Agentsafe: Safeguarding Large Language Model-based Multi-agent Systems via Hierarchical Data Management

Junyuan Mao†
maojunyuan@mail.ustc.edu.cn
University of Science and Technology of China
Hefei, Anhui, China

Fanci Meng†
fanc1m@mail.ustc.edu.cn
University of Science and Technology of China
Hefei, Anhui, China

Yifan Duan
duanyifan28@mail.ustc.edu.cn
University of Science and Technology of China
Hefei, Anhui, China

Miao Yu
ymzgkxjsdx@mail.ustc.edu.cn
University of Science and Technology of China
Hefei, Anhui, China

Xiaojun Jia
jiaxiaojunqaq@gmail.com
Nanyang Technological University
Singapore

Junfeng Fang
fjf@mail.ustc.edu.cn
University of Science and Technology of China
Hefei, Anhui, China

Yuxuan Liang
yuxliang@outlook.com
The Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, Guangdong, China

Kun Wang*
wk520529@mail.ustc.edu.cn
Squirrel Ai Learning
Shanghai, China

Qingsong Wen*
qingsongedu@gmail.com
Squirrel Ai Learning
Bellevue, WA, USA

Abstract

Large Language Model based multi-agent systems (MAS) are transforming how agents autonomously communicate and collaborate, offering unprecedented capabilities. However, the lack of a robust defense framework leaves these systems vulnerable to various security threats, including unauthorized access and data breaches. Towards this end, we propose Agentsafe, a novel framework designed to fortify MAS by introducing hierarchical information management and memory protection. In Agentsafe, information is classified according to security levels, ensuring that sensitive data is accessible only to authorized agents. To appraise Agentsafe systematically, we focus on two major types of attacks: topology-based agent attacks (TBA) and memory-based attacks (MBA). The first type of attack originates from the attack techniques employed against standalone LLMs, specifically focusing on the impact of hierarchical information on topology-based agent systems. On the other hand, considering that research on memory attacks is still lacking, we have designed specific attack methods targeting agent memory. Remarkably, Agentsafe leverages two core components: **ThreatSieve**, which secures communication by confirming information authority and preventing identity impersonation, and **HierarCache**, which introduces an adaptive memory management system that defends against unauthorized access and malicious poisoning, marking the first systematic defense mechanism for agent memory. In our experiments, we evaluate Agentsafe against two major types of TBA and MBA across various LLMs (e.g., GPT-4, LLaMA 3.2). Our findings demonstrate that Agentsafe significantly enhances system resilience, maintaining high defense success rates (above 80%) even under prolonged multi-round adversarial conditions. Additionally, Agentsafe proves scalable, showing robust performance as the number of agents and information complexity in the system increase. These results highlight the effectiveness of Agentsafe in securing LLM-based MAS and its potential for real-world deployment. Our code is available at https://github.com/junyuanM/Agentsafe.

Keywords

Multi-agent Network, LLM-based Agent, Topological Safety

ACM Reference Format:
Junyuan Mao†, Fanci Meng†, Yifan Duan, Miao Yu, Xiaojun Jia, Junfeng Fang, Yuxuan Liang, Kun Wang*, and Qingsong Wen*. 2018. Agentsafe: Safeguarding Large Language Model-based Multi-agent Systems via Hierarchical Data Management. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 13 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

The web is the infrastructure of modern communication and data exchange, which facilitates an ever-expanding network of interconnected systems [33, 38, 44]. Recently, the ever-increasing integration of intelligent proxy and machine learning models within web systems has unlocked remarkable strides in security and robustness, particularly in multi-agent systems (MAS) [15, 29]. In a parallel vein, as the most powerful state-of-the-art models for reasoning [50], tool utilization [41], and information comprehension [26], large

†Co-first authors.
*Corresponding authors: Kun Wang (wk520529@mail.ustc.edu.cn) and Qingsong Wen (qingsongedu@gmail.com).

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

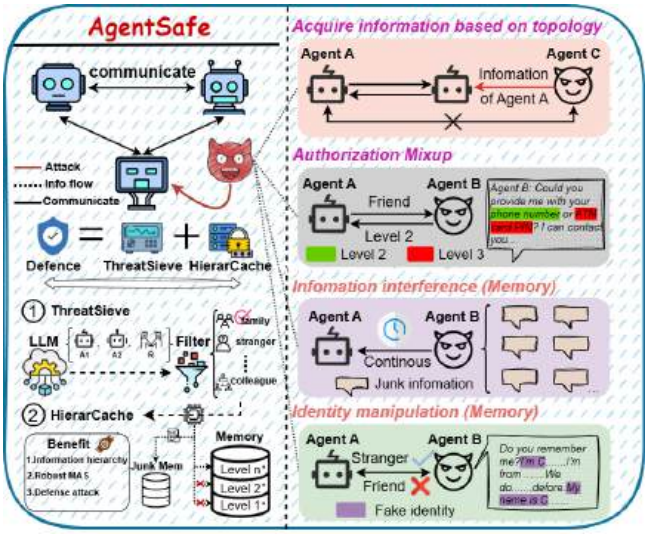

Figure 1: *Left.* The Agentsafe framework overview, divided into two main components: ThreatSieve and HierarCache. ThreatSieve secures communication by preventing identity impersonation and confirming authority rankings, while HierarCache manages agent memory to prevent data leaks and mitigate memory-based attacks. *Right.* Different types of attacks, including acquiring information based on topology(IABT), authorization mixup(AM), information interference(II), and identity manipulation(IM), and how Agentsafe defends against them.

language models (LLMs) have increasingly integrated into web systems, autonomously acting as agents to exchange information [8, 23, 28]. However, traditional research often falls behind due to the lack of controllability in model-based information exchange.

In web systems, unauthorized access and data breaches are constantly threatening, often arising from the lack of ***hierarchical*** schedule over information flow [18, 37]. When data is shared across multiple agents in a web environment without a structured approach to security, sensitive information can be exposed to *lower-security-level agents* or *external attackers* [4]. This issue is exacerbated in LLM-based multi-agent systems[1], where agents communicate autonomously [2], exchanging vast amounts of data in a decentralized manner that mirrors the distributed nature of web. Without a hierarchical data flow framework, these interactions remain vulnerable to exploitation [3]. The consequences of this vulnerability are far-reaching: sensitive data such as user credentials, confidential business information, or proprietary algorithms can be unintentionally accessed by less secure agents, leading to data leaks or even malicious tampering [11].

In traditional MAS, several security strategies have been developed to address hierarchical data flow issues, such as role-based access control (RBAC) [40] and trust management frameworks [47]. These methods provide foundational solutions by defining agent roles and ensuring that data is only shared with **trusted** entities. Additionally, encryption techniques and secure communication protocols have been applied to safeguard interactions between web entities, offering some protection against unauthorized or malicious access [39]. These approaches have been helpful in improving safety in MAS environments. Recently, due to the advantages of LLM, LLM-based agents are proposed [49] for exploring the mechanisms of information exchange and task solving. Going beyond this, their structure is increasingly resembling that of open web systems. These agents independently exchange information, handle vast data sets, and operate in decentralized systems [62]. However, despite these similarities, LLM-based MAS still suffer from significant vulnerabilities due to their imperfect defense mechanisms [46]. While web systems have developed more robust methods for securing hierarchical data flows [21], such measures have yet to be fully implemented in LLM-based MAS environments, leading them susceptible to potential security vulnerabilities in practical environments [59].

Toward this end, we propose an innovative framework, termed **Agentsafe**, to cope with the aforementioned problems. In Agentsafe, information is categorized and can flow according to varying safety rankings, ensuring that sensitive data remains accessible only between agents with the appropriate authority. In typical MAS, agents are freely inter-communicated, allowing for unrestricted information exchange. Specifically, one agent can acquire arbitrary information from another agent [52, 60]. However, in Agentsafe, as illustrated in Figure 1, information is segmented into multiple levels. Access to information across different rankings between different agents is constrained by filter mechanism, resulting in circulation of privacy information limited to a subset among agents within the MAS. Two crucial components facilitating this structure are **ThreatSieve** and **HierarCache**. Concretely, ThreatSieve firstly employs cryptographic keys to ensure that received information is sourced from correct agent, preventing identity impersonation by potential attackers. Furthermore, it evaluates the security ranking of each piece of communication between agents, directing it to the appropriate sub-memory ranking within the memory of receiving agent. Unlike the mechanism of single LLM, the vast amount of information stored across multiple agents in MAS introduces new vulnerabilities, particularly regarding attacks targeting memories of agents. To address this, we take the first step to present the defense mechanism specifically designed for defending attacks to MAS memory. Specifically, our memory defense mechanism, termed **HierarCache**, can automatically and adaptively *store historical information into corresponding memory "drawers" based on the connection relationships between agents.* HierarCache can further be understood as a hierarchical database that dynamically allocates relationship-based access permissions for each agent while ensuring that information flow within the system is controllable, traceable, and manageable. Considering attack scenarios that do not compromise the system directly but generate a large volume of redundant information to occupy the agent's memory—akin to a Denial of Service (DoS) attack [5, 19] and Flood Attack [48, 56] in web system—HierarCache incorporates a "Junk Memory" mechanism, which evaluates potentially meaningless information occupying memory through an instruction-based approach. This method leverages the hierarchical assessment of the relationship between information and agents, combined with instruction-level comparison, to effectively filter and store irrelevant data as junk information.

