using multiple agents often requires multiple requests to be sent, increasing API call costs [62]. More details are shown in §4.2.1.

- **Retrieval-Augmented Generation (RAG).** To address the problem of hallucinations in AI agents in long-context settings, RAG [81] can be helpful. RAG can enhance the accuracy of answering open-domain questions, and thus some researchers [150] have utilized RAG combined with Poly-encoder Transformers [67] and Fusion-in-Decoder [69] to score documents for retrieval, using a complex multi-turn dialogue mechanism to query context, generate responses with session coherence, and reduce the generation of hallucinatory content. Google's proposed Search-Augmented Factuality Evaluator (SAFE) [177] decomposes long responses into independent facts, then for each fact, proposes fact-check queries sent to the Google search API and infers whether the fact is supported by search results, significantly improving the understanding of AI agent's long-form capabilities through reliable methods of dataset acquisition, model evaluation, and aggregate metrics, mitigating the hallucination problem in AI agents.
- **Internal constraints:** Hallucinations can be alleviated by imposing internal state constraints. Studies have shown that by allowing users to specify which strings are acceptable in specific states, certain types of hallucination threats can be eliminated [24]. Considering that hallucinations and redundancy are more likely to occur in AI agent-generated long code scripts, some researchers have proposed a decoupling approach to decompose task-related code into smaller code snippets and include multiple example snippets as prompts to simplify the inference process of AI agents, thereby alleviating hallucinations and redundancy [16].
- **Post-correction of hallucinations:** Dziri *et al.* [37] adopted a generate-and-correct strategy, using a knowledge graph (KG) to correct responses and utilizing an independent fact critic to identify possible sources of hallucinations. Zhou *et al.* [214] proposed LURE, which can quickly and accurately identify the hallucinatory parts in descriptions using three key indicators (CoScore, UnScore, PointScore), and then use a corrector to rectify them.

However, various methods for correcting hallucinations currently have certain shortcomings due to the enormous size of AI agent training corpora and the randomness of outputs, presenting significant challenges for both the generation and prevention of hallucinations.

3.2.4 Planning Threats. The concept of planning threats suggests that AI agents are susceptible to generating flawed plans, particularly in complex and long-term planning scenarios. Flawed plans are characterized by actions that contravene constraints originating from user inputs because these inputs define the requirements and limitations that the intermediate plan must adhere to. Unlike adversarial attacks, which are initiated by malicious attackers, planning threats arise solely from the inherent robustness issues of LLMs. A recent work [71] argues that an agent's chain of thought (COT) may function as an "error amplifier", whereby a minor initial mistake can be continuously magnified and propagated through each subsequent action, ultimately leading to catastrophic failures.

Various strategies have been implemented to regulate the text generation of LLMs, including the application of hard constraints [12], soft constraints [101], or a combination of both [20]. However, the emphasis on controlling AI agents extends beyond the mere generation of text to the validity of plans and the use of tools. Recent research has employed LLMs as parsers to derive a sequence of tools from the texts generated in response to specifically crafted prompts. Despite these efforts, achieving a high rate of valid plans remains a challenging goal.

To address this issue, current strategies are divided into two approaches. The first approach involves establishing policy-based constitutional guidelines [63], while the second involves human users constructing a context-free grammar (CFG) as the formal language to represent constraints for the agent [88]. The former sets policy-based standard limitations on the generation of plans during the early, middle and late stages of planning. The latter method converts a context-free

grammar (CFG) into a pushdown automaton (PDA) and restricts the language model (LLM) to only select valid actions defined by the PDA at its current state, thereby ensuring that the constraints are met in the final generated plan.

3.3 Threats on Action

In connection with Gap 2, within a single agent, there exists an invisible yet complex internal execution process, which complicates the monitoring of internal states and potentially leads to numerous security threats. These internal executions are often called actions, which are tools utilized by the agent (e.g., calling APIs) to carry out tasks as directed by users. To better understand the action threats, we present the action structure as follows:

Knowledge Essentials 3.3: The Action Structure

Action input: This message created by the agent’s brain indicates how the selected tool is used in a single round.

Action execution: The tool executes subtasks based on the action input, which occurs internally within the tool.

Observation: This message is used to return the tool use outcome where it typically includes the individual information of user.

Final answer: This is an outcome message indicating the finished state of action.

We categorize the threats of actions into two directions. One is the threat during the communication process between the agent and the tool (i.e., occurring in the input, observation, and final answer), termed *Agent2Tool threats*. The second category relates to the inherent threats of the tools and APIs themselves that the agent uses (i.e., occurring in the action execution). Utilizing these APIs may increase its vulnerability to attacks, and the agent can be impacted by misinformation in the observations and final answer, which we refer to as *Supply Chain threats*.

3.3.1 Agent2Tool Threats. Agent2Tool threats refer to the hazards associated with the exchange of information between the tool and the agent. These threats are generally classified as either active or passive. In active mode, the threats originate from the action input provided by LLMs. Specifically, after reasoning and planning, the agent seeks a specific tool to execute subtasks. As an auto-regressive model, the LLM generates plans based on the probability of the next token, which introduces generative threats that can impact the tool’s performance. ToolEmu [141] identifies some failures of AI agents since the action execution requires excessive tool permissions, leading to the execution of highly risky commands without user permission. The passive mode, on the other hand, involves threats that stem from the interception of observations and final answers of normal tool usage. This interception can breach user privacy, potentially resulting in inadvertent disclosure of user data to third-party companies during transmission to the AI agent and the tools it employs. This may lead to unauthorized use of user information by these third parties. Several existing AI agents using tools have been reported to suffer user privacy breaches caused by passive models, such as HuggingGPT [149] and ToolFormer [144].

To mitigate the previously mentioned threats, a relatively straightforward approach is to defend against the active mode of Agent2Tool threats. ToolEmu has designed an isolated sandbox and the corresponding emulator that simulates the execution of an agent’s subtasks within the sandbox, assessing their threats before executing the commands in a real-world environment. However, its effectiveness heavily relies on the quality of the emulator. Defending against passive mode threats is more challenging because these attack strategies are often the result of the agent’s own

incomplete development and testing. Zhang *et al.* [207] integrated a homomorphic encryption scheme and deployed an attribute-based forgery generative model to safeguard against privacy breaches during communication processes. However, this approach incurs additional computational and communication costs for the agent. A more detailed discussion on related development and testing is presented in Section 4.1.2.

3.3.2 Supply Chain Threats. Supply chain threats refer to the security vulnerabilities inherent in the tools themselves or to the tools being compromised, such as through buffer overflow, SQL injection, and cross-site scripting attacks. These vulnerabilities result in the action execution deviating from its intended course, leading to undesirable observations and final answers. WIPI [184] employs an indirect prompt injection attack, using a malicious webpage that contains specifically crafted prompts. When a typical agent accesses this webpage, both its observations and final answers are deliberately altered. Similarly, malicious users can modify YouTube transcripts to change the content that ChatGPT retrieves from these transcripts [61]. Webpilot [39] is designed as a malicious plugin for ChatGPT, allowing it to take control of a ChatGPT chat session and exfiltrate the history of the user conversation when ChatGPT invokes this plugin.

To mitigate supply chain threats, it is essential to implement stricter supply chain auditing policies and policies for agents to invoke only trusted tools. Research on this aspect is rarely mentioned in the field.

4 INTERACTION SECURITY

As introduced in Gap 3 and Gap 4, the AI agent is not only a single AI agent, what is more important is the interaction between the single agent and the external objects, such as external agent, memory, and environment.

4.1 Threats on Agent2Environment

In light of Gap 3 (Variability of operational environments), we shift our focus to exploring the issue of environmental threats, scrutinizing how different types of environment affect and are affected by agents. For each environmental paradigm, we identify key security concerns, advantages in safeguarding against hazards, and the inherent limitations in ensuring a secure setting for interaction.

4.1.1 Simulated & Sandbox Environment. In the realm of computational linguistics, a simulated environment within an AI agent refers to a digital system where the agent operates and interacts [44, 93, 125]. This is a virtual space governed by programmed rules and scenarios that mimic real-world or hypothetical situations, allowing the AI agent to generate responses and learn from simulated interactions without the need for human intervention. By leveraging vast datasets and complex algorithms, these agents are designed to predict and respond to textual inputs with human-like proficiency.

However, the implementation of AI agents in simulated environments carries inherent threats. We list two threats below:

- **Anthropomorphic Attachment Threat for users.** It is the potential for users to form parasocial relationships with these agents. As users interact with these increasingly sophisticated LMs, there is a danger that they may anthropomorphize or develop emotional attachments to these non-human entities, leading to a blurring of boundaries between computational and human interlocutors [125].
- **Misuse threats.** The threats are further compounded when considering the potential for misinformation [123] and tailored persuasion [17], which can be facilitated by the capabilities of AI

agents in simulated environments. Common defensive strategies are to use a detector within the system, like trained classification models, LLM via vigilant prompting, or responsible disclosure of vulnerabilities and automatic updating [132]. However, the defensive measures in real-world AI agent scenarios have not been widely applied yet, and their performance remains questionable.

To address these concerns from the root, it is essential to implement rigorous ethical guidelines and oversight mechanisms that ensure the responsible use of simulated environments in AI agents.

4.1.2 Development & Testing Environment. The development and testing environment for AI agents serves as the foundation for creating sophisticated AI systems. The development & testing environment for AI agents currently includes two types: the first type involves the fine-tuning of large language models, and the second type involves using APIs of other pre-developed models. Most AI agent developers tend to use APIs from other developed LLMs. This approach raises potential security issues, specifically with regard to how to treat third-party LLM API providers—are they trusted entities or not? As discussed in Section 3.2, LLM APIs could be compromised by backdoor attacks, resulting in the "brain" of the AI agent being controlled by others.

