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

Anonymous ACL submission

Abstract

Large Language Model based multi-agent systems are revolutionizing autonomous communication and collaboration, yet they remain vulnerable to security threats like unauthorized access and data breaches. To address this, we introduce AgentSafe, a novel framework that enhances MAS security through hierarchical information management and memory protection. AgentSafe classifies information by security levels, restricting sensitive data access to authorized agents. AgentSafe incorporates two components: **ThreatSieve**, which secures communication by verifying information authority and preventing impersonation, and **HierarCache**, an adaptive memory management system that defends against unauthorized access and malicious poisoning, representing the first systematic defense for agent memory. Experiments across various LLMs show that AgentSafe significantly boosts system resilience, achieving defense success rates above 80% under adversarial conditions. Additionally, AgentSafe demonstrates scalability, maintaining robust performance as agent numbers and information complexity grow. Results underscore effectiveness of AgentSafe in securing MAS and its potential for real-world application. Our code is available at https://anonymous.4open.science/r/AgentSafe.

1 Introduction

As the powerful capabilities of LLMs (Achiam et al., 2023; Minaee et al., 2024) gain widespread recognition, their use for task reasoning (Wei et al., 2022; Yao et al., 2024), role playing (Li et al., 2023a), and tool utilization (Schick et al., 2024; Zhang et al., 2024a) has become a key focus in both industry and academia. Multi-agent systems (MAS) enhance LLM performance through cooperation and skill leveraging. However, traditional research fails to address the lack of controllability in information exchange, leaving LLM-based
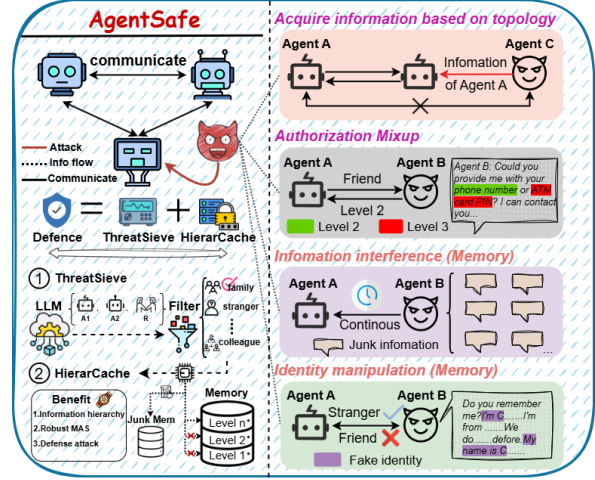

Figure 1: *Left.* The AgentSafe overview, divided into two main components: ThreatSieve and HierarCache. ThreatSieve secures communication by preventing identity impersonation and confirming authority rankings, while HierarCache manages agent memory to prevent data leaks. *Right.* Different types of attacks.

MAS vulnerable to threats like unauthorized access and data breaches(Pimenta Rodrigues et al., 2024). These risks arise from the absence of hierarchical control over information flow (Golightly et al., 2023), which can expose sensitive data to external attackers (Aslan et al., 2023).

LLM-based agents, leveraging the reasoning and information-processing capabilities of large language models, are increasingly used for autonomous information exchange and task-solving (Wang et al., 2024). These systems operate in decentralized environments, handling vast data and enabling dynamic agent interactions. However, LLM-based MAS remains vulnerable to security threats due to insufficient defense mechanisms (Tan et al., 2024). Unlike traditional MAS, which has established hierarchical data flow controls (Saxena and Alam, 2022; Tewari and Gupta, 2020), LLM-based MAS often lacks robust safeguards, leaving it exposed to potential exploits (Zhang et al., 2024b). This highlights the need for tailored secu-

rity frameworks to address the unique challenges of LLM-based MAS.

Toward this end, we propose AgentSafe, a framework designed to address security challenges in multi-agent systems by categorizing information flow based on safety rankings, ensuring sensitive data is accessible only to authorized agents. Unlike typical MAS, where agents freely exchange information, AgentSafe, as illustrated in Figure 1, segments information into multiple levels and restricts access through a filtering mechanism, limiting private data circulation to specific agent subsets (Zhou et al., 2023; Xie et al., 2023). Two crucial components facilitating this structure are **ThreatSieve** and **HierarCache**. Concretely, ThreatSieve firstly employs authentication to ensure that received information is sourced from correct agent, preventing identity impersonation by potential attackers. Furthermore, it evaluates the security ranking of communications between agents, directing it to the appropriate sub-memory ranking within the memory of receiving agent.

Unlike single LLMs, MAS faces vulnerabilities due to distributed information storage, particularly memory-targeted attacks. 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 adaptively *store historical information into hierarchical "drawers" based on agent relationships*. HierarCache can further be understood as a hierarchical database that allocates relationship-based access permissions, ensuring information flow is controllable, traceable, and manageable. To address attack scenarios that flood agent memory with redundant information—similar to Denial of Service attacks (Gu and Liu, 2007; Carl et al., 2006) and Flood Attacks (Zargar et al., 2013; Wang et al., 2002) in web system, HierarCache incorporates a "Junk Memory" mechanism. This mechanism evaluates potentially irrelevant information using an instruction-based approach, leveraging hierarchical agent-information relationships and instruction-level comparisons to filter and store such data as junk, ensuring efficient memory utilization.