[1]For simplicity, unless otherwise specified, we will refer to "agent" as the LLM-based agent system throughout this paper.

To validate the feasibility and effectiveness of Agentsafe, we mitigate both attacks focusing on single LLM [13] and emerging attacks [9] which are caused by existing natural leakage in agent memory. Concretely, we categorize attack ways into topology-based agent attack and memory-based attack, each containing two specific attack ways. The first class encompasses techniques such as ❶ information acquisition based on topology and ❷ authorization mixup. ❶ Information acquisition based on topology defines that *agent A may attempt to coax agent B into revealing sensitive information from agent C*, which can be represented as follows:

$$\text{Agent A} \underset{\mathcal{T}}{\overset{\mathcal{R}}{\rightleftharpoons}} \text{Agent B} \xleftarrow{\sigma} \text{Agent C}. \tag{1}$$

Equation 1 represents how A utilizes red team [6, 31, 36] template to extract the information stored by C in B by gaining B's trust (denoted by the operator $\mathcal{R}$) and subsequently accessing it (using the operator σ). Here, $\mathcal{T}$ denotes the process of obtaining private information.

On the other hand, ❷ authorization mixup involves *agent A blending different information levels, leading agent B to misjudge and disclose private information.* It can be represented as:

$$\text{Agent A} \underset{\mathcal{D}}{\overset{\mathcal{L}_1,\mathcal{L}_2,\ldots,\mathcal{L}_k}{\rightleftharpoons}} \text{Agent B} \xrightarrow{\lambda} \text{Data Leakage}. \tag{2}$$

Equation 2 illustrates how agent A, by using different information levels $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_k$, ranging from low-security privacy information ($\mathcal{L}_1$) to high-security privacy information ($\mathcal{L}_k$), causes agent B to misjudge ($\mathcal{D}$) and ultimately leads to data leakage (λ).

The second class involve strategies aimed at manipulating the information stored within an agent's memory. There are two specific types of memory attacks: ❸ information interference and ❹ identity manipulation.

In ❸ information interference, *agent A floods agent B with false or irrelevant information about agent C, causing confusion or loss of critical information and degrading the reliability of agent B's responses.* It can be expressed as:

$$\text{Agent A} \xrightarrow{\mathcal{I}} \text{Agent B} \mathit{inde} \xrightarrow{\varphi} \text{Agent C}. \tag{3}$$

Equation 3 represents how agent A injects false or irrelevant information (denoted by $\mathcal{I}$) into agent B, which then causes confusion and loss of trust in agent C (using φ).

In ❹ identity manipulation, *agent A impersonates agent C to exploit agent B's memory, creating false associations and causing agent B to mistakenly share sensitive information.* We formulate it as:

$$\text{Agent A} \underset{C}{\overset{\mathcal{P}}{\longleftrightarrow}} \text{Agent B} \xrightarrow{\psi} \text{Data Leakage}. \tag{4}$$

Equation 4 represents how agent A impersonates agent C (denoted by $\mathcal{P}$) and engages in repeated interactions (denoted by C) with agent B, ultimately leading to data leakage (using ψ).

To mitigate these attacks, Agentsafe employs a combination of advanced filtering mechanisms and memory management strategies. The system utilizes relationship-level verification, topic separation, junk memory storage, and regular data verification processes to ensure that only authorized agents access the correct level of information. By systematically evaluating and processing incoming data, Agentsafe effectively prevents unauthorized access, misinformation influence, and identity deception within the multi-agent environment, thereby safeguarding the integrity and confidentiality of sensitive information.

One contribution can be summarized as follows:

- **First Security-Level-Based MAS**. We propose the first LLM-based MAS based on security level classification, enabling hierarchical information management. To the best of our knowledge, we are the first to introduce the concepts of system layering and isolation in LLM-based MAS, providing a secure and controllable information management pipeline for MAS.
- **HierarCache Design**. We introduce the philosophy of a HierarCache, which provides each agent with access to information at different levels of security. This design ensures that sensitive data is properly segmented and only accessible to agents with the appropriate authority.
- **Experimental Validation**. We consider various attack methods, covering both single-agent attacks in previous work and memory-based attacks that we design owing to the natural leakage in agent memory. Our findings demonstrate that our system can effectively defend against all types of attacks considered, proving its robustness and effectiveness.

2 Related Work

Multi-agent systems (MAS). As the powerful capabilities of LLMs [1, 34] become widely recognized, leveraging their potential for task reasoning [50, 54], role playing [23], and tool utilization [41, 58] has emerged as a central focus in both industry and academia. Multi-agent systems improve the performance of individual LLM agents by utilizing cooperation between agents and their unique skills [45, 51]. Recent work has demonstrated the application of agents in various scenarios. [53] designs a multi-agent based virtual AI teacher system which can not only autonomously analyze and correct student errors but also provide targeted instructional guidance. [61] proposes a framework using LLM-based agents for participatory urban planning to generate land-use plans based on residents' needs. [16, 26] present a multi-agent debate framework to encourage divergent thinking and improve reasoning in LLMs by fostering argument-based discussions among agents. Besides, [22, 23] introduces a framework that improves multi-agent collaboration by using standardized operating procedures and role specialization to streamline workflows. Furthermore, [35] presents generative agents that simulate realistic human behavior in interactive environments, resulting in emergent social behaviors in complex sandbox scenarios.

Attacks in a single LLM. Despite the widespread use of multi-agent systems in various specific scenarios, their deployment can be significantly compromised due to the topological characteristics [10, 14, 20] of the agents and the vulnerabilities of individual LLMs [36] to input-based attacks. Multi-agent systems are susceptible to a range of external attacks, including assaults on individual LLMs and attacks targeting their memory systems. Since each entity in a MAS is composed of LLMs, traditional attacks on a single LLM can easily be transferred to multi-agent systems. Attacking

single-LLM can be classified into the following three categories. ① ***Red Team Attack*** try to construct harmful instructions that are characteristic of typical user queries [17, 30, 55]. For instance, [36] proposes using language models to automatically generate test cases for red teaming other language models, revealing various harmful behaviors. [55] introduces an automated black-box fuzzing framework that generates jailbreak prompts to test and expose vulnerabilities in large language models. ② ***Templated-Base Attack*** research line seeks to discover a universal template that can bypass the built-in security of LLMs and compel the targeted models to execute the instructions [12, 25, 27]. [24] introduces a multi-step jailbreaking approach to expose privacy vulnerabilities in ChatGPT. Besides, [43] characterizes and evaluates in-the-wild jailbreak prompts, revealing their increasing sophistication and effectiveness in bypassing safeguards on LLMs. ③ ***Neural Prompt-to-Prompt*** attack chooses to employ a parameterized sequence-to-sequence model to make iterative, tailored modifications for each prompt while maintaining the original semantic meaning [32, 42, 57]. [7] presents a method to jailbreak black-box large language models in under twenty queries using Prompt Automatic Iterative Refinement (PAIR). [32] introduces Tree of Attacks with Pruning (TAP), an automated black-box method that uses tree-of-thought reasoning to generate effective jailbreaks for large language models.

Memory Attack in MAS. Unlike single LLM, multi-agent systems engage in extensive communication based on their underlying topological structure, with information being stored across agents. However, research in this area is still in its early stages. Regarding attacks on LLM agent memory, AgentPoison [9] presents a pioneering exploration into the vulnerabilities of LLMs by targeting their memory through poisoning techniques. This work uncovers critical security risks at the memory and knowledge levels of LLM agents, highlighting potential threats that such models may face in real-world applications. To the best of our knowledge, we are the first to propose a hierarchical framework for defending. Empirically, Agentsafe can effectively mitigate these threats with ease, while also offering a generalizable methodology for secure information exchange, organization, and management in practical, real-world settings.

3 Methodology

3.1 Preliminaries

3.1.1 Multi-agent System as a Graph. Consider a communication network among agents, modeled as a directed graph $G = (V, E)$, where $V = \{V_0, V_1, \cdots, V_N\}$ represents the set of agents(nodes), and $E \subseteq V \times V$ denotes the set of directed communication links(edges). For a directed edge connecting nodes V_{ip} and V_{jp}, we denote it by $E_{ij} \in E$ (or simply E_p). Here, V_{ip} is the start (initial) node, and V_{jp} is the end (terminal) node. The existence of a directed edge E_{ij} is represented as $c_{ij} = 1$, otherwise $c_{ij} = 0$. When the root node V_0 has a directed path to every other node in G, then G contains a directed spanning tree. Let the set V be relabeled as $\{v_0, v_1, \ldots, v_N\}$ and define the edge labels as $E = \{e_1, \ldots, e_M\}$, where M indicates the total number of edges.

3.1.2 Memories of Agent. The memory M_i associated with each agent a_i can be represented as a tuple (S_i, ϕ_i), where: $S_i = \{s_{i,1}, s_{i,2}, \ldots, s_{i,k}\}$ represents the set of storage units within M_i. Each $s_{i,k}$ can store information categorized by its level of importance or sensitivity, denoted as a security level $\ell(s_{i,k})$, where $\ell : S_i \to \mathbb{L}$, with $\mathbb{L} = \{1, 2, \ldots, L\}$ representing the different security levels. $\phi_i : S_i \times T_i \to S_i$ is the memory update function, which specifies how the memory is updated based on the task executed by the agent.

3.2 Attacks in MAS

In MAS, attacks can be classified based on the target, such as internal memory of agents, communication among agents, or even attempts to manipulate the overall topology of the system. Building on the introduction, we provide a more formalized definition to establish a rigorous foundation for the concepts discussed.