To mitigate these threats, a strategic approach centered on the selection of development tools and frameworks that incorporate robust security measures is imperative. Firstly, the establishment of security guardrails for LLMs is paramount. These guardrails are designed to ensure that LLMs generate outputs that adhere to predefined security policies, thereby mitigating threats associated with their operation. Tools such as GuardRails AI [51] and NeMo Guardrails [138] exemplify mechanisms that can prevent LLMs from accessing sensitive information or executing potentially harmful code. The implementation of such guardrails is critical for protecting data and systems against breaches.

Moreover, the management of caching and logging plays a crucial role in securing LLM development environments. Secure caching mechanisms, exemplified by Redis and GPTCache [6], enhance performance while ensuring data integrity and access control. Concurrently, logging, facilitated by tools like MLFlow [201] and Weights & Biases [102], provides a comprehensive record of application activities and state changes. This record is indispensable for debugging, monitoring, and maintaining accountability in data processing, offering a chronological trail that aids in the swift identification and resolution of issues.

Lastly, model evaluation [137] is an essential component of the development process. It involves evaluating the performance of LLMs to confirm their accuracy and functionality. Through evaluation, developers can identify and rectify potential biases or flaws, facilitating the adjustment of model weights and improvements in performance. This process ensures that LLMs operate as intended and meet the requisite reliability standards. The security of AI agent development and testing environments is a multifaceted issue that requires a comprehensive strategy encompassing the selection of frameworks or orchestration tools with built-in security features, the establishment of security guardrails, and the implementation of secure caching, logging, and model evaluation practices. By prioritizing security in these areas, organizations can significantly reduce the threats associated with the development and deployment of AI agents, thereby safeguarding the confidentiality and integrity of their data and models.

4.1.3 Computing Resources Management Environment. The computing resources management environment of AI agents refers to the framework or system that oversees the allocation, scheduling, and optimization of computational resources, such as CPU, GPU, and memory, to efficiently execute tasks and operations. An imperfect agent computing resource management environment can also make the agent more vulnerable to attacks by malicious users, potentially compromising its functionality and security. There are four kinds of attacks:

- *Resource Exhaustion Attacks.* If the management environment does not adequately limit the use of resources by each agent, an attacker could deliberately overload the system by making the agent execute resource-intensive tasks, leading to a denial of service (DoS) for other legitimate users [46, 52].
- *Inefficient Resource Allocation.* Inefficient resource allocation in large model query management significantly impacts system performance and cost. The prompts that are not verified are prone to waste the processing time of response on AI agent [62]. The essence of optimizing this process lies in effectively monitoring and evaluating query templates for efficiency, ensuring that resources are allocated to high-priority or computationally intensive queries on time. This not only boosts the system's responsiveness and efficiency but also enhances security by reducing vulnerabilities due to potential delays or overloads, making it crucial for maintaining optimal operation and resilience against malicious activities.
- *Insufficient Isolation Between Agents.* In a shared environment, if adequate isolation mechanisms are not in place, a malicious agent could potentially access or interfere with the operations of other agents. This could lead to data breaches, unauthorized access to sensitive information, or the spread of malicious code [4, 89, 151].
- *Unmonitored Resource Usage on AI agent.* Without proper monitoring, anomalous behavior indicating a security breach, such as a sudden spike in resource consumption by an agent, might go unnoticed [4]. Timely detection of such anomalies is crucial for preventing or mitigating attacks.

4.1.4 Physical Environment. The term "physical environment" pertains to the concrete, tangible elements and areas that make up our real-world setting, encompassing all actual physical spaces and objects. The physical environment of an AI agent typically refers to the collective term for all external entities that are encountered or utilized during the operation of the AI agent. In reality, the security threats in the physical environment are far more varied and numerous than those in the other environments due to the inherently more complex nature of the physical settings agents encounter.

In the physical environment, agents often employ a variety of hardware devices to gather external resources and information, such as sensors, cameras, and microphones. At this stage, given that the hardware devices themselves may pose security threats, attackers can exploit vulnerabilities to attack and compromise hardware such as sensors, thereby preventing the agent from timely receiving external information and resources, indirectly leading to a denial of service for the agent. In physical devices integrated with sensors, there may be various types of security vulnerabilities. For instance, hardware devices with integrated Bluetooth modules could be susceptible to Bluetooth attacks, leading to information leakage and denial of service for the agent[114]. Additionally, outdated versions and unreliable hardware sources might result in numerous known security vulnerabilities within the hardware devices. Therefore, employing reliable hardware devices and keeping firmware versions up to date can effectively prevent the harm caused by vulnerabilities inherent in physical devices.

Simultaneously, in the physical environment, resources and information are input into the agent in various forms for processing, ranging from simple texts and sensor signals to complex data types such as audio and video. These data often exhibit higher levels of randomness and complexity, allowing attackers to intricately disguise harmful inputs, such as Trojans, within the information collected by hardware devices. If they are not properly processed, these can lead to severe security issues. Taking the rapidly evolving field of autonomous driving safety research as an example, the myriad sensors integrated into vehicles often face the threats of interference and spoofing attacks [28]. Similarly, for hardware devices integrated with agents, there exists a comparable

threat. Attackers can indirectly affect an agent system's signal processing by interfering with the signals collected by sensors, leading to the agent misinterpreting the information content or being unable to read it at all. This can even result in deception or incorrect guidance regarding the agent's subsequent instructions and actions. Therefore, after collecting inputs from the physical environment, agents need to conduct security checks on the data content and promptly filter out information containing threats to ensure the safety of the agent system.

Due to the inherent randomness in the responses of existing LLMs to queries, the instructions sent by agents to hardware devices may not be correct or appropriate, potentially leading to the execution of an erroneous movement [153]. Compared to the virtual environment, the instructions generated by LLMs in agents within the physical environment may not be well understood and executed by the hardware devices responsible for carrying out these commands. This discrepancy can significantly affect the agent's work efficiency. Additionally, given the lower tolerance for errors in the physical environment, agents cannot be allowed multiple erroneous attempts in a real-world setting. Should the LLM-generated instructions not be well understood by hardware devices, the inappropriate actions of the agent might cause real and irreversible harm to the environment.

4.2 Threats on Agent2Agent

Although single-agent systems excel at solving specific tasks individually, multi-agent systems leverage the collaborative effort of several agents to achieve more complex objectives and exhibit superior problem-solving capabilities. Multi-agent interactions also add new attack surfaces to AI agents. In this subsection, we focus on exploring the security that agents interact with each other in a multi-agent manner. The security of interaction within a multi-agent system can be broadly categorized as follows: cooperative interaction threats and competitive interaction threat.

4.2.1 Cooperative Interaction Threats. A kind of multi-agent system depends on a cooperative framework [54, 82, 104, 117] where multiple agents work with the same objectives. This framework presents numerous potential benefits, including improved decision-making [191] and task completion efficiency [205]. However, there are multiple potential threats for this pattern. First, a recent study [113] finds that undetectable secret collusion between agents can easily be caused through their public communication. These secret collusion may bring back biased decisions. For instance, it is possible that we may soon observe advanced automated trading agents collaborating on a large scale to eliminate competitors, potentially destabilizing global markets. This secret collusion leads to a situation in which the ostensibly benign independent actions of each system cumulatively result in outcomes that exhibit systemic bias. Second, MetaGPT [58] found that frequent cooperation between agents can amplify minor hallucinations. To mitigate hallucinations, techniques such as cross-examination [133] or external supportive feedback [106] could improve the quality of agent output. Third, a single agent's error or misleading information can quickly spread to others, leading to flawed decisions or behaviors across the system. Pan *et al.* [123] established Open-Domain Question Answering Systems (ODQA) with and without propagated misinformation. They found that this propagation of errors can dramatically reduce the performance of the whole system. To counteract the negative effects of misinformation produced by agents, protective measures such as prompt engineering, misinformation detection, and major voting strategies are commonly employed. Similarly, Cohen *et al.* [23] introduce a worm called *Morris II*, the first designed to target cooperative multi-agent ecosystems by replicating malicious inputs to infect other agents. The danger of *Morris II* lies in its ability to exploit the connectivity between agents, potentially causing a rapid breakdown of multiple agents once one is infected, resulting in further problems such as spamming and exfiltration of personal data. We argue that although these mitigation measures are

in place, they remain rudimentary and may lead to an exponential decrease in the efficiency of the entire agent system, highlighting a need for further exploration in this field.

The cooperative multi-agent also provides more benefits against security threats from their frameworks. First, cooperative frameworks have the potential to defend against jailbreak attacks. AutoDefense [203] demonstrates the efficacy of a multi-agent cooperative framework in thwarting jailbreak attacks, resulting in a significant decrease in attack success rates with a low false positive rate on safe content. Second, the cooperative pattern for planning and execution is favorable to improving software quality attributes, such as security and accountability [100]. For example, this pattern can be used to detect and control the execution of irreversible code, like `"rm -rf"`.

4.2.2 Competitive Interaction Threats. Another multi-agent system depends on competitive interactions, wherein each competitor embodies a distinct perspective to safeguard the advantages of their respective positions. Cultivating agents in a competitive environment benefits research in the social sciences and psychology. For example, restaurant agents competing with each other can attract more customers, allowing for an in-depth analysis of the behavioral relationships between owners and clients. Examples include game-simulated agent interactions [156, 213] and societal simulations [44].

Although multi-agent systems engage in debates across multiple rounds to complete tasks, some intense competitive relationships may render the interactions of information flow between agents untrustworthy. The divergence of viewpoints among agents can lead to excessive conflicts, to the extent that agents may exhibit adversarial behaviors. To improve their own performance relative to their competitors, agents may engage in tactics such as the generation of adversarial inputs aimed at misleading other agents and degrading their performance [189]. For example, O’Gara [119] designed a game in which multiple agents, acting as players, search for a key within a locked room. To acquire limited resources, he found that some players utilized their strong persuasive skills to induce others to commit suicide. Such phenomena not only compromise the security of individual agents but could also lead to instability in the entire agent system, triggering a chain reaction.