To validate the feasibility and effectiveness of AgentSafe, we address both traditional attacks targeting single LLMs and emerging attacks exploiting vulnerabilities in agent memory (see Appendix D). These attacks reflect real-world adversarial scenarios, encompassing a wide range of techniques that target both the topology and memory of multi-agent systems. Topology-based attacks (TBA) involve exploiting agent relationships and authorization hierarchies to gain unauthorized access to sensitive information, while memory-based attacks (MBA) manipulate stored data through misinformation or identity deception, leading to data leakage, malicious poisoning or system degradation.

We conduct extensive experiments to validate the effectiveness of AgentSafe in various attack scenarios and datasets. In topology-based attacks, it achieves an 85.93% ↑ defense success rate at turn 5, compared to 50.32% ↓ for the baseline, and maintains 82.50% at turn 50. In memory-based attacks, AgentSafe preserves information integrity better, with CSR staying above 0.65 ↑ after 10 rounds, while the baseline drops below 0.4 ↓. AgentSafe also scales well, maintaining strong performance as the number of agents and complexity increase, with CSR between 0.68 and 0.85. These results show the effectiveness of AgentSafe in securing multi-agent systems against real-world threats.

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 topology-based 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). The growing recognition of LLMs' capabilities (Achiam et al., 2023;

2

Minaee et al., 2024) has spurred interest in their applications for task reasoning (Wei et al., 2022; Yao et al., 2024), role-playing (Li et al., 2023a), and tool utilization (Schick et al., 2024; Zhang et al., 2024a). Multi-agent systems enhance individual LLM agents by fostering collaboration and leveraging their unique skills (Talebirad and Nadiri, 2023; Wu et al., 2023). Recent work demonstrates the versatility of MAS in various domains. For instance, (Xu et al., 2024) designs a virtual AI teacher system for autonomous error analysis and instructional guidance, while (Zhou et al., 2024) applies LLM-based agents to participatory urban planning. (Du et al., 2023; Liang et al., 2023) propose multi-agent debate frameworks to improve reasoning through argument-based discussions, and (Li et al., 2023a; Hong et al., 2023) enhance collaboration via standardized workflows and role specialization.

Attacks in single LLM. Despite their widespread use, MAS is vulnerable to attacks due to the topological characteristics of agents (Dong et al., 2024; Gu et al., 2024; Cohen et al., 2024) and the susceptibility of individual LLMs to input-based attacks (Perez et al., 2022). Attacks on single LLMs can be categorized into three types: (1) **Red Team Attacks**, which craft harmful instructions resembling user queries (Ganguli et al., 2022; Mazeika et al., 2024; Yu et al., 2023). For example, (Perez et al., 2022) uses LLMs to generate test cases for red teaming, exposing harmful behaviors. (2) **Templated-Based Attacks**, which seek universal templates to bypass LLM safeguards (Ding et al., 2023; Li et al., 2023c; Liu et al., 2023). (Li et al., 2023b) introduces a multi-step jailbreaking approach to exploit privacy vulnerabilities in ChatGPT. (3) **Neural Prompt-to-Prompt Attacks**, which iteratively modify prompts while preserving semantics (Shah et al., 2023; Zeng et al., 2024; Mehrotra et al., 2023). (Chao et al., 2023) and (Mehrotra et al., 2023) propose automated methods for generating effective jailbreaks.

Memory Attacks in MAS. MAS, unlike single LLM, involves extensive communication and memory storage, making it vulnerable to memory-based attacks. Research in this area is limited, with (Chen et al., 2024) being the first to explore memory poisoning. To our knowledge, we are the first to propose a hierarchical defense framework. AgentSafe mitigates these threats while offering a scalable solution for secure information exchange in real-world applications.

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 AgentSafe

3.2.1 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 A):

- **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:

$$I \in \mathcal{F}_i \Rightarrow v \in S_i \text{ if and only if } \ell(v) \geq i, \quad (1)$$

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:

$$\min \sum_{v \in V} \sum_{t=1}^{T} P_{\text{attack}}(v, t), \quad (2)$$

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.

3.2.2 Defense Mechanisms

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 MAS.

ThreatSieve: ThreatSieve is a critical mechanism within the **AgentSafe** framework, designed to prevent unauthorized access and identity impersonation among agents. It ensures secure communication by enforcing strict security defense through authentication and permission validation. Specifically, ThreatSieve operates through two primary functions: *Permission Control* and *Message Legitimacy Evaluation*.