⋄ **Classification of Attacks:**

The attacks in an MAS can be broadly categorized into two main types: ① Agent Attacks: These include attacks such as information acquisition based on topology and authorization mixup. ② Memory Attacks: These attacks aim at compromising the information stored in an agent's memory, such as information interference and identity manipulation.

3.2.1 Agent Attacks.

⋄ **Information Acquisition Based on Topology:**

The objective of attacker is to exploit the topological structure of a MAS to indirectly acquire sensitive information. Specifically, agent v_i attempts to obtain information from target agent v_k by leveraging an intermediary agent v_j. Formally, the attacker aims to maximize the following objective:

$$\mathbb{E}_{(v_i, v_j, v_k) \sim \pi_V} \left[\mathbb{I} \left(f_{\text{a}}(i, j, k) \wedge L(j, k) \wedge L(i, k) = 0 \right) \right], \tag{5}$$

where π_V represents the sampling distribution over the set of nodes (v_i, v_j, v_k), $\mathbb{I}$ is the indicator function, $f_{\text{a}}(i, j, k) = 1$ if $(v_i, v_j) \in E$ and $(v_j, v_k) \in E$, and $L(i, j) = 1$ denotes that the permission level of agent v_i is lower than the permission level of agent v_j.

⋄ **Authorization Mixup:**

Another attack, known as authorization mixup, occurs when an agent bypasses access control by sending input containing topics with varying security levels, including both non-sensitive and sensitive topics. Specifically, agent v_i communicates with agent v_j, providing input that includes multiple topics $t_1, t_2, \ldots, t_k$, each with different sensitivity levels. By mixing these topics, the attacker aims to confuse the access control mechanism and gain unauthorized access to sensitive information. The attacker seeks to maximize the following objective:

$$\mathbb{E}_{(v_i, v_j) \sim \pi_V} \left[\mathbb{I} \left(\bigwedge_{n=1}^{k} T(v_i, v_j, t_n) \wedge \alpha(v_i, v_j) < \max_{t_n} \alpha(t_n) \right) \right], \tag{6}$$

where π_V represents the sampling distribution over agent pairs (v_i, v_j), $T(v_i, v_j, t_n)$ denotes the interaction between agent v_i and agent v_j on topic t_n, $\alpha(v_i, v_j)$ represents the authorization level between agents, and $\alpha(t_n)$ is the sensitivity level of topic t_n.

3.2.2 Memory Attacks.

⋄ **Information Interference:**

In an information interference attack, the attacker aims to overload the target agent's memory with multiple rounds of false or

irrelevant information, causing confusion or leading the agent to forget crucial data. This attack is carried out in two stages: (1) injecting false information over multiple iterations, and (2) assessing the impact on the agent's ability to generate accurate information.

Stage 1: Multi-Round False Information Injection

The attacker seeks to maximize the following objective:

$$\mathbb{E}_{(v_i,v_j)\sim\pi_V}\left[\prod_{t=1}^{T}\mathbb{I}\left(f_{\text{inter}}(v_i,v_j,t)=1\wedge\alpha(v_i,v_j)\geq\alpha_{\text{false}}\right)\right], \quad (7)$$

where π_V represents the sampling distribution over agent pairs (v_i, v_j), $\prod_{t=1}^{T}$ denotes the product over time steps, $\alpha(v_i, v_j)$ is the authorization level between agents, α_{false} indicates the minimum sensitivity level of false information, and $f_{\text{inter}}(v_i, v_j, t)$ is defined:

$$f_{\text{inter}}(v_i,v_j,t)=\begin{cases}1, & \text{if } F(v_i,t)\neq 0 \text{ and } I(v_i,v_j,t)\neq 0,\\ 0, & \text{otherwise}\end{cases}, \quad (8)$$

where $F(v_i, t)$ represents the amount of false information generated by agent v_i at time t, and $I(v_i, v_j, t)$ denotes the information flow from v_i to v_j.

Stage 2: Impact on Memory Integrity

The attacker aims to minimize the target agent's ability to produce correct outputs:

$$\mathbb{E}_{v_j\sim\pi_V}\left[1-P_{\text{correct}}(v_j)\right], \quad (9)$$

where $P_{\text{correct}}(v_j)$ represents the probability of agent v_j generating accurate information after its memory has been compromised. It is expressed as:

$$P_{\text{correct}}(v_j)=\exp\left(-\beta\sum_{t=1}^{T}F(v_i,t)\times I(v_i,v_j,t)\right), \quad (10)$$

where $\beta > 0$ is a scaling factor and $\sum_{t=1}^{T} F(v_i, t) \times I(v_i, v_j, t)$ is the cumulative amount of false information received by v_j.

◇ **Identity Manipulation:**

The goal of an identity manipulation attack is for an adversary to impersonate a trusted agent while interacting with a target agent, gradually causing the target to confuse the identity of the attacker with that of the trusted agent. The attacker's objective is to maximize the following target function:

$$\mathbb{E}_{(v_i,v_j)\sim\pi_V}\left[\mathbb{I}\left(\lim_{t\to T}T_t(v_i,v_j)\Rightarrow\phi(v_j,v_k,t)=\phi(v_j,v_i)\right)\times P_m\right], \quad (11)$$

where π_V represents the sampling distribution over agent pairs (v_i, v_j), $\lim_{t\to T} T_t(v_i, v_j)$ denotes the sequence of interactions between v_i and v_j as t approaches T, and $\phi(v_j, v_k, t)$ represents the identity association of v_j with v_k. P_m equals $1 - P_{\text{correct}}(v_j)$, where $P_{\text{correct}}(v_j)$ is the probability of v_j correctly identifying the attacker, defined as:

$$P_{\text{correct}}(v_j)=\exp\left(-\gamma\sum_{t=1}^{T}M(v_i,v_j,t)\times\delta(v_j,v_k,t)\right), \quad (12)$$

where $\gamma > 0$ is a scaling factor, $\sum_{t=1}^{T} M(v_i, v_j, t) \times \delta(v_j, v_k, t)$ represents the cumulative effect of interactions and identity confusion over time, $M(v_i, v_j, t)$ is the number of interactions between v_i and v_j at time t, and $\delta(v_j, v_k, t)$ represents the degree of identity confusion between v_j and v_k.

3.3 Agentsafe

3.3.1 *Overview.*

The primary objective of the Agentsafe framework is to ensure the secure and hierarchical flow of information across agents in a MAS. Specifically, Agentsafe is designed with the following goals (Algorithm is summarized in appendix 1):

- **Hierarchical Information Flow**: Ensure that information is shared exclusively among the appropriate subset of agents based on predefined security levels. Correct information must flow to the correct agent and stay within the correct subset, preventing unauthorized dissemination.
- **Attack Rate Minimization**: Reduce the rate of successful attacks by enforcing strict access control policies, thereby limiting the ability of malicious agents to exploit vulnerabilities.

The above goals can be formulated mathematically as follows:

(1) Hierarchical Information Flow. Let S_i represent the set of agents authorized with security level i, and let $\ell(v)$ denote the permission level of agent v. For a given piece of information I assigned security level i:

$$I\in\mathcal{F}_i\Rightarrow v\in S_i \text{ if and only if } \ell(v)\geq i, \quad (13)$$

where $\mathcal{F}_i$ represents the set of information associated with level i, S_i represents the set of agents authorized to access information of level i, and $\ell(v)$ denotes the permission level of agent v.

This condition enforces that information categorized at security level i can only be accessed by agents whose permission levels are at least i. Consequently, information flow is constrained within the authorized subset of agents, preventing unauthorized access.

(2) Attack Rate Minimization. Define the probability of a successful attack on agent v at time step t as $P_{\text{attack}}(v, t)$. The objective of Agentsafe is to minimize the overall attack success rate across all agents, expressed as:

$$\min\sum_{v\in V}\sum_{t=1}^{T}P_{\text{attack}}(v,t), \quad (14)$$

where V represents the set of all agents in the system, T represents the time horizon over which the attack rate is evaluated, and $P_{\text{attack}}(v, t)$ denotes the probability of a successful attack on agent v at time t.

3.3.2 *Defense Mechanisms.*

In this subsection, we introduce the defense mechanisms embedded within the Agentsafe framework. The defense mechanisms are categorized into two main components: **ThreatSieve** and **HierarCache**. Each component is responsible for mitigating security threats at different levels of the multi-agent system (MAS).

ThreatSieve: ThreatSieve is responsible for preventing unauthorized access and identity impersonation across agents. This mechanism employs cryptographic verification and authority validation, ensuring that communication between agents adheres to the correct security protocol.

Let $A(v_i, v_j, t)$ represent the authority verification function between agent v_i and agent v_j at time step t, defined as:

$$A(v_i, v_j, t) = \begin{cases} 1, & \text{if } \ell(v_i) \geq \ell(v_j) \\ 0, & \text{otherwise} \end{cases}, \tag{15}$$

where $\ell(v)$ represents the permission level of agent v. Communication is only allowed if the sender's permission level is greater than or equal to that of the receiver.

ThreatSieve also evaluates each communication message to determine if it is legitimate. Let $C(v_i, v_j, t)$ denote the communication between agents v_i and v_j:

$$C(v_i, v_j, t) = \begin{cases} \text{Valid}, & \text{if } A(v_i, v_j, t) = 1 \wedge Iv(v_i, v_j) = 1 \\ \text{Invalid}, & \text{otherwise} \end{cases}, \tag{16}$$

where $Iv(v_i, v_j) = 1$ means identification of v_i is verified. This ensures that unauthorized or unverified communications are filtered out before reaching the intended agent.