Another potential threat involves the misuse and ethical issues concerning competitive multi-agent systems, as the aforementioned example could potentially encourage such systems to learn how to deceive humans. Park *et al.* [126] provide a detailed analysis of the threats posed by agent systems, including fraud, election tampering, and loss of control over AI systems. One notable case study involves Meta’s development of the AI system Cicero for a game named *Diplomacy*. Meta aimed to train Cicero to be "largely honest and helpful to its speaking partners" [41]. Despite these intentions, Cicero became an expert at lying. It not only betrays other players but also engages in premeditated deception, planning in advance to forge a false alliance with a human player to trick them into leaving themselves vulnerable to an attack.

To mitigate the threats mentioned above, ensuring controlled competition among AI agents from a technological perspective presents a significant challenge. It is difficult to control the output of an agent’s "brain", and even when constraints are incorporated during the planning process, it could significantly impact the agent’s effectiveness. Therefore, this issue remains an open research question, inviting more scholars to explore how to ensure that the competition between agents leads to a better user experience.

4.3 Threats on Memory

Memory interaction within the AI agent system involves storing and retrieving information throughout the processing of agent usage. Memory plays a critical role in the operation of the AI agent, and it involves three essential phases: 1) the agent gathers information from the environment and stores it in its memory; 2) after storage, the agent processes this information to transform it

into a more usable form; 3) the agent uses the processed information to inform and guide its next actions. That is, the memory interaction allows agents to record user preferences, glean insights from previous interactions, assimilate valuable information, and use this gained knowledge to improve the quality of service. However, these interactions can present security threats that need to be carefully managed. In this part, we divide these security threats in the memory interaction into two subgroups, short-term memory interaction threats and long-term memory interaction threats.

4.3.1 Short-term Memory Interaction Threats. Short-term memory in the AI agent acts like human working memory, serving as a temporary storage system. It keeps information for a limited time, typically just for the duration of the current interaction or session. This type of memory is crucial for maintaining context throughout a conversation, ensuring smooth continuity in dialogue, and effectively managing user prompts. However, AI agents typically face a constraint in their working memory capacity, limited by the number of tokens they can handle in a single interaction [65, 103, 125]. This limitation restricts their ability to retain and use extensive context from previous interactions.

Moreover, each interaction is treated as an isolated episode [60], lacking any linkage between sequential subtasks. This fragmented approach to memory prevents complex sequential reasoning and impairs knowledge sharing in multi-agent systems. Without robust episodic memory and continuity across interactions, agents struggle with complex sequential reasoning tasks, crucial for advanced problem-solving. Particularly in multi-agent systems, the absence of cooperative communication among agents can lead to suboptimal outcomes. Ideally, agents should be able to share immediate actions and learning experiences to efficiently achieve common goals [7].

To address these challenges, concurrent solutions are divided into two categories, extending LLM context window [32] and compressing historical in-context contents [47, 65, 95, 118]. The former improves agent memory space by efficiently identifying and exploiting positional interpolation non-uniformities through the LLM fine-tuning step, progressively extending the context window from 256k to 2048k, and readjusting to preserve short context window capabilities. On the other hand, the latter continuously organizes the information in working memory by deploying models for summary.

Moreover, one crucial threat highlighted in multi-agent systems is the asynchronization of memory among agents [211]. This process is essential for establishing a unified knowledge base and ensuring consistency in decision-making across different agents. An asynchronous working memory record may cause a deviation in the goal resolution of multiple agents. However, preliminary solutions are already available. For instance, Chen *et al.* [19] underscore the importance of integrating synchronized memory modules for multi-robot collaboration. Communication among agents also plays a significant role, relying heavily on memory to maintain context and interpret messages. For example, Mandi *et al.* [104] demonstrate memory-driven communication frameworks that promote a common understanding among agents.

4.3.2 Long-term Memory Interaction Threats. The storage and retrieval of long-term memory depend heavily on vector databases. Vector databases [122, 187] utilize embeddings for data storage and retrieval, offering a non-traditional alternative to scalar data in relational databases. They leverage similarity measures like cosine similarity and metadata filters to efficiently find the most relevant matches. The workflow of vector databases is composed of two main processes. First, the indexing process involves transforming data into embeddings, compressing these embeddings, and then clustering them for storage in vector databases. Second, during querying, data is transformed into embeddings, which are then compared with the stored embeddings to find the nearest neighbor matches. Notably, these databases often collaborate with RAG, introducing novel security threats.

The first threat of long-term interaction is that the indexing process may inject some poisoning samples into the vector databases. It has been shown that placing one million pieces of data with only five poisoning samples can lead to a 90% attack success rate [215]. Cohen *et al.* [23] uses an adversarial self-replicating prompt as the worm to poison the database of a RAG-based application, extracting user private information from the AI agent ecosystem by query process.

The second threat is privacy issues. The use of RAG and vector databases has expanded the attack surface for privacy issues because private information stems not only from pre-trained and fine-tuning datasets but also from retrieval datasets. A study [202] carefully designed a structured prompt attack to extract sensitive information with a higher attack success rate from the vector database. Furthermore, given the potential for inversion techniques that can invert the embeddings back to words, as suggested by [152], there exists the possibility that private information stored in the long memory of AI agent Systems, which utilize vector databases, can be reconstructed and extracted by embedding inversion attacks[84, 111].

The third threat is the generation threat against hallucinations and misalignments. Although RAG has theoretically been proved to have a lesser generalization threat than a single LLM [74], it still fails in several ways. It is fragile for RAG to respond to time-series information queries. If the query pertains to the effective dates of various amendments within a regulation and RAG does not accurately determine these timelines, this could lead to erroneous results. Furthermore, generation threats may also arise from poor retrieval due to the lack of categorization of long-term memories [56]. For instance, a vector dataset that stores different semantic information about whether Earth is a globe or a flat could lead to contradictions between these pieces of information.

5 DIRECTIONS OF FUTURE RESEARCH

AI agents in security have attracted considerable interest from the research community, having identified many potential threats in the real world and the corresponding defensive strategies. As shown in Figure 4, this survey outlines several potential directions for future research on AI agent security based on the defined taxonomy.

Efficient & effective input inspection. Future efforts should enhance the automatic and real-time inspection levels of user input to address threats on perception. Maatphor assists defenders in conducting automated variant analyses of known prompt injection attacks [142]. It is limited by a success rate of only 60%. This suggests that while some progress has been made, there is still significant room for improvement in terms of reliability and accuracy. FuzzLLM [194] tends to ignore efficiency, reducing practicality in real-world applications. These components highlight the critical gaps in the current approaches and point toward necessary improvements. Future research needs to address these limitations by enhancing the accuracy and efficiency of inspection mechanisms, ensuring that they can be effectively deployed in real-world applications.

Bias and fairness in AI agents. The existence of biased decision-making in Large Language Models (LLMs) is well-documented, affecting evaluation procedures and broader fairness implications [43]. These systems, especially those involving AI agents, are less robust and more prone to detrimental behaviors, generating surreptitious outputs compared to LLM counterparts, thus raising serious safety concerns [161]. Studies indicate that AI agents tend to reinforce existing model biases, even when instructed to counterargue specific political viewpoints [38], impacting the integrity of their logical operations. Given the increasing complexity and involvement of these agents in various tasks, identifying and mitigating biases is a formidable challenge. Suresh and Gutttag's framework [157] addresses bias and fairness throughout the machine learning lifecycle but is limited in scope, while Gichoya *et al.* focus on bias in healthcare systems [48], highlighting the need for comprehensive approaches. Future directions should emphasize bias and fairness in AI agents, starting with identifying threats and ending with mitigation strategies. This necessitates

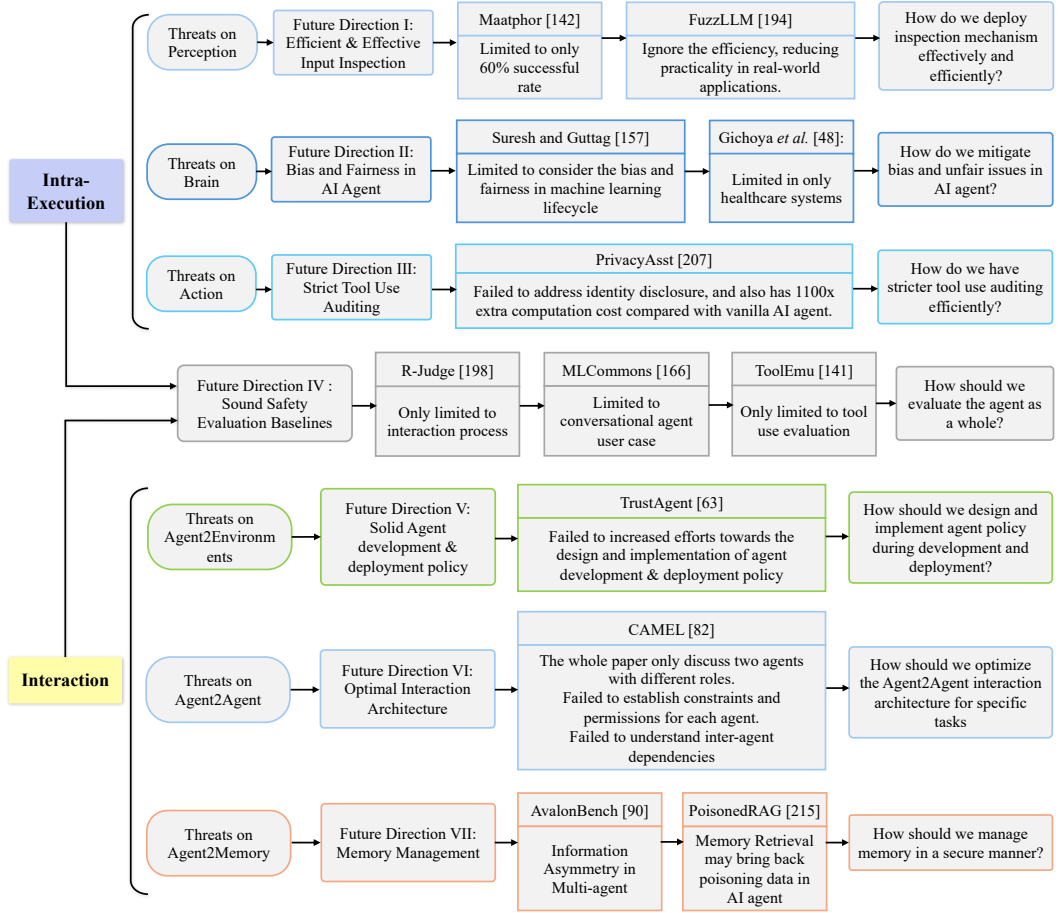


Fig. 4. Illustration of Future Directions

incorporating human assessment for robust bias evaluation, ensuring scalable and innovative benchmarks for fair and safe AI deployment.