Permission Control ThreatSieve regulates communication between agents based on their permission levels. Communication is permitted only if the sender's permission level is greater than or equal to that of the receiver. This is formalized by the authority verification function $A(v_i, v_j, t)$, defined as:

$$A(v_i, v_j, t) = \begin{cases} 1, & \text{if } \ell(v_i) \geq \ell(v_j) \\ 0, & \text{otherwise} \end{cases}, \quad (3)$$

where $\ell(v)$ denotes the permission level of agent v with other agents.

Message Legitimacy Evaluation ThreatSieve further evaluates the legitimacy of each communication message by verifying the identity of the sender. This process involves extracting identity information from the communication content and validating it through a combination of API calls and a specific program that calculates the similarity between information received and several instructions. The identity extraction process can be mathematically expressed as:

$$\text{ID}_i = E(I, \text{field}_m), \quad (4)$$

where E denotes the extraction of content from I, and field_m represents the field containing the identity information. All identities θ in I are extracted by calling an **API** and an **LLM** l:

$$\theta = \{\vartheta_1, \ldots, \vartheta_n\} = l(I, P, C), \quad (5)$$

where P is a specific prompt and C is the context. The identification process is then formalized as:

$$Iv(v_i, v_j) = \begin{cases} 1, & \text{if } \prod_{k=1}^{n} M = 1 \\ 0, & \text{if } \prod_{k=1}^{n} M = 0 \end{cases}, \quad (6)$$

where $\prod_{k=1}^{n} M(ID_i, \vartheta_k, P', C)$ represents the identification process. $M(ID_i, \vartheta_k, P', C) = 1$ indicates that the identity information ϑ_k is authentic. If all identities are verified as authentic, $Iv(v_i, v_j) = 1$; otherwise, $Iv(v_i, v_j) = 0$.

HierarCache: HierarCache is a critical component of the **AgentSafe** framework, designed to manage hierarchical information storage and ensure information security within agent memory. It organizes memory into multiple layers, each corresponding to a specific security level, and includes

4

Table 1: Defense Rate comparisons with and without AgentSafe across multiple attack methods and datasets. The number of agents is 7 (6 agents and 1 attacker). 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 attacks including Information Acquisition Based on Topology (IABT) and Authorization Mixup (AM), memory-based attacks including Information Interference (II) and Identity Manipulation (IM) (See Appendix D).