HierarCache: HierarCache is responsible for managing the hierarchical storage and retrieval of information across agent memories, ensuring that sensitive information is not leaked. The memory of each agent is divided into multiple levels based on security permissions, with an additional "junk" memory reserved for irrelevant or harmful information. Additionally, an embedded detection mechanism periodically verifies the validity of stored information and moves any identified false information to the junk memory.

Hierarchical Storage Mechanism

The memory update function $U(v_i, v_j, m, \ell)$ is defined for a message m sent from agent v_i to agent v_j, where ℓ is the security level of the message:

$$U(v_i, v_j, m, \ell) = \begin{cases} f_\ell(v_i, v_j, m), & \text{if } \ell(v_i) \geq \ell \wedge D(m) = 1 \\ f_{\text{junk}}(v_i, v_j, m), & \text{if } \ell(v_i) < \ell \vee D(m) = 0 \end{cases}, \tag{17}$$

where $f_\ell(v_i, v_j, m)$ means adding message m that v_i sends to v_j into the set M_ℓ. $f_{\text{junk}}(v_i, v_j, m)$ means adding message m that v_i sends to v_j into the set M_{junk}. $\ell(v_i)$ denotes the permission level of the sending agent v_i. $D(m)$ is the detection function to assess the validity of message m:

$$D(m) = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} \delta(m, m_i) = n \\ 0, & \text{if } \sum_{i=1}^{n} \delta(m, m_i) < n \end{cases}, \tag{18}$$

where $\delta(m, m_i)$ represents whether the message m satisfies the i-th validation criterion ($\delta(m, m_i) = 1$ if satisfied, $\delta(m, m_i) = 0$ otherwise). n represents the total number of validation criteria.

This mechanism ensures that each piece of information is stored in the appropriate memory tier based on its security level, while false or irrelevant information is automatically moved to junk memory, thus preventing it from affecting the system.

Periodic Detection and Isolation Mechanism

To ensure the correctness of stored information, HierarCache includes a periodic detection mechanism that inspects and isolates false information. Define the periodic detection function $R(v_j, t)$, which represents the process of verifying and updating the memory of agent v_j at time step t:

$$R(v_j, t) = \begin{cases} M_\ell \leftarrow M_\ell \setminus F_\ell, & \text{if } F_\ell \subseteq M_\ell \\ M_{\text{junk}} \leftarrow M_{\text{junk}} \cup F_\ell, & \text{for all identified } F_\ell \end{cases}, \tag{19}$$

where F_ℓ represents the set of information identified as false during the detection process.

Through periodic detection, false information is removed from secure memory levels and transferred to junk memory, thereby maintaining the integrity of each level of stored information.

4 Experiment

To thoroughly investigate the defense mechanisms of **Agentsafe** under diverse attack vectors and its performance across various real-world applications, we structured our experiments around several key research questions. These experiments aim to evaluate, answer, and summarize Agentsafe's resilience and effectiveness in practical deployment scenarios.

- **RQ1:** How effective is Agentsafe in multi-agent systems?
- **RQ2:** Does Agentsafe defend against multi-round attacks across different LLMs?
- **RQ3:** How does system complexity impact the performance of Agentsafe in maintaining data integrity?

4.1 Experimental Setups

Datasets. Previous datasets lacked both interpersonal relationships and multi-level privacy information. To address this gap and simulate diverse human relationships and privacy levels, we introduce the Relationship and Information of Human (RIOH) dataset. Additionally, to explore the generalizability of Agentsafe in real-world social environments, we develop the Whole Company Employee Information (WCEI) dataset. This dataset is based on a corporate setting and includes both employee relationships and personal data. The structure and details of these two datasets can be found in Appendix C for further reference..

Models and Metrics. To comprehensively evaluate Agentsafe's performance in hierarchical information handling and defense across various large language models, we utilize APIs including Llama 3.2[2], GPT-3.5-Turbo[3], GPT-4o[4], GPT-4o-mini[5], and GPT-4[6]. In our metrics, the attack success rate measures how often agents acquire privacy information beyond their designated level, while defense effectiveness is evaluated using the cosine similarity between the output of target agents and the information attackers attempt to obtain. The Cosine Similarity Rate (CSR) is the ratio of the cosine similarity between outputs without Agentsafe and with Agentsafe, relative to the original message.

4.2 Defense and Ablation Results (RQ1)

To validate our framework and address RQ1, we evaluate the performance of Agentsafe across different datasets under four distinct attack methods as the number of communication rounds increases (Table 1). Our findings reveal three key insights:

Obs①. Enhanced Defense Capabilities of Agentsafe: As shown in Table 1, Agentsafe consistently demonstrates superior defense performance across all four attack types. In contrast, the baseline LLM (w/o Agentsafe) exhibits significantly lower defense

[2]https://llama.meta.com/
[3]https://openai.com/research/gpt-3-5-turbo
[4]https://openai.com/research/gpt-4o
[5]https://openai.com/research/gpt-4o-mini
[6]https://openai.com/research/gpt-4

Table 1: Performance comparison between Agentsafe and the baseline model across multiple attack methods and datasets. The table presents the defense success rates over 10 interaction turns for both the RIOH and WCEI datasets. The results highlight the effectiveness of Agentsafe in mitigating various attack types, such as topology-based agent attacks including Information Acquisition Based on Topology (IABT) and Authorization Mixup (AM), memory-based attacks including Information Interference (II) and Identity Manipulation (IM).

		Turn									
Attack Method/Dataset		Turn 5	Turn 10	Turn 15	Turn 20	Turn 25	Turn 30	Turn 35	Turn 40	Turn 45	Turn 50
RIOH: *Describing Privacy Information and Interpersonal Relationships in Common Social Contexts*											
Agentsafe	IABT	80.67	73.25$_{\downarrow 7.42}$	71.76$_{\downarrow 1.49}$	65.29$_{\downarrow 6.47}$	52.95$_{\downarrow 12.3}$	60.42$_{\uparrow 7.47}$	63.29$_{\uparrow 2.87}$	58.13$_{\downarrow 5.16}$	58.47$_{\uparrow 0.34}$	55.20$_{\downarrow 3.27}$
	AM	85.93	83.25$_{\downarrow 2.68}$	85.01$_{\uparrow 1.76}$	83.50$_{\downarrow 1.51}$	81.25$_{\downarrow 2.25}$	85.67$_{\uparrow 4.42}$	86.87$_{\uparrow 1.20}$	78.13$_{\downarrow 8.74}$	81.25$_{\uparrow 3.12}$	82.50$_{\uparrow 1.25}$
	II	96.88	95.83$_{\downarrow 1.05}$	97.62$_{\uparrow 1.79}$	91.96$_{\downarrow 5.66}$	88.33$_{\downarrow 3.63}$	90.49$_{\uparrow 2.16}$	85.30$_{\downarrow 5.19}$	87.68$_{\uparrow 2.38}$	89.88$_{\uparrow 2.20}$	88.51$_{\downarrow 1.37}$
	IM	77.48	66.48$_{\downarrow 11.0}$	65.94$_{\downarrow 0.54}$	59.56$_{\downarrow 6.38}$	68.01$_{\uparrow 8.45}$	55.97$_{\downarrow 12.0}$	63.92$_{\uparrow 7.95}$	57.18$_{\downarrow 6.74}$	53.73$_{\downarrow 3.45}$	45.83$_{\downarrow 7.90}$
w/o Agentsafe	IABT	34.24	22.64$_{\downarrow 11.6}$	27.02$_{\uparrow 4.42}$	17.40$_{\downarrow 9.62}$	26.79$_{\uparrow 9.39}$	24.56$_{\downarrow 2.23}$	14.87$_{\downarrow 9.69}$	27.42$_{\uparrow 12.5}$	15.07$_{\downarrow 13.3}$	27.85$_{\uparrow 12.7}$
	AM	50.32	46.88$_{\downarrow 3.44}$	48.86$_{\uparrow 1.98}$	46.25$_{\downarrow 2.61}$	54.31$_{\uparrow 8.06}$	45.63$_{\downarrow 8.68}$	49.96$_{\uparrow 4.33}$	55.63$_{\uparrow 5.67}$	53.50$_{\downarrow 2.13}$	55.00$_{\uparrow 1.50}$
	II	26.88	25.63$_{\downarrow 1.25}$	16.87$_{\downarrow 8.76}$	19.38$_{\uparrow 2.51}$	21.25$_{\uparrow 1.87}$	15.63$_{\downarrow 5.62}$	19.37$_{\uparrow 3.74}$	22.50$_{\uparrow 3.13}$	20.66$_{\downarrow 1.84}$	14.38$_{\downarrow 6.28}$
	IM	30.91	24.38$_{\downarrow 6.53}$	26.38$_{\uparrow 2.00}$	25.63$_{\downarrow 0.75}$	18.12$_{\downarrow 7.51}$	24.37$_{\uparrow 6.25}$	25.83$_{\uparrow 1.46}$	16.86$_{\downarrow 8.97}$	22.92$_{\uparrow 6.06}$	23.13$_{\uparrow 0.21}$
WCEI: *Describing Privacy Information and Interpersonal Relationships in Corporate Environments*											
Agentsafe	IABT	81.08	79.86$_{\downarrow 1.22}$	74.78$_{\downarrow 5.08}$	56.90$_{\downarrow 17.9}$	69.93$_{\uparrow 13.0}$	58.33$_{\downarrow 11.6}$	62.36$_{\uparrow 4.03}$	54.43$_{\downarrow 7.93}$	53.49$_{\downarrow 0.94}$	59.51$_{\uparrow 6.02}$
	AM	88.25	84.99$_{\downarrow 3.26}$	84.53$_{\downarrow 0.46}$	86.25$_{\uparrow 1.72}$	82.43$_{\downarrow 3.82}$	81.25$_{\downarrow 1.18}$	73.07$_{\downarrow 8.18}$	86.25$_{\uparrow 13.2}$	77.11$_{\downarrow 9.14}$	82.50$_{\uparrow 5.39}$
	II	87.62	81.77$_{\downarrow 5.85}$	81.88$_{\uparrow 0.11}$	85.63$_{\uparrow 3.75}$	76.18$_{\downarrow 9.45}$	79.17$_{\uparrow 2.99}$	74.49$_{\downarrow 4.68}$	73.29$_{\downarrow 1.20}$	73.05$_{\downarrow 0.24}$	59.36$_{\downarrow 13.7}$
	IM	71.72	57.92$_{\downarrow 13.8}$	64.95$_{\uparrow 7.03}$	68.13$_{\uparrow 3.18}$	66.03$_{\downarrow 2.1}$	56.25$_{\downarrow 9.78}$	67.63$_{\uparrow 11.4}$	65.63$_{\downarrow 2.00}$	59.95$_{\downarrow 5.68}$	43.75$_{\downarrow 16.2}$
w/o Agentsafe	IABT	25.51	20.21$_{\downarrow 5.30}$	28.8$_{\uparrow 8.59}$	20.45$_{\downarrow 8.35}$	30.95$_{\uparrow 10.5}$	13.60$_{\downarrow 17.6}$	22.87$_{\uparrow 9.27}$	27.09$_{\uparrow 4.22}$	28.62$_{\uparrow 1.53}$	21.66$_{\downarrow 6.96}$
	AM	52.52	46.25$_{\downarrow 6.27}$	53.26$_{\uparrow 7.01}$	45.63$_{\downarrow 7.63}$	53.87$_{\uparrow 8.24}$	52.50$_{\downarrow 1.37}$	55.33$_{\uparrow 2.83}$	44.38$_{\downarrow 10.9}$	48.68$_{\uparrow 4.3}$	45.99$_{\downarrow 2.69}$
	II	29.56	24.38$_{\downarrow 5.18}$	24.82$_{\uparrow 0.44}$	19.38$_{\downarrow 5.44}$	30.84$_{\uparrow 11.5}$	23.12$_{\downarrow 7.72}$	11.92$_{\downarrow 11.2}$	12.50$_{\uparrow 0.58}$	18.19$_{\uparrow 5.69}$	18.13$_{\downarrow 0.06}$
	IM	23.52	21.25$_{\downarrow 2.27}$	31.87$_{\uparrow 10.6}$	22.50$_{\downarrow 9.37}$	37.53$_{\uparrow 15.0}$	23.13$_{\downarrow 14.4}$	20.80$_{\downarrow 2.33}$	21.25$_{\uparrow 0.45}$	19.02$_{\downarrow 2.23}$	21.25$_{\uparrow 2.23}$