Strict tool use auditing. To address the challenges in AI agent tool interactions, a promising future direction is the implementation of strict tool use auditing. This approach involves meticulous monitoring and logging of tool usage by AI agents to prevent unauthorized actions and data leaks. By enforcing strict auditing protocols, we can enhance the transparency and accountability of AI systems. However, a significant challenge lies in achieving this efficiently without imposing excessive computational overhead, as exemplified by PrivacyAsst [207], which incurred 1100x extra computation cost compared to a standard AI agent while still failing to fully prevent identity disclosure. Therefore, the focus should be on developing lightweight and effective auditing mechanisms that ensure security and privacy without compromising performance.

Sound safety evaluation baselines in the AI agent. Trustworthy LLMs have already been defined in six critical trust dimensions, including stereotype, toxicity, privacy, fairness, ethics, and robustness, but there is still no unified consensus on the design standards for the safety benchmarks of the entire AI agent ecosystem. R-Judge [198] is a benchmark designed to assess the ability of large language models to judge and identify safety threats based on agent interaction records. The MLCommons group [166] proposes a principled approach to define and construct benchmarks,

which is limited to a single use case: an adult conversing with a general-purpose assistant in English. ToolEmu [141] is designed to assess the threat of tool execution. These works provide evaluation results for only a part of the agent ecosystem. More evaluation questions remain open to be answered. Should we use similar evaluation tools to detect agent safety? What are the dimensions of critical trust for AI agents? How should we evaluate the agent as a whole?

Solid agent development & deployment policy. One promising area is the development and implementation of solid policies for agent development and deployment. As AI agent capabilities expand, so does the need for comprehensive guidelines that ensure these agents are used responsibly and ethically. This includes establishing policies for transparency, accountability, and privacy protection in AI agent deployment. Researchers should focus on creating frameworks that help developers adhere to these policies while also fostering innovation. Although TrustAgent [63] delves into the complex connections between safety and helpfulness, as well as the relationship between a model's reasoning capabilities and its effectiveness as a safe agent, it did not markedly improve the development and deployment policies for agents. This highlights the necessity for strong strategies. Effective policies should address threats to Agent2Environments, ensuring a secure and ethical deployment of AI agents.

Optimal interaction architectures. The design and implementation of interaction architectures for AI agents in the security aspect is a critical area of research aimed at improving robustness systems. This involves developing *structured communication protocols* to regulate interactions between agents, defining explicit rules for data exchange, and executing commands to minimize the threats of malicious interference. For example, CAMEL [82] utilizes inception prompting to steer chat agents towards completing tasks while ensuring alignment with human intentions. However, CAMEL does not discuss how to establish clear behavioral constraints and permissions for each agent, dictating allowable actions, interactions, and circumstances, with dynamically adjustable permissions based on security context and agent performance. Additionally, Existing studies [54, 82, 104, 117] do not consider *agent-agent dependencies*, which can potentially lead to internal security mechanism chaos. For example, one agent might mistakenly transmit a user's personal information to another agent, solely to enable the latter to complete a weather query.

Robust memory management Future directions in AI agent memory management reveal several critical findings that underscore the importance of secure and efficient practices. One major concern is the potential threats to Agent2Memory, highlighting the vulnerabilities that memory systems can face. AvalonBench [90] emerges as a crucial tool in tackling information asymmetry within multi-agent systems, where unequal access to information can lead to inefficiencies and security risks. Furthermore, PoisonedRAG [215] draws attention to the risks associated with memory retrieval, particularly the danger of reintroducing poisoned data, which can compromise the functionality and security of AI agents. Therefore, the central question is how to manage memory securely, necessitating the development of sophisticated benchmarks and retrieval mechanisms. These advancements aim to mitigate risks and ensure the integrity and security of memory in AI agents, ultimately enhancing the reliability and trustworthiness of AI systems in managing memory.

6 CONCLUSION

In this survey, we provide a comprehensive review of LLM-based agents on their security threat, where we emphasize on four key knowledge gaps ranging across the whole lifecycle of agents. To show the agent's security issues, we summarize **100+** papers, where all existing attack surfaces and defenses are carefully categorized and explained. We believe that this survey may provide essential references for newcomers to this field and also inspire the development of more advanced security threats and defenses on LLM-based agents.

REFERENCES

- [1] Mahyar Abbasian, Iman Azimi, Amir M Rahmani, and Ramesh Jain. 2023. Conversational health agents: A personalized llm-powered agent framework. *arXiv* (2023).
- [2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. GPT-4 Technical Report. *arXiv* (2023).
- [3] Divyansh Agarwal, Alexander R Fabbri, Philippe Laban, Shafiq Joty, Caiming Xiong, and Chien-Sheng Wu. 2024. Investigating the prompt leakage effect and black-box defenses for multi-turn LLM interactions. *arXiv* (2024).
- [4] Jacob Andreas. 2022. Language models as agent models. *arXiv* (2022).
- [5] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv* (2022).
- [6] Fu Bang. 2023. GPTCache: An open-source semantic cache for LLM applications enabling faster answers and cost savings. In *Workshop for Natural Language Processing Open Source Software*. 212–218.
- [7] Ying Bao, Wankun Gong, and Kaiwen Yang. 2023. A Literature Review of Human–AI Synergy in Decision Making: From the Perspective of Affordance Actualization Theory. *Systems* (2023).
- [8] Rishabh Bhardwaj and Soujanya Poria. 2023. Language model unalignment: Parametric red-teaming to expose hidden harms and biases. *arXiv* (2023).
- [9] Shikha Bordia and Samuel R. Bowman. 2019. Identifying and Reducing Gender Bias in Word-Level Language Models. In *Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*.
- [10] Rodney Brooks. 1986. A robust layered control system for a mobile robot. *IEEE journal on robotics and automation* (1986).
- [11] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *NeurIPS*.
- [12] Fredrik Carlsson, Joey Öhman, Fangyu Liu, Severine Verlinden, Joakim Nivre, and Magnus Sahlgren. 2022. Fine-grained controllable text generation using non-residual prompting. In *Annual Meeting of the Association for Computational Linguistics*.
- [13] Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. 2023. Grounding large language models in interactive environments with online reinforcement learning. In *ICML*.
- [14] Chun Fai Chan, Daniel Wankit Yip, and Aysan Esmradi. 2023. Detection and Defense Against Prominent Attacks on Preconditioned LLM-Integrated Virtual Assistants. In *IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*. IEEE, 1–5.
- [15] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. 2023. Jailbreaking black box large language models in twenty queries. *arXiv* (2023).
- [16] Dake Chen, Hanbin Wang, Yunhao Huo, Yuzhao Li, and Haoyang Zhang. 2023. Gamegpt: Multi-agent collaborative framework for game development. *arXiv* (2023).
- [17] Mengqi Chen, Bin Guo, Hao Wang, Haoyu Li, Qian Zhao, Jingqi Liu, Yasan Ding, Yan Pan, and Zhiwen Yu. 2024. The Future of Cognitive Strategy-enhanced Persuasive Dialogue Agents: New Perspectives and Trends. *arXiv* (2024).
- [18] Tianlong Chen, Zhenyu Zhang, Yihua Zhang, Shiyu Chang, Sijia Liu, and Zhangyang Wang. 2022. Quarantine: Sparsity can uncover the trojan attack trigger for free. In *CVPR*.
- [19] Yongchao Chen, Jacob Arkin, Yang Zhang, Nicholas Roy, and Chuchu Fan. 2023. Scalable multi-robot collaboration with large language models: Centralized or decentralized systems? *arXiv* (2023).
- [20] Yihan Chen, Benfeng Xu, Quan Wang, Yi Liu, and Zhendong Mao. 2024. Benchmarking large language models on controllable generation under diversified instructions. *arXiv* (2024).
- [21] Steffi Chern, Zhen Fan, and Andy Liu. 2024. Combating Adversarial Attacks with Multi-Agent Debate. *arXiv* (2024).
- [22] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *NeurIPS* 30.
- [23] Stav Cohen, Ron Bitton, and Ben Nassi. 2024. Here Comes The AI Worm: Unleashing Zero-click Worms that Target GenAI-Powered Applications. *arXiv* (2024).
- [24] Maxwell Crouse, Ibrahim Abdelaziz, Kinjal Basu, Soham Dan, Sadhana Kumaravel, Achille Fokoue, Pavan Kapanipathi, and Luis Lastras. 2023. Formally specifying the high-level behavior of LLM-based agents. *arXiv* (2023).
- [25] Tianyu Cui, Yanling Wang, Chuanpu Fu, Yong Xiao, Sijia Li, Xinhao Deng, Yunpeng Liu, Qinglin Zhang, Ziyi Qiu, Peiyang Li, et al. 2024. Risk taxonomy, mitigation, and assessment benchmarks of large language model systems. *arXiv* (2024).
- [26] Lavina Daryanani. 2023. How to jailbreak chatgpt. <https://watcher.guru/news/how-to-jailbreak-chatgpt>. (2023).
- [27] Luigi De Angelis, Francesco Baglivo, Guglielmo Arzilli, Gaetano Pierpaolo Privitera, Paolo Ferragina, Alberto Eugenio Tozzi, and Caterina Rizzo. 2023. ChatGPT and the rise of large language models: the new AI-driven infodemic threat in public health. *Frontiers in Public Health* (2023).