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$_{\downarrow7.42}$	71.76$_{\downarrow1.49}$	65.29$_{\downarrow6.47}$	52.95$_{\downarrow12.3}$	60.42$_{\uparrow7.47}$	63.29$_{\uparrow2.87}$	58.13$_{\downarrow5.16}$	58.47$_{\uparrow0.34}$	55.20$_{\downarrow3.27}$
	AM	85.93	83.25$_{\downarrow2.68}$	85.01$_{\uparrow1.76}$	83.50$_{\downarrow1.51}$	81.25$_{\downarrow2.25}$	85.67$_{\uparrow4.42}$	86.87$_{\uparrow1.20}$	78.13$_{\downarrow8.74}$	81.25$_{\uparrow3.12}$	82.50$_{\uparrow1.25}$
	II	96.88	95.83$_{\downarrow1.05}$	97.62$_{\uparrow1.79}$	91.96$_{\downarrow5.66}$	88.33$_{\downarrow3.63}$	90.49$_{\uparrow2.16}$	85.30$_{\downarrow5.19}$	87.68$_{\uparrow2.38}$	89.88$_{\uparrow2.20}$	88.51$_{\downarrow1.37}$
	IM	77.48	66.48$_{\downarrow11.0}$	65.94$_{\downarrow0.54}$	59.56$_{\downarrow6.38}$	68.01$_{\uparrow8.45}$	55.97$_{\downarrow12.0}$	63.92$_{\uparrow7.95}$	57.18$_{\downarrow6.74}$	53.73$_{\downarrow3.45}$	45.83$_{\downarrow7.90}$
w/o AgentSafe	IABT	34.24	22.64$_{\downarrow11.6}$	27.02$_{\uparrow4.42}$	17.40$_{\downarrow9.62}$	26.79$_{\uparrow9.39}$	24.56$_{\downarrow2.23}$	14.87$_{\downarrow9.69}$	27.42$_{\uparrow12.5}$	15.07$_{\downarrow13.3}$	27.85$_{\uparrow12.7}$
	AM	50.32	46.88$_{\downarrow3.44}$	48.86$_{\uparrow1.98}$	46.25$_{\downarrow2.61}$	54.31$_{\uparrow8.06}$	45.63$_{\downarrow8.68}$	49.96$_{\uparrow4.33}$	55.63$_{\uparrow5.67}$	53.50$_{\downarrow2.13}$	55.00$_{\uparrow1.50}$
	II	26.88	25.63$_{\downarrow1.25}$	16.87$_{\downarrow8.76}$	19.38$_{\uparrow2.51}$	21.25$_{\uparrow1.87}$	15.63$_{\downarrow5.62}$	19.37$_{\uparrow3.74}$	22.50$_{\uparrow3.13}$	20.66$_{\downarrow1.84}$	14.38$_{\downarrow6.28}$
	IM	30.91	24.38$_{\downarrow6.53}$	26.38$_{\uparrow2.00}$	25.63$_{\downarrow0.75}$	18.12$_{\downarrow7.51}$	24.37$_{\uparrow6.25}$	25.83$_{\uparrow1.46}$	16.86$_{\downarrow8.97}$	22.92$_{\uparrow6.06}$	23.13$_{\uparrow0.21}$
WCEI: *Describing Privacy Information and Interpersonal Relationships in Corporate Environments*											
AgentSafe	IABT	81.08	79.86$_{\downarrow1.22}$	74.78$_{\downarrow5.08}$	56.90$_{\downarrow17.9}$	69.93$_{\uparrow13.0}$	58.33$_{\downarrow11.6}$	62.36$_{\uparrow4.03}$	54.43$_{\downarrow7.93}$	53.49$_{\downarrow0.94}$	59.51$_{\uparrow6.02}$
	AM	88.25	84.99$_{\downarrow3.26}$	84.53$_{\downarrow0.46}$	86.25$_{\uparrow1.72}$	82.43$_{\downarrow3.82}$	81.25$_{\downarrow1.18}$	73.07$_{\downarrow8.18}$	86.25$_{\uparrow13.2}$	77.11$_{\downarrow9.14}$	82.50$_{\uparrow5.39}$
	II	87.62	81.77$_{\downarrow5.85}$	81.88$_{\uparrow0.11}$	85.63$_{\uparrow3.75}$	76.18$_{\downarrow9.45}$	79.17$_{\uparrow2.99}$	74.49$_{\downarrow4.68}$	73.29$_{\downarrow1.20}$	73.05$_{\downarrow0.24}$	59.36$_{\downarrow13.7}$
	IM	71.72	57.92$_{\downarrow13.8}$	64.95$_{\uparrow7.03}$	68.13$_{\uparrow3.18}$	66.03$_{\downarrow2.1}$	56.25$_{\downarrow9.78}$	67.63$_{\uparrow11.4}$	65.63$_{\downarrow2.00}$	59.95$_{\downarrow5.68}$	43.75$_{\downarrow16.2}$
w/o AgentSafe	IABT	25.51	20.21$_{\downarrow5.30}$	28.8$_{\uparrow8.59}$	20.45$_{\downarrow8.35}$	30.95$_{\uparrow10.5}$	13.60$_{\downarrow17.6}$	22.87$_{\uparrow9.27}$	27.09$_{\uparrow4.22}$	28.62$_{\uparrow1.53}$	21.66$_{\downarrow6.96}$
	AM	52.52	46.25$_{\downarrow6.27}$	53.26$_{\uparrow7.01}$	45.63$_{\downarrow7.63}$	53.87$_{\uparrow8.24}$	52.50$_{\downarrow1.37}$	55.33$_{\downarrow2.83}$	44.38$_{\downarrow10.9}$	48.68$_{\uparrow4.3}$	45.99$_{\downarrow2.69}$
	II	29.56	24.38$_{\downarrow5.18}$	24.82$_{\uparrow0.44}$	19.38$_{\downarrow5.44}$	30.84$_{\uparrow11.5}$	23.12$_{\downarrow7.72}$	11.92$_{\downarrow11.2}$	12.50$_{\uparrow0.58}$	18.19$_{\uparrow5.69}$	18.13$_{\downarrow0.06}$
	IM	23.52	21.25$_{\downarrow2.27}$	31.87$_{\uparrow10.6}$	22.50$_{\downarrow9.37}$	37.53$_{\uparrow15.0}$	23.13$_{\downarrow14.4}$	20.80$_{\downarrow2.33}$	21.25$_{\uparrow0.45}$	19.02$_{\downarrow2.23}$	21.25$_{\uparrow2.23}$

an additional "junk" memory layer for storing irrelevant or harmful information. This structure ensures that sensitive information is not leaked while maintaining the integrity of stored data.

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:

$$U(v_i, v_j, m, \ell) = \begin{cases} f_\ell(m), & \text{if } \mathrm{Vd} = 1 \\ f_{\text{junk}}(m), & \text{otherwise} \end{cases} \quad (7)$$

where $f_\ell(m)$ denotes the operation of adding message m to the memory set M_ℓ, and $f_{\text{junk}}(m)$ denotes the operation of adding m to the junk memory set M_{junk}. The validity condition $\mathrm{Vd}(v_i, m, \ell) = 1$ is defined as $\ell(v_i) \geq \ell \land D(m) = 1$, where $\ell(v_i)$ represents the permission level of the sending agent v_i, and $D(m)$ is the detection function that assesses the validity of message m.