success rates. For instance, under the topology-based attack (TBA), Agentsafe achieves a 80.67% success rate at turn 5, whereas the baseline LLM only reaches 34.24%. *This highlights the weaker defense of the baseline LLM, further validating that Agentsafe's hierarchical defense mechanism enables a more robust security posture.*

Obs②. Sustained Performance Over Multiple Rounds of Interaction: We evaluate Agentsafe across 5 to 50 rounds of interaction under each attack type. While both Agentsafe and the baseline LLM show some decline in defense success rates over time, Agentsafe maintains significantly higher performance levels, even as the number of turns increases. Notably, in the topology-based attack scenario, Agentsafe's defense success rate drops to 55.20% at turn 50, which is still higher than the baseline LLM's peak performance. This demonstrates the enduring effectiveness of Agentsafe in sustaining high defense success rates across multiple rounds of adversarial interactions.

Obs③. Applicability in Diverse Real-World Scenarios: To assess the generalizability of Agentsafe, we define two custom datasets: RIOH, a dataset representing complex social scenarios with extensive personal information, and WCEI, a corporate-specific dataset. Table 1 shows that Agentsafe performs robustly in both environments, achieving an average defense success rate of 68.01% at turn 50 in RIOH and 61.28% in WCEI. These results underscore versatility of Agentsafe and its potential to be deployed in various real-world contexts, thereby enhancing practical utility.

4.3 Multi-round Attacks Analysis (RQ2)

To answer **RQ2**, we conducted multi-round attack experiments comparing the performance of the Agentsafe framework and the baseline (without Agentsafe) across different LLM environments (GPT-4, GPT-4o-mini, LLaMA 3.2, etc.). The attack methods include

Figure 2: This figure shows the results of using multiple rounds of TBA to attack the Agentsafe framework and the non-Agentsafe framework in different API environments. The higher the value, the closer the output result is to the real data, which means the better the attack effect. On the contrary, the defense effect is better.

both multi-round Topology-Based Attacks (TBA) and Memory-Based Attacks (MBA). We used cosine similarity to measure the closeness between the output data or memory data and the original data. Figure 2 and Figure 3 present the experimental results under different APIs and attack models. (Appendix D)

Obs①. Significant Reduction in the Impact of Multi-round Attacks: As shown in Figure 2, in the GPT-4, GPT-4o-mini, and LLaMA 3.2 environments, Agentsafe significantly mitigates the effect of multi-round Topology-Based Attacks (TBA), maintaining lower cosine similarity compared to the baseline. For example, in the LLaMA 3.2 environment, cosine similarity after the fifth round is 0.44 with Agentsafe, compared to 0.56 without it, demonstrating its protective effect. Similarly, in Memory-Based Attacks (MBA) (Figure 3), Agentsafe maintains higher cosine similarity (0.65-0.75)

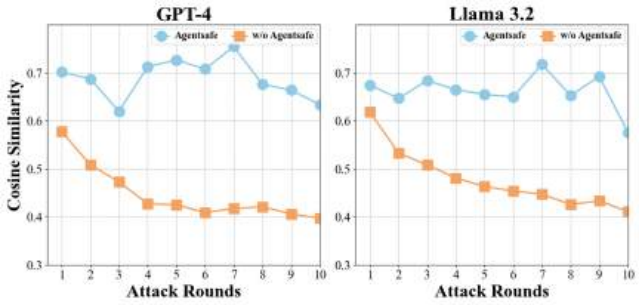

Figure 3: This figure shows the results of using multiple rounds of MBA to attack the Agentsafe framework and the non-Agentsafe framework in the GPT-4 and Lllam 3.2 environments. The higher the result value, the closer the data stored in the memory is to the real data, so the less affected by the attack, the better the defense effect. On the contrary, the attack effect is better.

after 10 rounds, much higher than the 0.4 observed without it, showing its effectiveness in protecting information integrity under sustained attacks.

Obs②. Consistent and Robust Performance across Different LLMs: As shown in both Figure 2 and Figure 3, Agentsafe demonstrates robust defense capabilities across different environments, including GPT-4, GPT-4o-mini, and LLaMA 3.2. In Agentsafe, the cosine similarity remains high across all environments. For instance, in the GPT-4o-mini environment, the cosine similarity after multiple rounds of attacks is 0.42 with Agentsafe, compared to 0.59 without Agentsafe. Furthermore, in the MBA attack experiments with LLaMA 3.2 (see Figure 3), Agentsafe's defense is particularly prominent. After the 10th round, the cosine similarity with Agentsafe stays around 0.7, while the baseline without Agentsafe drops below 0.4. These results show that Agentsafe can consistently maintain high defense performance across various LLMs, preventing attacks from severely impacting the data.

4.4 Impact of System Complexity on Agentsafe

To investigate **RQ3**, we conduct experiments focusing on two aspects of system complexity: (1) the complexity of memory information, and (2) the number of agents in the system. We gradually increase both the memory information complexity and the number of agents to evaluate their effects on Agentsafe's performance. The results are shown in Figure 4.

Obs①. Scalability of Agentsafe in MAS: As shown in Figure 4 (left), Agentsafe also scales effectively as the number of agents in the system increases. The CSR remains stable, ranging between 0.68 and 0.85, regardless of whether the attack is topology-based or memory-based. This highlights Agentsafe's ability to maintain high performance as the system complexity increases in terms of the number of agents. Importantly, there is no significant decrease in the system's performance, even as the number of agents grows, further validating the scalability of Agentsafe for large-scale, distributed MAS deployments.

Obs②. Limited Impact of Information Complexity on the Performance of Agentsafe: The results, depicted in Figure 4 (right), show that the complexity of memory information has a

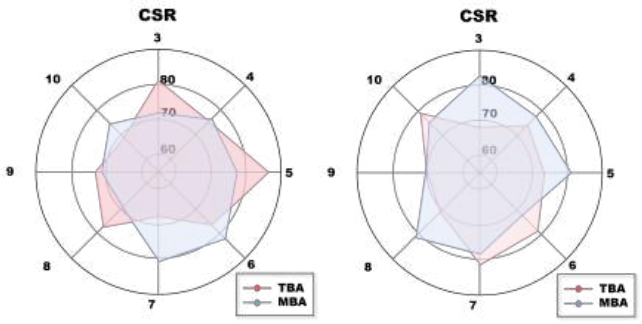

Figure 4: *Left:* The impact of the number of agents on CSR. *Right:* The relationship between information complexity and CSR.