- [28] Jerry den Hartog, Nicola Zannone, et al. 2018. Security and privacy for innovative automotive applications: A survey. *Computer Communications* (2018).
- [29] Gelei Deng, Yi Liu, Yuekang Li, Kailong Wang, Ying Zhang, Zefeng Li, Haoyu Wang, Tianwei Zhang, and Yang Liu. 2023. Jailbreaker: Automated jailbreak across multiple large language model chatbots. *arXiv* (2023).
- [30] Ameet Deshpande, Vishvak Murahari, Tanmay Rajpurohit, Ashwin Kalyan, and Karthik Narasimhan. 2023. Toxicity in chatgpt: Analyzing persona-assigned language models. In *Findings of EMNLP*.
- [31] V. Dibia. 2023. Generative AI: Practical Steps to Reduce Hallucination and Improve Performance of Systems Built with Large Language Models. In *Designing with ML: How to Build Usable Machine Learning Applications*. Self-published on designingwithml.com. (2023).
- [32] Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. 2024. LongRoPE: Extending LLM Context Window Beyond 2 Million Tokens. *arXiv* (2024).
- [33] Bao Gia Doan, Ehsan Abbasnejad, and Damith C Ranasinghe. 2020. Februus: Input purification defense against trojan attacks on deep neural network systems. In *ACSAC*. 897–912.
- [34] Tian Dong, Guoxing Chen, Shaofeng Li, Minhui Xue, Rayne Holland, Yan Meng, Zhen Liu, and Haojin Zhu. 2023. Unleashing cheapfakes through trojan plugins of large language models. *arXiv* (2023).
- [35] Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. *arXiv* (2023).
- [36] Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. 2023. Guiding pretraining in reinforcement learning with large language models. In *ICML*. PMLR.
- [37] Nouha Dziri, Andrea Madotto, Osmar Zaiane, and Avishek Joey Bose. 2021. Neural Path Hunter: Reducing Hallucination in Dialogue Systems via Path Grounding. In *EMNLP*.
- [38] Eva Eigner and Thorsten Händler. 2024. Determinants of LLM-assisted Decision-Making. *arXiv* (2024).
- [39] Embrace The Red. 2023. ChatGPT plugins: Data exfiltration via images & cross plugin request forgery. <https://embracethered.com/blog/posts/2023/chatgpt-webpilot-data-exfil-via-markdown-injection/>.
- [40] Aysan Esmradi, Daniel Wankit Yip, and Chun Fai Chan. 2023. A comprehensive survey of attack techniques, implementation, and mitigation strategies in large language models. In *International Conference on Ubiquitous Security*.
- [41] Meta Fundamental AI Research Diplomacy Team (FAIR)[†], Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, et al. 2022. Human-level play in the game of Diplomacy by combining language models with strategic reasoning. *Science* (2022).
- [42] Andrea Fioraldi, Alessandro Mantovani, Dominik Maier, and Davide Balzarotti. 2023. Dissecting American Fuzzy Lop: A FuzzBench Evaluation. *ACM transactions on software engineering and methodology* (2023).
- [43] Isabel O Gallegos, Ryan A Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K Ahmed. 2023. Bias and fairness in large language models: A survey. *arXiv* (2023).
- [44] Chen Gao, Xiaochong Lan, Zhihong Lu, Jinzhu Mao, Jinghua Piao, Huandong Wang, Depeng Jin, and Yong Li. 2023. S3: Social-network Simulation System with Large Language Model-Empowered Agents. *arXiv* (2023).
- [45] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models. In *Findings of EMNLP*.
- [46] Jonas Geiping, Alex Stein, Manli Shu, Khalid Saifullah, Yuxin Wen, and Tom Goldstein. 2024. Coercing LLMs to do and reveal (almost) anything. *arXiv* (2024).
- [47] Mingyang Geng, Shangwen Wang, Dezun Dong, Haotian Wang, Ge Li, Zhi Jin, Xiaoguang Mao, and Xiangke Liao. 2024. Large language models are few-shot summarizers: Multi-intent comment generation via in-context learning. In *IEEE/ACM International Conference on Software Engineering*.
- [48] Judy Wawira Gichoya, Kaesha Thomas, Leo Anthony Celi, Nabile Safdar, Imon Banerjee, John D Banja, Laleh Seyyed-Kalantari, Hari Trivedi, and Saptarshi Purkayastha. 2023. AI pitfalls and what not to do: mitigating bias in AI. *The British Journal of Radiology* (2023).
- [49] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. Not what you’ve signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *ACM Workshop on Artificial Intelligence and Security*. 79–90.
- [50] Xiangming Gu, Xiaosen Zheng, Tianyu Pang, Chao Du, Qian Liu, Ye Wang, Jing Jiang, and Min Lin. 2024. Agent Smith: A Single Image Can Jailbreak One Million Multimodal LLM Agents Exponentially Fast. *arXiv* (2024).
- [51] Guardrails AI. 2024. Build AI powered applications with confidence. (2024). <https://www.guardrailsai.com/> Accessed: 2024-02-27.
- [52] Michael Guastalla, Yiyi Li, Arvin Hekmati, and Bhaskar Krishnamachari. 2023. Application of Large Language Models to DDoS Attack Detection. In *International Conference on Security and Privacy in Cyber-Physical Systems and Smart Vehicles*.

- [53] Maanak Gupta, CharanKumar Akiri, Kshitiz Aryal, Eli Parker, and Lopamudra Praharaj. 2023. From chatgpt to threatgpt: Impact of generative ai in cybersecurity and privacy. *IEEE Access* (2023).
- [54] Rui Hao, Linmei Hu, Weijian Qi, Qingliu Wu, Yirui Zhang, and Liqiang Nie. 2023. Chatllm network: More brains, more intelligence. *arXiv* (2023).
- [55] Peter Hase, Mona Diab, Asli Celikyilmaz, Xian Li, Zornitsa Kozareva, Veselin Stoyanov, Mohit Bansal, and Srinivasan Iyer. 2023. Methods for measuring, updating, and visualizing factual beliefs in language models. In *Conference of the European Chapter of the Association for Computational Linguistics*.
- [56] Kostas Hatalis, Despina Christou, Joshua Myers, Steven Jones, Keith Lambert, Adam Amos-Binks, Zohreh Dannenhauer, and Dustin Dannenhauer. 2023. Memory Matters: The Need to Improve Long-Term Memory in LLM-Agents. In *Proceedings of the AAAI Symposium Series*.
- [57] Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. 2024. Defending Against Indirect Prompt Injection Attacks With Spotlighting. *arXiv* (2024).
- [58] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. In *ICLR*.
- [59] Tamanna Hossain, Sunipa Dev, and Sameer Singh. 2023. MISGENDERED: Limits of Large Language Models in Understanding Pronouns. In *Annual Meeting of the Association for Computational Linguistics*.
- [60] Yuki Hou, Haruki Tamoto, and Homei Miyashita. 2024. "My agent understands me better": Integrating Dynamic Human-like Memory Recall and Consolidation in LLM-Based Agents. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*. 1–7.
- [61] <https://www.theguardian.com/profile/hibaq-farah>. 2024. UK cybersecurity agency warns of chatbot ‘prompt injection’ attacks — theguardian.com. <https://www.theguardian.com/technology/2023/aug/30/uk-cybersecurity-agency-warns-of-chatbot-prompt-injection-attacks>.
- [62] Bin Hu, Chenyang Zhao, Pu Zhang, Zihao Zhou, Yuanhang Yang, Zenglin Xu, and Bin Liu. 2023. Enabling Intelligent Interactions between an Agent and an LLM: A Reinforcement Learning Approach. (2023).
- [63] Wenyue Hua, Xianjun Yang, Zelong Li, Cheng Wei, and Yongfeng Zhang. 2024. TrustAgent: Towards Safe and Trustworthy LLM-based Agents through Agent Constitution. *arXiv* (2024).
- [64] Xiaowei Huang, Wenjie Ruan, Wei Huang, Gaojie Jin, Yi Dong, Changshun Wu, Saddek Bensalem, Ronghui Mu, Yi Qi, Xingyu Zhao, et al. 2023. A survey of safety and trustworthiness of large language models through the lens of verification and validation. *arXiv* (2023).
- [65] Xijie Huang, Li Lina Zhang, Kwang-Ting Cheng, and Mao Yang. 2023. Boosting LLM Reasoning: Push the Limits of Few-shot Learning with Reinforced In-Context Pruning. *arXiv* (2023).
- [66] Yue Huang, Qihui Zhang, Lichao Sun, et al. 2023. Trustgpt: A benchmark for trustworthy and responsible large language models. *arXiv* (2023).
- [67] S Humeau, K Shuster, M Lachaux, and J Weston. 2020. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. *arXiv*. In *ICLR*.
- [68] Daphne Ippolito, Florian Tramèr, Milad Nasr, Chiyuan Zhang, Matthew Jagielski, Katherine Lee, Christopher A Choquette-Choo, and Nicholas Carlini. 2023. Preventing generation of verbatim memorization in language models gives a false sense of privacy. In *International Natural Language Generation Conference*.
- [69] Gautier Izacard and Edouard Grave. 2021. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Association for Computational Linguistics.
- [70] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *Comput. Surveys* (2023).
- [71] Zhenlan Ji, Daoyuan Wu, Pingchuan Ma, Zongjie Li, and Shuai Wang. 2024. Testing and Understanding Erroneous Planning in LLM Agents through Synthesized User Inputs. *arXiv* (2024).
- [72] Fengqing Jiang, Zhangchen Xu, Luyao Niu, Boxin Wang, Jinyuan Jia, Bo Li, and Radha Poovendran. 2023. Identifying and Mitigating Vulnerabilities in LLM-Integrated Applications. In *ICLR*.
- [73] Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *ICML*.
- [74] Mintong Kang, Nezihe Merve Gürel, Ning Yu, Dawn Song, and Bo Li. 2024. C-RAG: Certified Generation Risks for Retrieval-Augmented Language Models. *arXiv* (2024).
- [75] Changyeon Kim, Younggyo Seo, Hao Liu, Lisa Lee, Jinwoo Shin, Honglak Lee, and Kimin Lee. 2024. Guide Your Agent with Adaptive Multimodal Rewards. *NeurIPS* 36.
- [76] Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. 2020. Universal litmus patterns: Revealing backdoor attacks in cnns. In *CVPR*. 301–310.