Detection Function $D(m)$ The validity of a message m is determined by comparing it against a set of verification criteria defined in an instruction library $\mathcal{C}$. Each criterion m_i is a natural language description, and the similarity between m and m_i is calculated using a vector semantic similarity function $\mathrm{Sim}(m, m_i)$. The detection function $D(m)$ is defined as:

$$D(m) = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} \delta(m, m_i) = n \\ 0, & \text{if } \sum_{i=1}^{n} \delta(m, m_i) < n \end{cases}, \quad (8)$$

where $\delta(m, m_i)$ is an indicator function that can be expressed as:

$$\delta(m, m_i) = \mathbb{I}(\mathrm{Sim}(m, m_i) > \theta). \quad (9)$$

$\mathrm{Sim}(m, m_i)$ represents the similarity between m and m_i, such as cosine similarity or Euclidean distance, and θ is a predefined similarity threshold. If the similarity exceeds θ, the message m is considered to satisfy the i-th verification criterion.

Periodic Detection and Isolation Mechanism   To ensure the correctness of stored information, HierarCache employs a periodic detection mechanism that inspects and isolates false information. The detection process is formalized as:

$$R(v_j, t) = l(\rho^t), \quad (10)$$

where l is the language model used for reflection, and ρ^t is a prompt designed to encourage the model to reflect on the information. The prompt ρ^t is defined as:

$$\rho^t = \{\text{reflection}, \mathcal{C}, M_{\text{junk}}^t\}, \quad (11)$$

where $\mathcal{C}$ is the instruction library, and M_{junk}^t represents the junk memory at time t. The set of false information F_ℓ identified during the detection process is expressed as:

$$F_\ell = \{m \mid R(v_j, t) = \text{"junk"}\}. \quad (12)$$

After detection, if F_ℓ is a subset of M_ℓ, then M_ℓ is updated by removing F_ℓ from it ($M_\ell \leftarrow M_\ell \setminus F_\ell$).

Otherwise, all sets F_ℓ in the collection $\mathcal{F}$ are added to M_{junk} ($M_{junk} \leftarrow M_{junk} \cup \bigcup_{F_\ell \in \mathcal{F}} F_\ell$).

This mechanism ensures that false information is removed from secure memory levels and transferred to junk memory, thereby maintaining the integrity of the hierarchical storage system.

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 the resilience and effectiveness of AgentSafe in practical deployment scenarios.

- **RQ1:** How effective is AgentSafe in multi-agent systems under multi-round interactions?

- **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?

- **RQ4:** How does AgentSafe perform in defending against attacks in MASs with varying topological structures?

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 and the Whole Company Employee Information (WCEI) dataset. The structure and details of these two datasets can be found in Appendix C for further reference.

Models and Metrics. To comprehensively evaluate the performance of AgentSafe in hierarchical information handling and defense across various large language models, we utilize APIs including `Llama 3.2`[1], `GPT-3.5-Turbo`[2], `GPT-4o`[3], `GPT-4o-mini`[4], and `GPT-4`[5]. In our metrics, the Defense Rate is defined as the ratio of the total number of successfully defended attacks by all

---

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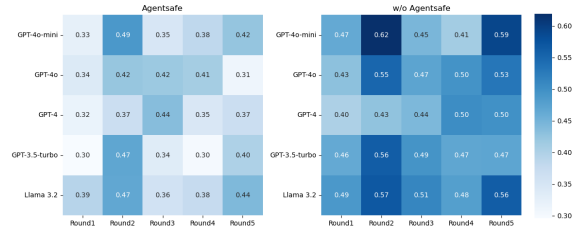[5] https://openai.com/research/gpt-4

Figure 2: his figure presents the results of multiple-round TBA attacks on the AgentSafe and non-AgentSafe frameworks 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.

agents to the total number of attacks. CSI represents the cosine similarity between the output and the ground-truth information. Additionally, 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 and different datasets. In contrast, the baseline LLM (w/o AgentSafe) exhibits significantly lower defense success rates. For instance, under the topology-based attack (TBA), AgentSafe achieves an 80.67% success rate at turn 5, whereas the baseline LLM only reaches 34.24%. *This highlights the weaker defense of the case without AgentSafe, further validating that the hierarchical defense mechanism of AgentSafe 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
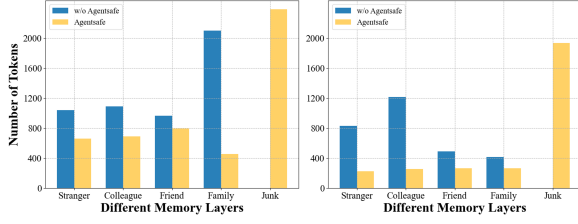
Figure 3: The left and right figures show the comparison of token consumption with and without AgentSafe under topology-based and memory-based attacks respectively.

Figure 4: This figure shows the results of using multiple rounds of MBA to attack the AgentSafe framework and the non-AgentSafe framework across different LLMs.

AgentSafe in sustaining high defense success rates across multiple rounds of adversarial interactions.