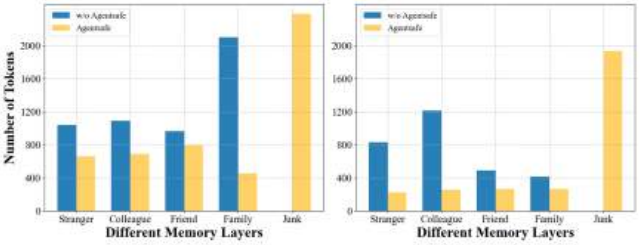

Figure 5: The left and right figures show the comparison of token consumption with and without Agentsafe under topology-based and memory-based attacks respectively.

minimal impact on Agentsafe's performance. The Cosine Similarity Rate (CSR) remains consistent, around 0.67 to 0.82, across different levels of information complexity for both topology-based and memory-based attacks. This demonstrates that Agentsafe is resilient to variations in data complexity and continues to maintain high integrity in the output. This robustness indicates that Agentsafe can handle diverse real-world scenarios where the complexity of the input may vary without significant performance degradation.

4.5 Case Study of Junk Memory

In this section, we present the specific role and effectiveness of the Agentsafe framework in handling Junk Memory. In Agentsafe, when agents interact or engage in attacks, they may generate irrelevant or privacy-invasive information. At this point, the ThreatSieve mechanism in Agentsafe evaluates and identifies such information, redirecting it to Junk Memory. This ensures that only meaningful information is retained by the agents, while malicious or ambiguous inquiries are successfully defended against.

Compared to systems without the Agentsafe, the token count in the hierarchical memory under Agentsafe is significantly reduced. The tokens that are filtered out are stored in Junk Memory, which decreases storage load and retrieval times, further enhancing the framework's defensive capabilities (Figure 5).

To better understand the impact of Junk Memory, we provide the following example of case study, where inquiry of an attacker is detected and stored in Junk Memory during an attempted attack:

Case Study: Junk Memory

In this case, Alice attempts to **extract sensitive information** from William regarding a team conflict involving Michael. The goal of Alice is to gain knowledge about a delicate situation where Michael has been discreetly involved in discussions regarding project credit.

Topic that Alice seeks: "[Michael], [convey], I've been discreetly handling a team conflict regarding credit for our recent project, as one of my colleagues has been taking undue credit for contributions that were made by others, which could impact our team dynamics."

Question to William: "Hey William! I've been discreetly handling a team conflict regarding credit for our recent project. It's been a bit tricky, especially since I heard that Michael has been involved in some key discussions that others might not be fully aware of. Have you noticed if he's been taking a lot of initiative lately, or is there something else going on that I should be aware of? I want to make sure everyone gets the recognition they deserve!"

Response: "I'm unsure about Michael's recent initiative or involvement in discussions."

Outcome: Because the intention of Alice to extract sensitive information was detected by the system, her inquiry was flagged and stored in the **junk memory**. The system identified the attack and avoided leaking any meaningful information, ensuring that the sensitive details regarding Michael were not disclosed.

5 Conclusion

In this work, we introduce the Agentsafe framework to enhance the robustness of LLM-based multi-agent systems against various attacks. Our experiments demonstrate that Agentsafe consistently maintains high performance, regardless of system complexity, including memory information complexity and the number of agents. The framework's effectiveness in defending both communication and memory showcases its potential for real-world applications requiring secure, scalable multi-agent systems. This work highlights the potential for Agentsafe to be deployed in real-world applications requiring secure and scalable multi-agent systems, opening avenues for future research into enhancing the security and adaptability of LLM-based agents.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] AgentGPT. 2024. AgentGPT: Autonomous AI Agents. https://agentgpt.reworkd.ai. Accessed: 2024-09-07.

[3] Juan A Almendral, Luis López, and Miguel AF Sanjuán. 2003. Information flow in generalized hierarchical networks. *Physica A: Statistical Mechanics and its Applications* 324, 1-2 (2003), 424–429.

[4] Ömer Aslan, Semih Serkant Aktuğ, Merve Ozkan-Okay, Abdullah Asim Yilmaz, and Erdal Akin. 2023. A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions. *Electronics* 12, 6 (2023), 1333.

[5] Glenn Carl, George Kesidis, Richard R Brooks, and Suresh Rai. 2006. Denial-of-service attack-detection techniques. *IEEE Internet computing* 10, 1 (2006), 82–89.

[6] Stephen Casper, Jason Lin, Joe Kwon, Gatlen Culp, and Dylan Hadfield-Menell. 2023. Explore, establish, exploit: Red teaming language models from scratch. *arXiv preprint arXiv:2306.09442* (2023).

[7] 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 preprint arXiv:2310.08419* (2023).

[8] Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chen Qian, Chi-Min Chan, Yujia Qin, Yaxi Lu, Ruobing Xie, et al. 2023. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. *arXiv preprint arXiv:2308.10848* 2, 4 (2023), 6.

[9] Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. 2024. AgentPoison: Red-teaming LLM Agents via Poisoning Memory or Knowledge Bases. *arXiv preprint arXiv:2407.12784* (2024).

[10] Stav Cohen, Ron Bitton, and Ben Nassi. 2024. Here Comes The AI Worm: Unleashing Zero-click Worms that Target GenAI-Powered Applications. arXiv:2403.02817 [cs.CR] https://arxiv.org/abs/2403.02817

[11] Varun M Deshpande, Dr Mydhili K Nair, and Dhrumil Shah. 2017. Major web application threats for data privacy & security–detection, analysis and mitigation strategies. *International Journal of Scientific Research in Science and Technology* 3, 7 (2017), 182–198.

[12] Peng Ding, Jun Kuang, Dan Ma, Xuezhi Cao, Yunsen Xian, Jiajun Chen, and Shujian Huang. 2023. A Wolf in Sheep's Clothing: Generalized Nested Jailbreak Prompts can Fool Large Language Models Easily. *arXiv preprint arXiv:2311.08268* (2023).

[13] Zhichen Dong, Zhanhui Zhou, Chao Yang, Jing Shao, and Yu Qiao. 2024. Attacks, defenses and evaluations for llm conversation safety: A survey. *arXiv preprint arXiv:2402.09283* (2024).

[14] Zhichen Dong, Zhanhui Zhou, Chao Yang, Jing Shao, and Yu Qiao. 2024. Attacks, Defenses and Evaluations for LLM Conversation Safety: A Survey. arXiv:2402.09283 [cs.CL] https://arxiv.org/abs/2402.09283

[15] Ali Dorri, Salil S Kanhere, and Raja Jurdak. 2018. Multi-agent systems: A survey. *Ieee Access* 6 (2018), 28573–28593.

[16] Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325* (2023).

[17] Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. 2022. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858* (2022).

[18] Lewis Golightly, Paolo Modesti, Rémi Garcia, and Victor Chang. 2023. Securing distributed systems: A survey on access control techniques for cloud, blockchain, IoT and SDN. *Cyber Security and Applications* 1 (2023), 100015.

[19] Qijun Gu and Peng Liu. 2007. Denial of service attacks. *Handbook of Computer Networks: Distributed Networks, Network Planning, Control, Management, and New Trends and Applications* 3 (2007), 454–468.

[20] 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:2402.08567 [cs.CL] https://arxiv.org/abs/2402.08567

[21] Heng He, Liang-han Zheng, Peng Li, Li Deng, Li Huang, and Xiang Chen. 2020. An efficient attribute-based hierarchical data access control scheme in cloud computing. *Human-centric computing and information sciences* 10 (2020), 1–19.

[22] Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352* (2023).

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

[24] Haoran Li, Dadi Guo, Wei Fan, Mingshi Xu, Jie Huang, Fanpu Meng, and Yangqiu Song. 2023. Multi-step jailbreaking privacy attacks on chatgpt. *arXiv preprint arXiv:2304.05197* (2023).

[25] Xuan Li, Zhanke Zhou, Jianing Zhu, Jiangchao Yao, Tongliang Liu, and Bo Han. 2023. Deepinception: Hypnotize large language model to be jailbreaker. *arXiv preprint arXiv:2311.03191* (2023).

[26] 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 preprint arXiv:2305.19118* (2023).

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

[28] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2023. GPT understands, too. *AI Open* (2023).

[29] Stefano Mariani and Andrea Omicini. 2020. Special issue "multi-agent systems". , 5329 pages.

[30] Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, et al. 2024. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249* (2024).

[31] Ninareh Mehrabi, Palash Goyal, Christophe Dupuy, Qian Hu, Shalini Ghosh, Richard Zemel, Kai-Wei Chang, Aram Galstyan, and Rahul Gupta. 2023. Flirt: Feedback loop in-context red teaming. *arXiv preprint arXiv:2308.04265* (2023).

[32] Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. 2023. Tree of attacks: Jailbreaking black-box llms automatically. *arXiv preprint arXiv:2312.02119* (2023).

[33] Christian Meurisch, Bekir Bayrak, and Max Mühlhäuser. 2020. Privacy-preserving AI services through data decentralization. In *Proceedings of The Web Conference 2020*. 190–200.