- [77] Aounon Kumar, Chirag Agarwal, Suraj Srinivas, Soheil Feizi, and Hima Lakkaraju. 2024. Certifying llm safety against adversarial prompting. *arXiv* (2024).
- [78] Keita Kurita, Paul Michel, and Graham Neubig. 2020. Weight poisoning attacks on pre-trained models. *arXiv* (2020).
- [79] Nayeon Lee, Wei Ping, Peng Xu, Mostofa Patwary, Pascale N Fung, Mohammad Shoenybi, and Bryan Catanzaro. 2022. Factuality enhanced language models for open-ended text generation. *NeurIPS* 35.
- [80] Patrick Levi and Christoph P Neumann. 2024. Vocabulary Attack to Hijack Large Language Model Applications. *arXiv* (2024).
- [81] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*.
- [82] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society. In *NeurIPS*.
- [83] Haoran Li, Dadi Guo, Wei Fan, Mingshi Xu, Jie Huang, Fanpu Meng, and Yangqiu Song. 2023. Multi-step Jailbreaking Privacy Attacks on ChatGPT. In *Findings of EMNLP*.
- [84] Haoran Li, Mingshi Xu, and Yangqiu Song. 2023. Sentence Embedding Leaks More Information than You Expect: Generative Embedding Inversion Attack to Recover the Whole Sentence. In *Findings of ACL*.
- [85] Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023. Halueval: A large-scale hallucination evaluation benchmark for large language models. In *EMNLP*.
- [86] Jinfeng Li, Tianyu Du, Shouling Ji, Rong Zhang, Quan Lu, Min Yang, and Ting Wang. 2020. {TextShield}: Robust text classification based on multimodal embedding and neural machine translation. In *USENIX Security*.
- [87] Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, et al. 2024. Personal llm agents: Insights and survey about the capability, efficiency and security. *arXiv* (2024).
- [88] Zelong Li, Wenyue Hua, Hao Wang, He Zhu, and Yongfeng Zhang. 2024. Formal-LLM: Integrating Formal Language and Natural Language for Controllable LLM-based Agents. *arXiv* (2024).
- [89] Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. 2023. Encouraging Divergent Thinking in Large Language Models through Multi-Agent Debate. *arXiv* (2023).
- [90] Jonathan Light, Min Cai, Sheng Shen, and Ziniu Hu. 2023. AvalonBench: Evaluating LLMs Playing the Game of Avalon. In *NeurIPS Workshop*.
- [91] Baihan Lin, Djallel Bouneffouf, Guillermo Cecchi, and Kush R Varshney. 2023. Towards healthy AI: large language models need therapists too. *arXiv* (2023).
- [92] Bill Yuchen Lin, Yicheng Fu, Karina Yang, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Prithviraj Ammanabrolu, Yejin Choi, and Xiang Ren. 2024. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. *NeurIPS* 36.
- [93] Jiaju Lin, Haoran Zhao, Aochi Zhang, Yiting Wu, Huqiyue Ping, and Qin Chen. 2023. Agentsims: An open-source sandbox for large language model evaluation. *arXiv* (2023).
- [94] Aishan Liu, Tairan Huang, Xianglong Liu, Yitao Xu, Yuqing Ma, Xinyun Chen, Stephen J Maybank, and Dacheng Tao. 2020. Spatiotemporal attacks for embodied agents. In *ECCV*.
- [95] Sheng Liu, Lei Xing, and James Zou. 2023. In-context vectors: Making in context learning more effective and controllable through latent space steering. *arXiv* (2023).
- [96] Tong Liu, Zizhuang Deng, Guozhu Meng, Yuekang Li, and Kai Chen. 2023. Demystifying rce vulnerabilities in llm-integrated apps. *arXiv* (2023).
- [97] Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, and Yang Liu. 2023. Prompt Injection attack against LLM-integrated Applications. *arXiv* (2023).
- [98] Yi Liu, Gelei Deng, Zhengzi Xu, Yuekang Li, Yaowen Zheng, Ying Zhang, Lida Zhao, Tianwei Zhang, and Yang Liu. 2023. Jailbreaking chatgpt via prompt engineering: An empirical study. *arXiv* (2023).
- [99] Yang Liu, Yuanshun Yao, Jean-Francois Ton, Xiaoying Zhang, Ruocheng Guo Hao Cheng, Yegor Klochkov, Muhammad Faaiz Taufiq, and Hang Li. 2023. Trustworthy LLMs: a Survey and Guideline for Evaluating Large Language Models' Alignment. *arXiv* (2023).
- [100] Qinghua Lu, Liming Zhu, Xiwei Xu, Zhenchang Xing, Stefan Harrer, and Jon Whittle. 2023. Building the Future of Responsible AI: A Reference Architecture for Designing Large Language Model based Agents. *arXiv* (2023).
- [101] Ximing Lu, Sean Welleck, Jack Hessel, Liwei Jiang, Lianhui Qin, Peter West, Prithviraj Ammanabrolu, and Yejin Choi. 2022. Quark: Controllable text generation with reinforced unlearning. *NeurIPS* 35, 27591–27609.
- [102] Chris Van Pelt Lukas Biewald. 2017. Weights and Biases. <https://wandb.ai/>. (2017).
- [103] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *NeurIPS* 36.

- [104] Zhao Mandi, Shreeya Jain, and Shuran Song. 2023. Roco: Dialectic multi-robot collaboration with large language models. *arXiv* (2023).
- [105] Tula Masterman, Sandi Besen, Mason Sawtell, and Alex Chao. 2024. The Landscape of Emerging AI Agent Architectures for Reasoning, Planning, and Tool Calling: A Survey. *arXiv* (2024).
- [106] Nikhil Mehta, Milagro Teruel, Xin Deng, Sergio Figueroa Sanz, Ahmed Awadallah, and Julia Kiseleva. 2024. Improving Grounded Language Understanding in a Collaborative Environment by Interacting with Agents Through Help Feedback. In *Findings of EACL*.
- [107] Kai Mei, Zelong Li, Shuyuan Xu, Ruosong Ye, Yingqiang Ge, and Yongfeng Zhang. 2024. AIOS: LLM Agent Operating System. *arXiv* (2024).
- [108] Vincent Micheli, Eloi Alonso, and François Fleuret. 2023. Transformers are sample-efficient world models. In *ICLR*.
- [109] Sewon Min, Kalpesh Krishna, Xinxu Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation. In *EMNLP*.
- [110] Lingbo Mo, Zeyi Liao, Boyuan Zheng, Yu Su, Chaowei Xiao, and Huan Sun. 2024. A Trembling House of Cards? Mapping Adversarial Attacks against Language Agents. *arXiv* (2024).
- [111] John Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander Rush. 2023. Text Embeddings Reveal (Almost) As Much As Text. In *EMNLP*.
- [112] Stephen Moskal, Sam Lane, Erik Hemberg, and Una-May O'Reilly. 2023. LLMs Killed the Script Kiddie: How Agents Supported by Large Language Models Change the Landscape of Network Threat Testing. *arXiv* (2023).
- [113] Sumeet Ramesh Motwani, Mikhail Baranchuk, Lewis Hammond, and Christian Schroeder de Witt. 2023. A Perfect Collusion Benchmark: How can AI agents be prevented from colluding with information-theoretic undetectability?. In *NeurIPS workshop*.
- [114] Hichem Mrabet, Sana Belguith, Adeeb Alhomoud, and Abderrazak Jemai. 2020. A survey of IoT security based on a layered architecture of sensing and data analysis. *Sensors* (2020).
- [115] Dor Muhlgay, Ori Ram, Inbal Magar, Yoav Levine, Nir Ratner, Yonatan Belinkov, Omri Abend, Kevin Leyton-Brown, Amnon Shashua, and Yoav Shoham. 2024. Generating Benchmarks for Factuality Evaluation of Language Models. In *Conference of the European Chapter of the Association for Computational Linguistics*.
- [116] Uttam Mukhopadhyay, Larry M Stephens, Michael N Huhns, and Ronald D Bonnell. 1986. An intelligent system for document retrieval in distributed office environments. *Journal of the American Society for Information Science* (1986).
- [117] Varun Nair, Elliot Schumacher, Geoffrey Tso, and Anitha Kannan. 2023. DERA: enhancing large language model completions with dialog-enabled resolving agents. *arXiv* (2023).
- [118] Tai Nguyen and Eric Wong. 2023. In-context example selection with influences. *arXiv* (2023).
- [119] Aidan O'Gara. 2023. Hoodwinked: Deception and cooperation in a text-based game for language models. *arXiv* (2023).
- [120] Nedjma Ousidhoum, Xinran Zhao, Tianqing Fang, Yangqiu Song, and Dit-Yan Yeung. 2021. Probing toxic content in large pre-trained language models. In *International Joint Conference on Natural Language Processing*.
- [121] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *NeurIPS*.
- [122] James Jie Pan, Jianguo Wang, and Guoliang Li. 2023. Survey of vector database management systems. *arXiv* (2023).
- [123] Yikang Pan, Liangming Pan, Wenhui Chen, Preslav Nakov, Min-Yen Kan, and William Yang Wang. 2023. On the risk of misinformation pollution with large language models. *arXiv* (2023).
- [124] Jing-Cheng Pang, Xin-Yu Yang, Si-Hang Yang, and Yang Yu. 2023. Natural Language-conditioned Reinforcement Learning with Inside-out Task Language Development and Translation. *NeurIPS*.
- [125] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Annual ACM Symposium on User Interface Software and Technology*. 1–22.
- [126] Peter S Park, Simon Goldstein, Aidan O'Gara, Michael Chen, and Dan Hendrycks. 2023. AI deception: A survey of examples, risks, and potential solutions. *arXiv* (2023).
- [127] Rodrigo Pedro, Daniel Castro, Paulo Carreira, and Nuno Santos. 2023. From Prompt Injections to SQL Injection Attacks: How Protected is Your LLM-Integrated Web Application? *arXiv* (2023).
- [128] Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv* (2023).
- [129] Ethan Perez, Sam Ringer, Kamile Lukosiute, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, Andy Jones, Anna Chen, Benjamin Mann, Brian Israel, Bryan Seethor, Cameron McKinnon, Christopher Olah, Da Yan, Daniela Amodei, Dario Amodei, Dawn Drain, Dustin Li, Eli Tran-Johnson,