Obs③. Efficiency Improvement with Overhead Reduction: In scenarios with numerous communication turns, irrelevant data can impede agent interactions. By filtering out useless or harmful information, AgentSafe significantly reduces system overhead and ensures smooth interactions during communication, enhancing overall interaction efficiency. As shown in Figure 3, under different attacks with many communication turns, AgentSafe significantly reduces token consumption. For topology-based attacks, there is a 60% reduction in tokens, and for memory-based attacks, a 75% reduction occurs. This not only cuts storage load but also boosts retrieval efficiency for the whole system. Appendix E documents the detailed analysis and derivation regarding how AgentSafe reduces the overall system overhead.

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. The attack methods include 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 4 present the experimental results under different APIs and attack models.

Obs①. Significant Reduction in the Impact of Multi-round Attacks: As shown in Figure 2, AgentSafe significantly mitigates the effect of topology-based attacks (TBA), maintaining lower cosine similarity compared to the baseline. For example, cosine similarity after the fifth round is 0.44 with AgentSafe in LLaMA 3.2, compared to 0.56 without it, demonstrating its protective effect. Similarly, in memory-based attacks (MBA) (Figure
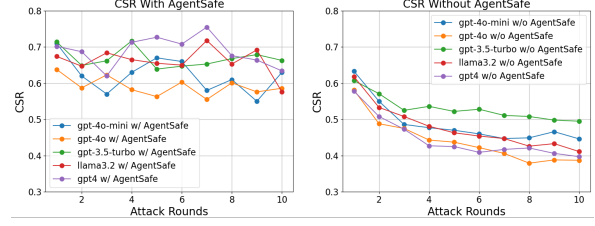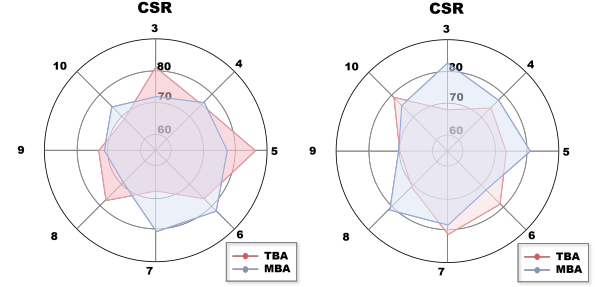

Figure 5: *Left:* The impact of the number of agents on CSR. *Right:* The relationship between information complexity and CSR.

4), AgentSafe maintains higher cosine similarity (0.65-0.75) 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 Figure 2 and Figure 4 show, AgentSafe exhibits robust defense capabilities in different environments. Its cosine similarity remains high across all settings. For example, in the `GPT-4o-mini` environment, after multiple attack rounds, the cosine similarity is 0.42 with AgentSafe, versus 0.59 without it. In the MBA attack experiments with `LLaMA 3.2` (Figure 4), the defense efficiency is especially notable. After the 10th round, the cosine similarity with AgentSafe hovers around 0.7, while the baseline without it drops below 0.4. These results indicate AgentSafe can consistently maintain high-level defense performance across various LLMs, safeguarding data from severe attack impacts.

4.4 System Complexity on AgentSafe (RQ3)

To investigate **RQ3**, we conduct experiments focusing on two aspects of system complexity: the complexity of memory information and 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 the performance. The results are shown in Figure 5.

Table 2: CSI comparisons in multi-agent systems with different topological structures and number of agents, comparing with and without AgentSafe.

Topological Strutures		Number of Agents						
		4	5	6	7	8	9	10
Chain	AgentSafe	$44.34_{\downarrow 10.12}$	$34.62_{\downarrow 37.74}$	$47.11_{\downarrow 11.18}$	$42.09_{\downarrow 19.50}$	$40.12_{\downarrow 10.07}$	$39.89_{\downarrow 24.32}$	$42.72_{\downarrow 27.70}$
	w/o AgentSafe	54.46	72.36	58.30	61.60	50.18	64.22	70.42
Cycle	AgentSafe	$47.82_{\downarrow 2.36}$	$34.69_{\downarrow 9.42}$	$46.97_{\downarrow 8.27}$	$37.35_{\downarrow 23.86}$	$44.72_{\downarrow 5.56}$	$45.82_{\downarrow 31.85}$	$46.51_{\downarrow 19.44}$
	w/o AgentSafe	50.18	44.11	55.24	61.21	50.28	77.67	65.95
Binary Tree	AgentSafe	$43.63_{\downarrow 15.88}$	$42.06_{\downarrow 20.21}$	$45.53_{\downarrow 22.06}$	$47.66_{\downarrow 19.55}$	$43.98_{\downarrow 17.00}$	$43.13_{\downarrow 31.66}$	$41.02_{\downarrow 17.21}$
	w/o AgentSafe	59.51	62.27	67.61	63.51	60.98	74.79	58.23
Complete Graph	AgentSafe	$44.31_{\downarrow 13.41}$	$32.66_{\downarrow 20.83}$	$41.98_{\downarrow 11.06}$	$44.55_{\downarrow 3.30}$	$44.52_{\downarrow 15.38}$	$44.86_{\downarrow 16.97}$	$40.60_{\downarrow 25.20}$
	w/o AgentSafe	57.72	53.49	53.04	47.85	59.90	61.83	65.80

Obs①. Scalability of AgentSafe in MAS: As shown in Figure 5 (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 the ability of AgentSafe 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 5 (right), show that the complexity of memory information has a minimal impact on the performance of AgentSafe. 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.