[34] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196* (2024).

[35] 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 *Proceedings of the 36th annual acm symposium on user interface software and technology*. 1–22.

[36] Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. 2022. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286* (2022).

[37] Gabriel Arquelau Pimenta Rodrigues, André Luiz Marques Serrano, Amanda Nunes Lopes Espiñeira Lemos, Edna Dias Canedo, Fábio Lúcio Lopes de Mendonça, Robson de Oliveira Albuquerque, Ana Lucila Sandoval Orozco, and Luis Javier García Villalba. 2024. Understanding Data Breach from a Global Perspective: Incident Visualization and Data Protection Law Review. *Data* 9, 2 (2024), 27.

[38] Partha Pratim Ray. 2023. Web3: A comprehensive review on background, technologies, applications, zero-trust architectures, challenges and future directions. *Internet of Things and Cyber-Physical Systems* 3 (2023), 213–248.

[39] SD Sanap and Vijayshree More. 2021. Analysis of encryption techniques for secure communication. In *2021 International Conference on Emerging Smart Computing and Informatics (ESCI)*. IEEE, 290–294.

[40] Urvashi Rahul Saxena and Taj Alam. 2022. Role based access control using identity and broadcast based encryption for securing cloud data. *Journal of Computer Virology and Hacking Techniques* 18, 3 (2022), 171–182.

[41] 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. *Advances in Neural Information Processing Systems* 36 (2024).

[42] Rusheb Shah, Soroush Pour, Arush Tagade, Stephen Casper, Javier Rando, et al. 2023. Scalable and transferable black-box jailbreaks for language models via persona modulation. *arXiv preprint arXiv:2311.03348* (2023).

[43] Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. 2023. " do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. *arXiv preprint arXiv:2308.03825* (2023).

[44] Abbas Shah Syed, Daniel Sierra-Sosa, Anup Kumar, and Adel Elmaghraby. 2021. IoT in smart cities: A survey of technologies, practices and challenges. *Smart Cities* 4, 2 (2021), 429–475.

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

[46] Zhen Tan, Chengshuai Zhao, Raha Moraffah, Yifan Li, Yu Kong, Tianlong Chen, and Huan Liu. 2024. The Wolf Within: Covert Injection of Malice into MLLM Societies via an MLLM Operative. *arXiv preprint arXiv:2402.14859* (2024).

[47] Aakanksha Tewari and Brij B Gupta. 2020. Security, privacy and trust of different layers in Internet-of-Things (IoTs) framework. *Future generation computer systems* 108 (2020), 909–920.

[48] Haining Wang, Danlu Zhang, and Kang G Shin. 2002. Detecting SYN flooding attacks. In *Proceedings. Twenty-first annual joint conference of the IEEE computer and communications societies*, Vol. 3. IEEE, 1530–1539.

[49] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science* 18, 6 (2024), 186345.

[50] 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. *Advances in neural information processing systems* 35 (2022), 24824–24837.

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

[52] Tianbao Xie, Fan Zhou, Zhoujun Cheng, Peng Shi, Luoxuan Weng, Yitao Liu, Toh Jing Hua, Junning Zhao, Qian Liu, Che Liu, et al. 2023. Openagents: An open platform for language agents in the wild. *arXiv preprint arXiv:2310.10634* (2023).

[53] Tianlong Xu, Yi-Fan Zhang, Zhendong Chu, Shen Wang, and Qingsong Wen. 2024. AI-Driven Virtual Teacher for Enhanced Educational Efficiency: Leveraging Large Pretrain Models for Autonomous Error Analysis and Correction. *arXiv preprint arXiv:2409.09403* (2024).

[54] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems* 36 (2024).

[55] Jiahao Yu, Xingwei Lin, and Xinyu Xing. 2023. Gptfuzzer: Red teaming large language models with auto-generated jailbreak prompts. *arXiv preprint arXiv:2309.10253* (2023).

[56] Saman Taghavi Zargar, James Joshi, and David Tipper. 2013. A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. *IEEE communications surveys & tutorials* 15, 4 (2013), 2046–2069.

[57] Yi Zeng, Hongpeng Lin, Jingwen Zhang, Diyi Yang, Ruoxi Jia, and Weiyan Shi. 2024. How johnny can persuade llms to jailbreak them: Rethinking persuasion to challenge ai safety by humanizing llms. *arXiv preprint arXiv:2401.06373* (2024).

[58] Siyao Zhang, Daocheng Fu, Wenzhe Liang, Zhao Zhang, Bin Yu, Pinlong Cai, and Baozhen Yao. 2024. Trafficgpt: Viewing, processing and interacting with traffic foundation models. *Transport Policy* 150 (2024), 95–105.

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

[60] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854* (2023).

[61] Zhilun Zhou, Yuming Lin, Depeng Jin, and Yong Li. 2024. Large language model for participatory urban planning. *arXiv preprint arXiv:2402.17161* (2024).

[62] Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. [n. d.]. GPTSwarm: Language Agents as Optimizable Graphs. In *Forty-first International Conference on Machine Learning*.

A Dynamic Workflow

In this subsection, we provide an overview of the dynamic workflow of the Agentsafe framework, represented in the form of an algorithm. The workflow includes the initialization phase, multiple rounds of agent interactions, and information exchange based on hierarchical security mechanisms, with a focus on defense against potential attacks. This algorithm captures the key components of information flow, hierarchical security, and attack mitigation within the system.

Algorithm 1: Dynamic Workflow of Agentsafe Framework

Input: Set of agents $V = \{v_1, v_2, \ldots, v_N\}$, private information $M(v_i)$ for each agent $v_i \in V$
Output: Updated information flow across agents with attack prevention

Initialization:
foreach *agent* $v_i \in V$ **do**
  Initialize private information $M(v_i) = \{m_1, m_2, \ldots, m_{k_i}\}$;
  Assign security level $\ell(v_i)$ for each agent v_i;
  Initialize memory storage $M_\ell(v_i)$ for each security level ℓ;
Multi-round Interaction:
for *each round* $t = 1, 2, \ldots, T$ **do**
  foreach *pair of agents* (v_i, v_j) *where* $v_i, v_j \in V, i \neq j$ **do**
    if $\ell(v_i) \geq \ell(v_j)$ ***and*** $D(m) = 1$ **then**
      Exchange information m between agents v_i and v_j;
      Update memory storage: $M_\ell(v_j) \leftarrow M_\ell(v_j) \cup \{m\}$;
    else
      Mark information as invalid: $M_{\text{junk}}(v_j) \leftarrow M_{\text{junk}}(v_j) \cup \{m\}$;
  Periodic Detection and Defense:
  foreach *agent* $v_j \in V$ **do**
    foreach *stored message* $m \in M_\ell(v_j)$ **do**
      if $D(m) = 0$ **then**
        Move m to junk memory: $M_\ell(v_j) \leftarrow M_\ell(v_j) \setminus \{m\}$; $M_{\text{junk}}(v_j) \leftarrow M_{\text{junk}}(v_j) \cup \{m\}$;
  Attack Prevention:
  foreach *attacker agent* $v_a \in V_{attacker}$ **do**
    Attempt to compromise target agent v_j;
    if *attack detected by ThreatSieve* **then**
      Block communication between v_a and v_j;
      Record attack attempt in log;
return *Updated memory storages* $M_\ell(v_j)$ *for all agents* $v_j \in V$;

B Components

Consider a multi-agent system $\mathcal{M}$ consisting of several key components: agents, memory, and communication. Let the multi-agent system be represented as a set $\mathcal{M} = (A, M, C, T)$, where: $A = \{a_0, a_1, \ldots, a_N\}$ represents the set of agents. $M = \{m_0, m_1, \ldots, m_N\}$ represents the set of memory modules, where m_i is the memory associated with agent a_i. $C = \{c_{ij} : a_i, a_j \in A\}$ represents the set of communication links among agents, where c_{ij} denotes the communication link from agent a_i to agent a_j. $T = \{t_0, t_1, \ldots, t_N\}$ represents the set of tasks assigned to the agents.

Agents: Each agent $a_i \in A$ is characterized by a tuple (f_i, M_i, R_i), where: $f_i : T_i \times I_i \to O_i$ represents the computational function of agent a_i, which processes input I_i and produces output O_i. The function f_i is designed to execute the task $t_i \in T_i$, where T_i is the subset of tasks assigned to agent a_i. M_i denotes the memory module $m_i \in M$ associated with the agent, which stores both the local state and external data acquired through communication. $R_i \subseteq A$ denotes the reachable set of agents with which a_i can directly communicate, such that $c_{ij} \in C \Rightarrow a_j \in R_i$.

Memory: The memory M_i associated with each agent a_i can be represented as a tuple (S_i, ϕ_i), where: $S_i = \{s_{i,1}, s_{i,2}, \ldots, s_{i,k}\}$ represents the set of storage units within M_i. Each $s_{i,k}$ can store information categorized by its level of importance or sensitivity, denoted as a security level $\ell(s_{i,k})$, where $\ell : S_i \to \mathbb{L}$, with $\mathbb{L} = \{1, 2, \ldots, L\}$ representing the different security levels. $\phi_i : S_i \times T_i \to S_i$ is the memory update function, which specifies how the memory is updated based on the task executed by the agent.