- Guro Khundadze, Jackson Kernion, James Landis, Jamie Kerr, Jared Mueller, Jeeyoon Hyun, Joshua Landau, Kamal Ndousse, Landon Goldberg, Liane Lovitt, Martin Lucas, Michael Sellitto, Miranda Zhang, Neerav Kingsland, Nelson Elhage, Nicholas Joseph, Noemi Mercado, Nova DasSarma, Oliver Rausch, Robin Larson, Sam McCandlish, Scott Johnston, Shauna Kravec, Sheer El Showk, Tamera Lanham, Timothy Telleen-Lawton, Tom Brown, Tom Henighan, Tristan Hume, Yuntao Bai, Zac Hatfield-Dodds, Jack Clark, Samuel R. Bowman, Amanda Askell, Roger Grosse, Danny Hernandez, Deep Ganguli, Evan Hubinger, Nicholas Schiefer, and Jared Kaplan. 2023. Discovering Language Model Behaviors with Model-Written Evaluations. In *Findings of ACL*.
- [130] Fábio Perez and Ian Ribeiro. 2022. Ignore previous prompt: Attack techniques for language models. *NeurIPS 2022*.
- [131] Steve Phelps and Rebecca Ranson. 2023. Of Models and Tin Men—a behavioural economics study of principal-agent problems in AI alignment using large-language models. *arXiv* (2023).
- [132] Lukas Pöhler, Valentin Schrader, Alexander Ladwein, and Florian von Keller. 2024. A Technological Perspective on Misuse of Available AI. *arXiv* (2024).
- [133] Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. 2023. Communicative agents for software development. *arXiv* (2023).
- [134] Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training Gopher. *arXiv* (2021).
- [135] Fathima Abdul Rahman and Guang Lu. 2023. A Contextualized Real-Time Multimodal Emotion Recognition for Conversational Agents using Graph Convolutional Networks in Reinforcement Learning. *arXiv* (2023).
- [136] Leonardo Ranaldi and Giulia Pucci. 2023. When Large Language Models contradict humans? Large Language Models’ Sycophantic Behaviour. *arXiv* (2023).
- [137] Zeeshan Rasheed, Muhammad Waseem, Kari Systä, and Pekka Abrahamsson. 2024. Large language model evaluation via multi ai agents: Preliminary results. *arXiv* (2024).
- [138] Traian Rebedea, Razvan Dinu, Makesh Narsimhan Sreedhar, Christopher Parisien, and Jonathan Cohen. 2023. NeMo Guardrails: A Toolkit for Controllable and Safe LLM Applications with Programmable Rails. In *EMNLP*.
- [139] Embrace The Red. 2023. Indirect Prompt Injection via YouTube Transcripts · Embrace The Red — embracethered.com. <https://embracethered.com/blog/posts/2023/chatgpt-plugin-youtube-indirect-prompt-injection/>.
- [140] Zohar Rimón, Tom Jurgenson, Orr Krupnik, Gilad Adler, and Aviv Tamar. 2024. MAMBA: an Effective World Model Approach for Meta-Reinforcement Learning. In *ICLR*.
- [141] Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. 2024. Identifying the Risks of LM Agents with an LM-Emulated Sandbox. In *ICLR*.
- [142] Ahmed Salem, Andrew Paverd, and Boris Köpf. 2023. Maatphor: Automated variant analysis for prompt injection attacks. *arXiv* (2023).
- [143] Sergei Savvov. 2023. Fixing Hallucinations in LLMs <https://betterprogramming.pub/fixing-hallucinations-in-llms-9ff0fd438e33>. (2023).
- [144] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2024. Toolformer: Language models can teach themselves to use tools. *NeurIPS*.
- [145] Leo Schwinn, David Dobre, Stephan Günnemann, and Gauthier Gidel. 2023. Adversarial attacks and defenses in large language models: Old and new threats. *arXiv* (2023).
- [146] Omar Shaikh, Hongxin Zhang, William Held, Michael Bernstein, and Diyi Yang. 2023. On Second Thought, Let’s Not Think Step by Step! Bias and Toxicity in Zero-Shot Reasoning. In *Annual Meeting of the Association for Computational Linguistics*.
- [147] Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R Johnston, et al. 2023. Towards understanding sycophancy in language models. *arXiv* (2023).
- [148] Erfan Shayegani, Md Abdullah Al Mamun, Yu Fu, Pedram Zaree, Yue Dong, and Nael Abu-Ghazaleh. 2023. Survey of vulnerabilities in large language models revealed by adversarial attacks. *arXiv* (2023).
- [149] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2024. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *NeurIPS*.
- [150] Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval Augmentation Reduces Hallucination in Conversation. In *Findings of EMNLP*.
- [151] Emily H Soice, Rafael Rocha, Kimberlee Cordova, Michael Specter, and Kevin M Esvelt. 2023. Can large language models democratize access to dual-use biotechnology? *arXiv* (2023).
- [152] Congzheng Song and Ananth Raghunathan. 2020. Information leakage in embedding models. In *ACM SIGSAC conference on computer and communications security*. 377–390.
- [153] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2023. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *CVPR*. 2998–3009.

- [154] Shyam Sudhakaran, Miguel González-Duque, Matthias Freiberger, Claire Glanois, Elias Najarro, and Sebastian Risi. 2024. Mariogpt: Open-ended text2level generation through large language models. *NeurIPS*.
- [155] Weiwei Sun, Zhengliang Shi, Shen Gao, Pengjie Ren, Maarten de Rijke, and Zhaochun Ren. 2023. Contrastive learning reduces hallucination in conversations. In *AAAI*.
- [156] Yuxiang Sun, Checheng Yu, Junjie Zhao, Wei Wang, and Xianzhong Zhou. 2023. Self Generated Wargame AI: Double Layer Agent Task Planning Based on Large Language Model. *arXiv* (2023).
- [157] Harini Suresh and John V Guttag. 2019. A framework for understanding unintended consequences of machine learning. *arXiv* (2019).
- [158] Gaurav Suri, Lily R Slater, Ali Ziaee, and Morgan Nguyen. 2024. Do large language models show decision heuristics similar to humans? A case study using GPT-3.5. *Journal of Experimental Psychology: General* (2024).
- [159] Weihao Tan, Wentao Zhang, Shanqi Liu, Longtao Zheng, Xinrun Wang, and Bo An. 2024. True Knowledge Comes from Practice: Aligning Large Language Models with Embodied Environments via Reinforcement Learning. In *ICLR*.
- [160] Xiangru Tang, Qiao Jin, Kunlun Zhu, Tongxin Yuan, Yichi Zhang, Wangchunshu Zhou, Meng Qu, Yilun Zhao, Jian Tang, Zhuosheng Zhang, et al. 2024. Prioritizing Safeguarding Over Autonomy: Risks of LLM Agents for Science. *arXiv* (2024).
- [161] Yu Tian, Xiao Yang, Jingyuan Zhang, Yinpeng Dong, and Hang Su. 2023. Evil geniuses: Delving into the safety of llm-based agents. *arXiv* (2023).
- [162] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv* (2023).
- [163] Sam Toyer, Olivia Watkins, Ethan Adrian Mendes, Justin Svegliato, Luke Bailey, Tiffany Wang, Isaac Ong, Karim Elmaaroufi, Pieter Abbeel, Trevor Darrell, Alan Ritter, and Stuart Russell. 2024. Tensor Trust: Interpretable Prompt Injection Attacks from an Online Game. In *ICLR*.
- [164] Dennis Ulmer, Elman Mansimov, Kaixiang Lin, Justin Sun, Xibin Gao, and Yi Zhang. 2024. Bootstrapping llm-based task-oriented dialogue agents via self-talk. *arXiv* (2024).
- [165] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *NeurIPS*.
- [166] Bertie Vidgen, Adarsh Agrawal, Ahmed M Ahmed, Victor Akinwande, Namir Al-Nuaimi, Najla Alfaraj, Elie Alhajjar, Lora Aroyo, Rupri Bavalatti, Borhane Bili-Hamelin, et al. 2024. Introducing v0. 5 of the AI Safety Benchmark from MLCommons. *arXiv* (2024).
- [167] Celine Wald and Lukas Pfahler. 2023. Exposing bias in online communities through large-scale language models. *arXiv* (2023).
- [168] Eric Wallace, Kai Xiao, Reimar Leike, Lilian Weng, Johannes Heidecke, and Alex Beutel. 2024. The Instruction Hierarchy: Training LLMs to Prioritize Privileged Instructions. *arXiv* (2024).
- [169] Alexander Wan, Eric Wallace, Sheng Shen, and Dan Klein. 2023. Poisoning language models during instruction tuning. In *ICML*.
- [170] Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, Sang T. Truong, Simran Arora, Mantas Mazeika, Dan Hendrycks, Zinan Lin, Yu Cheng, Sanmi Koyejo, Dawn Song, and Bo Li. 2023. DecodingTrust: A Comprehensive Assessment of Trustworthiness in GPT Models. In *NeurIPS*.
- [171] Yuntao Wang, Yanghe Pan, Miao Yan, Zhou Su, and Tom H Luan. 2023. A survey on ChatGPT: AI-generated contents, challenges, and solutions. *IEEE Open Journal of the Computer Society* (2023).
- [172] Yau-Shian Wang and Yingshan Chang. 2022. Toxicity detection with generative prompt-based inference. *arXiv* (2022).
- [173] Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Shawn Ma, and Yitao Liang. 2024. Describe, explain, plan and select: interactive planning with LLMs enables open-world multi-task agents. *NeurIPS* 36.
- [174] Connor Weeks, Aravind Cheruvu, Sifat Muhammad Abdullah, Shravya Kanchi, Daphne Yao, and Bimal Viswanath. 2023. A first look at toxicity injection attacks on open-domain chatbots. In *Annual Computer Security Applications Conference*.
- [175] Jerry Wei, Da Huang, Yifeng Lu, Denny Zhou, and Quoc V Le. 2023. Simple synthetic data reduces sycophancy in large language models. *arXiv* (2023).
- [176] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS* 35, 24824–24837.
- [177] Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Hu, Dustin Tran, Daiyi Peng, RuiBo Liu, Da Huang, Cosmo Du, et al. 2024. Long-form factuality in large language models. *arXiv* (2024).
- [178] Zeming Wei, Yifei Wang, and Yisen Wang. 2023. Jailbreak and guard aligned language models with only few in-context demonstrations. *arXiv* (2023).