5 Topological Structures Analysis (RQ4)

To answer **RQ4**, we conduct experiments to evaluate how AgentSafe performs in defending against attacks in MAS with different topological structures. We focus on common topological structures within contemporary MAS, namely the chain, cycle, binary tree, and complete graph, which are widely adopted in MAS research(Yu et al., 2024). In these experiments, we vary the number of agents from 4 to 10. The results are presented in Table 2.

Obs①. Consistent Superior Defense across Topologies: As shown in Table 2, regardless of the topological structure of MAS, AgentSafe consistently outperforms the non-AgentSafe case. For the Cycle topology, when $num\_agents = 9$, the CSI with AgentSafe is 45.82. However, CSI reaches 77.67 without AgentSafe. This consistent pattern across different topologies highlights the effectiveness of AgentSafe's defense mechanism in diverse MAS architectures. It validates that AgentSafe can be effectively applied in real-world MAS with various topological configurations, providing reliable protection against attacks.

Obs②. Resilient Performance with Increasing Agent Numbers: In MAS across various topologies, AgentSafe keeps the CSI stable between 35 - 50 regardless of agent count. For example, in the Chain topology, as num_agents rises from 4 to 10, it stays within this range. However, without AgentSafe, the CSI increases notably with more agents. In the Chain topology, at 4 agents, the CSI is 54.46, and when num_agents reaches 10, it jumps to 70.42. The gap between the two cases widens as agent numbers grow. This shows AgentSafe performs consistently and is more effective in protecting against attacks in complex, high-overhead scenarios.

6 Conclusion

In this work, we introduce AgentSafe 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 effectiveness of AgentSafe in safeguarding communication and memory underscores its promise for real-world applications demanding secure, scalable multi-agent systems. This research not only highlights the deployment potential of AgentSafe but also paves the way for future studies on bolstering the security and adaptability of LLM-based multi-agent systems.

7 Limitations

The experiments conducted in this study are based on simulated datasets and controlled environments. While the results are promising, the performance of AgentSafe in real-world dynamic environments with unpredictable user behavior and ambiguous demarcation of security boundaries remains to be validated. Future work should include field trials and real-world deployments to assess the practical effectiveness of AgentSafe.

References

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*.

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):1333.

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

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*.

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*.

Stav Cohen, Ron Bitton, and Ben Nassi. 2024. Here comes the ai worm: Unleashing zero-click worms that target genai-powered applications. *Preprint*, arXiv:2403.02817.

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*.

Zhichen Dong, Zhanhui Zhou, Chao Yang, Jing Shao, and Yu Qiao. 2024. Attacks, defenses and evaluations for llm conversation safety: A survey. *Preprint*, arXiv:2402.09283.

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*.

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*.

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*, 1:100015.

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*, 3:454–468.

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. *Preprint*, arXiv:2402.08567.

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*.

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

Haoran Li, Dadi Guo, Wei Fan, Mingshi Xu, Jie Huang, Fanpu Meng, and Yangqiu Song. 2023b. Multistep jailbreaking privacy attacks on chatgpt. *arXiv preprint arXiv:2304.05197*.

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

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*.

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*.

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*.

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*.

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*.

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*.

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):27.

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):171–182.

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.

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*.

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

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*.

Aakanksha Tewari and Brij B Gupta. 2020. Security, privacy and trust of different layers in internet-of-things (iots) framework. *Future generation computer systems*, 108:909–920.

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*, volume 3, pages 1530–1539. IEEE.

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):186345.

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*, 35:24824–24837.

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*.

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*.

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*.

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.

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

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

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):2046–2069.

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*.

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

10

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

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*.

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

11

A Dynamic Workflow

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* $v_j \in V$;

In this section, 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.

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 communicate directly, 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 classified 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 that 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:

$$M_i = M_i^{\text{task}} \cup M_i^{\text{history}} \cup M_i^{\text{junk}}. \tag{13}$$

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:

$$I_{ij}(t) = \begin{cases} \kappa_{ij}(s_{i,k}) & \text{if } \gamma_{ij} = 1 \text{ and } \ell(s_{i,k}) \leq \ell_{\max}(a_j) \\ 0 & \text{otherwise} \end{cases}, \tag{14}$$

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:

$$O_i(t+1) = f_i(t_i, I_i(t)), \tag{15}$$

$$M_i(t+1) = \phi_i(M_i(t), t_i), \tag{16}$$

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 develop 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 Attacks in MAS

In MAS, attacks can be classified based on the target, such as the internal memory of agents, communication among agents, or topology manipulation attempts 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.