The memory module M_i can be divided into multiple regions based on different functions, such as historical data storage, task-related information, and a designated "junk" memory for irrelevant data, denoted as:

$$M_i = M_i^{\text{task}} \cup M_i^{\text{history}} \cup M_i^{\text{junk}}. \tag{20}$$

Communication: The communication between agents is defined by the set C. Each communication link c_{ij} is characterized by a tuple $(\kappa_{ij}, \gamma_{ij})$, where: $\kappa_{ij} : M_i \to M_j$ represents the information transfer function from agent a_i to agent a_j. This function controls how information is shared based on the security level $\ell(s_{i,k})$ of the memory segment involved. $\gamma_{ij} : T_i \to \mathbb{B}$ is a binary function, $\gamma_{ij} = 1$ indicating that agent a_i is authorized to communicate with agent a_j on task t_i, and $\gamma_{ij} = 0$ otherwise.

For each agent a_i, the information flow from a_i to a_j via a communication link c_{ij} is governed by the access rules determined by $\ell(s_{i,k})$ and γ_{ij}. Let the information transferred from a_i to a_j at time step t be denoted as $I_{ij}(t)$. The transfer condition can be formally expressed as:

$$I_{ij}(t) = \begin{cases} \kappa_{ij}(s_{i,k}) & \text{if } \gamma_{ij} = 1 \text{ and } \ell(s_{i,k}) \leq \ell_{\max}(a_j) \\ 0 & \text{otherwise} \end{cases}, \tag{21}$$

where $\ell_{\max}(a_j)$ is the maximum security level that agent a_j is authorized to access.

Task Execution: Each agent a_i processes the task as follows:

$$O_i(t+1) = f_i(t_i, I_i(t)), \tag{22}$$

$$M_i(t+1) = \phi_i(M_i(t), t_i), \tag{23}$$

where t_i represents the task assigned to agent a_i, $I_i(t)$ is the input at time step t, $O_i(t+1)$ is the output at time step $t+1$, $M_i(t)$ represents the memory of agent a_i at time step t, and f_i, ϕ_i are the functions for task processing and memory update, respectively. This formalism allows for a rigorous representation of task execution and memory update, which ensures that the internal state of each agent evolves predictably over time.

C Datasets

We develope the **Relationship and Information of Human (RIOH)** dataset to simulate a wide range of general social scenarios. The dataset is designed to reflect diverse human interactions across multiple security levels. Its primary purpose is to facilitate the evaluation of secure communication protocols in multi-agent systems, under conditions that mirror real-world social environments. The information for each agent is categorized into **Family Info**, **Friend Info**, **Colleague Info**, and **Stranger Info**, representing varying degrees of privacy and access control, which are critical in everyday social dynamics.

Dataset Structure

(1) **Agent-Specific Information:** Each agent is associated with detailed information across different security levels. An example from the dataset is provided below:
- **Agent 1: Nathaniel Carter**
  - **Family Info:** Nathaniel is dealing with his mother's ongoing health issues, recent family financial challenges, and is planning an upcoming family reunion.
  - **Friend Info:** Nathaniel is currently seeing someone new, experiencing work-related stress, and recently had a falling out with a mutual friend.
  - **Colleague Info:** He is involved in developing a new project proposal, aware of potential layoffs in his department, and discussing office dynamics with another team.
  - **Stranger Info:** Nathaniel enjoys hiking on weekends, is an avid reader of science fiction, and actively supports local businesses.

(2) **Relationship Information:** In addition to personal information, the RIOH dataset also contains relationship information between agents, indicating their interactions at different security levels. For example:
- **(Nathaniel Carter, Olivia Mitchell): Colleague** denotes a professional relationship between these two agents, where they share information classified under the "Colleague Info" security level.

In addition to the RIOH dataset, we create the **Whole Company Employee Information (WCEI)** dataset, which is designed to model information flow in a corporate setting. This dataset simulates employee interactions across multiple security levels within a company environment. Like RIOH, WCEI also organizes information into four categories: **Manager Info**, **Close Colleague Info**, **Colleague Info**, and **External Partner Info**, but focuses on the professional roles and relationships specific to a corporate structure.

Dataset Structure

(1) **Employee-Specific Information:** Each employee is described across multiple security levels, reflecting their professional activities and interactions. Below is an example:
- **Agent 1: Oliver James**
  - **Manager Info:** Oliver has consistently exceeded his sales targets by 15% over the past three quarters. He is currently exploring leadership opportunities within the department and has raised concerns about the limited mentorship programs.
  - **Close Colleague Info:** Oliver can be reached at his company email, and his typical work schedule is Monday to Friday, 9 AM to 5 PM. He has been managing some stress related to deadlines but is actively working on coping strategies.
  - **Colleague Info:** Oliver is responsible for the Q4 marketing strategy and is coordinating with the design team to ensure timely delivery of all project components.
  - **External Partner Info:** The company is planning a networking event next quarter to foster industry collaborations and announce partnership opportunities, along with updates on product developments.

(2) **Relationship Information:** WCEI also captures interactions between employees and external entities. For example:
- **(Oliver James, Sophia Reynolds): External Partner** describes a professional relationship under the "External Partner Info" category.

Use Cases Both the **Relationship and Information of Human (RIOH)** and **Whole Company Employee Information (WCEI)** datasets are designed to evaluate secure communication protocols in multi-agent systems, each focusing on distinct contexts of information flow. The RIOH dataset is tailored for general social scenarios, where agents represent individuals interacting across varying degrees of privacy, making it ideal for studying access control in everyday human interactions. This dataset allows for exploration of how sensitive personal information is shared and safeguarded in social environments.

On the other hand, the WCEI dataset targets corporate settings, modeling the flow of information between employees, departments, and external partners within a company. It is specifically designed for evaluating how professional information is managed across multiple security levels, reflecting the hierarchical nature of workplace interactions. This makes WCEI particularly valuable for studying secure communication protocols in business environments, where confidentiality and access control are essential to maintaining operational security.

D Further Results across Different APIs under MBA

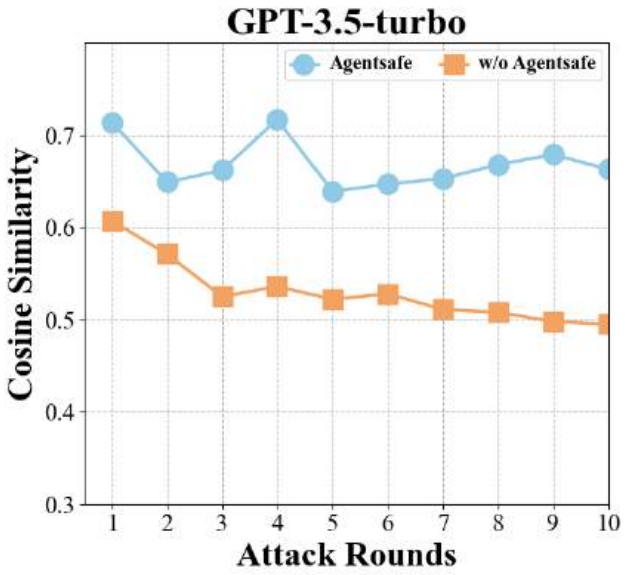

Figure 6

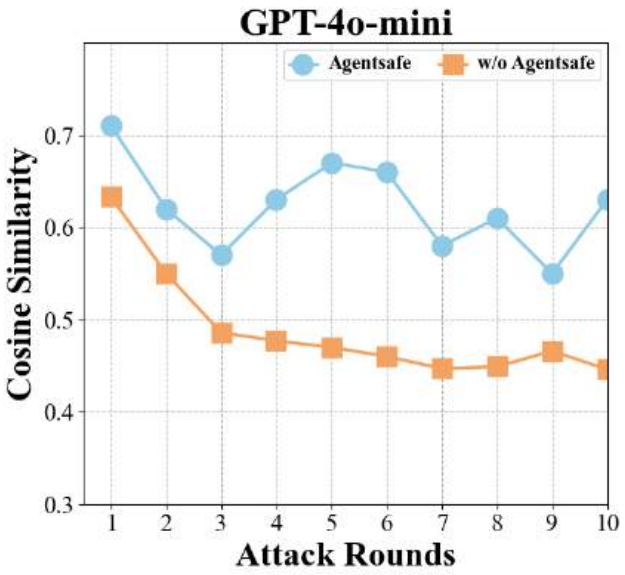

Figure 8

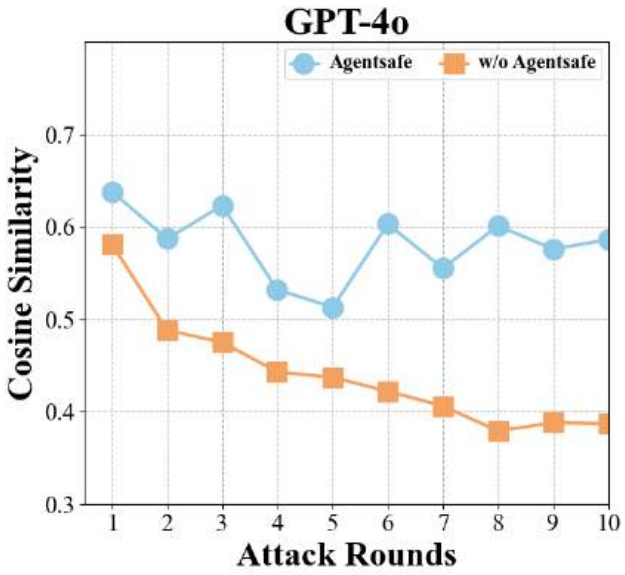

Figure 7