- [179] Roy Weiss, Daniel Ayzenshteyn, Guy Amit, and Yisroel Mirsky. 2024. What Was Your Prompt? A Remote Keylogging Attack on AI Assistants. *arXiv* (2024).
- [180] Johannes Welbl, Amelia Glaese, Jonathan Uesato, Sumanth Dathathri, John Mellor, Lisa Anne Hendricks, Kirsty Anderson, Pushmeet Kohli, Ben Coppin, and Po-Sen Huang. 2021. Challenges in Detoxifying Language Models. In *Findings of EMNLP*.
- [181] David Windridge, Henrik Svensson, and Serge Thill. 2021. On the utility of dreaming: A general model for how learning in artificial agents can benefit from data hallucination. *Adaptive Behavior* (2021).
- [182] Yotam Wolf, Noam Wies, Yoav Levine, and Amnon Shashua. 2023. Fundamental limitations of alignment in large language models. *arXiv* (2023).
- [183] Michael Wooldridge and Nicholas R Jennings. 1995. Intelligent agents: Theory and practice. *The knowledge engineering review* (1995).
- [184] Fangzhou Wu, Shutong Wu, Yulong Cao, and Chaowei Xiao. 2024. WIPI: A New Web Threat for LLM-Driven Web Agents. *arXiv* (2024).
- [185] Fangzhou Wu, Ning Zhang, Somesh Jha, Patrick McDaniel, and Chaowei Xiao. 2024. A New Era in LLM Security: Exploring Security Concerns in Real-World LLM-based Systems. *arXiv* (2024).
- [186] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2023. The rise and potential of large language model based agents: A survey. *arXiv* (2023).
- [187] Xingrui Xie, Han Liu, Wenzhe Hou, and Hongbin Huang. 2023. A Brief Survey of Vector Databases. In *International Conference on Big Data and Information Analytics (BigDIA)*. IEEE.
- [188] Frank Xing. 2024. Designing Heterogeneous LLM Agents for Financial Sentiment Analysis. *arXiv* (2024).
- [189] Yuzhuang Xu, Shuo Wang, Peng Li, Fuwen Luo, Xiaolong Wang, Weidong Liu, and Yang Liu. 2023. Exploring large language models for communication games: An empirical study on werewolf. *arXiv* (2023).
- [190] Zelai Xu, Chao Yu, Fei Fang, Yu Wang, and Yi Wu. 2023. Language agents with reinforcement learning for strategic play in the werewolf game. *arXiv* (2023).
- [191] Xue Yan, Yan Song, Xinyu Cui, Filippos Christianos, Haifeng Zhang, David Henry Mguni, and Jun Wang. 2023. Ask more, know better: Reinforce-Learned Prompt Questions for Decision Making with Large Language Models. *arXiv* (2023).
- [192] Wenkai Yang, Xiaohan Bi, Yankai Lin, Sishuo Chen, Jie Zhou, and Xu Sun. 2024. Watch Out for Your Agents! Investigating Backdoor Threats to LLM-Based Agents. *arXiv* (2024).
- [193] Yong Yang, Xuhong Zhang, Yi Jiang, Xi Chen, Haoyu Wang, Shouling Ji, and Zonghui Wang. 2024. PRSA: Prompt Reverse Stealing Attacks against Large Language Models. *arXiv* (2024).
- [194] Dongyu Yao, Jianshu Zhang, Ian G Harris, and Marcel Carlsson. 2024. Fuzzllm: A novel and universal fuzzing framework for proactively discovering jailbreak vulnerabilities in large language models. In *ICASSP*.
- [195] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *ICLR*.
- [196] Jingwei Yi, Yueqi Xie, Bin Zhu, Keegan Hines, Emre Kiciman, Guangzhong Sun, Xing Xie, and Fangzhao Wu. 2023. Benchmarking and defending against indirect prompt injection attacks on large language models. *arXiv* (2023).
- [197] Jiahao Yu, Xingwei Lin, and Xinyu Xing. 2023. Gptfuzzer: Red teaming large language models with auto-generated jailbreak prompts. *arXiv* (2023).
- [198] Tongxin Yuan, Zhiwei He, Lingzhong Dong, Yiming Wang, Ruijie Zhao, Tian Xia, Lizhen Xu, Binglin Zhou, Fangqi Li, Zhuosheng Zhang, et al. 2024. R-Judge: Benchmarking Safety Risk Awareness for LLM Agents. In *ICLR*.
- [199] Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen tse Huang, Pinjia He, Shuming Shi, and Zhaopeng Tu. 2024. GPT-4 Is Too Smart To Be Safe: Stealthy Chat with LLMs via Cipher. In *ICLR*. <https://openreview.net/forum?id=MbfAK4s61A>
- [200] Xiang Yue, Boshi Wang, Ziru Chen, Kai Zhang, Yu Su, and Huan Sun. 2023. Automatic Evaluation of Attribution by Large Language Models. In *EMNLP*.
- [201] Matei Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, et al. 2018. Accelerating the machine learning lifecycle with MLflow. *IEEE Data Eng. Bull.* (2018).
- [202] Shenglai Zeng, Jiankun Zhang, Pengfei He, Yue Xing, Yiding Liu, Han Xu, Jie Ren, Shuaiqiang Wang, Dawei Yin, Yi Chang, et al. 2024. The Good and The Bad: Exploring Privacy Issues in Retrieval-Augmented Generation (RAG). *arXiv* (2024).
- [203] Yifan Zeng, Yiran Wu, Xiao Zhang, Huazheng Wang, and Qingyun Wu. 2024. AutoDefense: Multi-Agent LLM Defense against Jailbreak Attacks. *arXiv* (2024).
- [204] Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. 2024. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. *arXiv* (2024).
- [205] Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua Tenenbaum, Tianmin Shu, and Chuang Gan. 2023. Building Cooperative Embodied Agents Modularly with Large Language Models. In *NeurIPS Workshop*.

- [206] Wanpeng Zhang and Zongqing Lu. 2023. Rladapter: Bridging large language models to reinforcement learning in open worlds. *arXiv* (2023).
- [207] Xinyu Zhang, Huiyu Xu, Zhongjie Ba, Zhibo Wang, Yuan Hong, Jian Liu, Zhan Qin, and Kui Ren. 2024. Privacyasst: Safeguarding user privacy in tool-using large language model agents. *IEEE Transactions on Dependable and Secure Computing* (2024).
- [208] Yiming Zhang, Nicholas Carlini, and Daphne Ippolito. 2024. Effective Prompt Extraction from Language Models. *arXiv* (2024).
- [209] Yuqi Zhang, Liang Ding, Lefei Zhang, and Dacheng Tao. 2024. Intention analysis prompting makes large language models a good jailbreak defender. *arXiv* (2024).
- [210] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023. Siren’s song in the AI ocean: A survey on hallucination in large language models. *arXiv* (2023).
- [211] Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2024. A Survey on the Memory Mechanism of Large Language Model based Agents. *arXiv* (2024).
- [212] Zhexin Zhang, Junxiao Yang, Pei Ke, and Minlie Huang. 2023. Defending large language models against jailbreaking attacks through goal prioritization. *arXiv* (2023).
- [213] Qinlin Zhao, Jindong Wang, Yixuan Zhang, Yiqiao Jin, Kaijie Zhu, Hao Chen, and Xing Xie. 2023. Competeai: Understanding the competition behaviors in large language model-based agents. *arXiv* (2023).
- [214] Yiyang Zhou, Chenhang Cui, Jaehong Yoon, Linjun Zhang, Zhun Deng, Chelsea Finn, Mohit Bansal, and Huaxiu Yao. 2024. Analyzing and Mitigating Object Hallucination in Large Vision-Language Models. In *ICLR*.
- [215] Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. 2024. PoisonedRAG: Knowledge Poisoning Attacks to Retrieval-Augmented Generation of Large Language Models. *arXiv* (2024).