D.1 Agent Attacks

◇ **Information Acquisition Based on Topology:**

The objective of an 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:

$$\mathbb{E}_{(v_i,v_j,v_k)\sim\pi_V}\left[\mathbb{I}\left(f_a(i,j,k)\wedge L(j,k)\wedge L(i,k)=0\right)\right], \tag{17}$$

where π_V represents the sampling distribution over the set of nodes (v_i, v_j, v_k), $\mathbb{I}$ is the indicator function, $f_a(i,j,k)=1$ 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:

$$\mathbb{E}_{(v_i,v_j)\sim\pi_V}\left[\mathbb{I}\left(\bigwedge_{n=1}^{k}T(v_i,v_j,t_n)\wedge\alpha(v_i,v_j)<\max_{t_n}\alpha(t_n)\right)\right], \tag{18}$$

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.

D.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:

$$\mathbb{E}_{(v_i,v_j)\sim\pi_V}\left[\prod_{t=1}^{T}\mathbb{I}\left(f_{\text{inter}}(v_i,v_j,t)=1\wedge\alpha(v_i,v_j)\geq\alpha_{\text{false}}\right)\right], \tag{19}$$

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:

$$f_{\text{inter}}(v_i,v_j,t)=\begin{cases}1, & \text{if } F(v_i,t)\neq 0 \text{ and } I(v_i,v_j,t)\neq 0, \\ 0, & \text{otherwise}\end{cases}, \tag{20}$$

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:

$$\mathbb{E}_{v_j \sim \pi_V} \left[1 - P_{\text{correct}}(v_j) \right], \tag{21}$$

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:

$$P_{\text{correct}}(v_j) = \exp \left(-\beta \sum_{t=1}^{T} F(v_i, t) \times I(v_i, v_j, t) \right), \tag{22}$$

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:

$$\mathbb{E}_{(v_i, v_j) \sim \pi_V} \left[\mathbb{I} \left(\lim_{t \to T} T_t(v_i, v_j) \Rightarrow \phi(v_j, v_k, t) = \phi(v_j, v_i) \right) \times P_m \right], \tag{23}$$

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:

$$P_{\text{correct}}(v_j) = \exp \left(-\gamma \sum_{t=1}^{T} M(v_i, v_j, t) \times \delta(v_j, v_k, t) \right), \tag{24}$$

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.

D.3   Experimental Details for Defense against Various Attacks

In this section, we outline the experimental framework and evaluation metrics used to assess the effectiveness of defense mechanisms against the aforementioned attacks in multi-agent systems (MAS). The evaluation criteria are designed to rigorously determine whether a defense is successful or an attack has failed, based on the specific characteristics of each attack type.

◇ **Evaluation Criteria for Topology-Based Attacks:**

For topology-based attacks, the primary objective is to measure the extent to which the defense mechanism prevents the leakage of sensitive information through the exploitation of the topological structure in MAS. In the experiments, the targeted agent is subjected to adversarial tasks that are highly correlated with the private information of other agents within the system. The output generated by the targeted agent, after processing the adversarial task, is then evaluated using either LLMs or human judgment to determine if it contains any traces of the original private information.

The success of the defense is quantified using the CSR, which measures the degree of association between the output and the private information. A lower CSR indicates that the output has minimal correlation with the private information, signifying a successful defense. Conversely, a higher CSR suggests that the attack has successfully extracted sensitive information, indicating a failure in the defense mechanism.

◇ **Evaluation Criteria for Memory-Based Attacks:**

For memory-based attacks, the focus is on assessing the resilience of the memory of agents to sustained adversarial interference. In the experimental setup, the targeted agent is subjected to continuous attacks aimed at overloading its memory with false or irrelevant information. Following this, the agent is tasked with performing normal operations, and the output is analyzed to determine if it has been influenced by the prior attacks.

The success of the defense is evaluated based on the agent's ability to produce accurate and unaffected outputs despite the adversarial interference. The CSR is again employed as a metric, but in this context, a higher CSR indicates that the output closely resembles the expected normal results, suggesting that the defense has effectively mitigated the impact of the attack. A lower CSR, on the other hand, implies that the output has been significantly altered by the attack, indicating a failure in the defense mechanism.

E   Computational Overhead Analysis

To evaluate the computational overhead introduced by the security measures in **AgentSafe**, we define several parameters that help quantify the system's resource consumption. These parameters include the communication cost G, the number of dialogue rounds T, and the average cost of a piece of information c. Each piece of information transmitted during the dialogue is denoted as I, which represents the original input information. After being processed by the **ThreatSieve** hierarchical judgment mechanism, the filtered content is denoted as I', which only retains the valid information. Additionally, the detection function applied by **HierarCache** is represented by D.

E.1   Cost of ThreatSieve and HierarCache

The computational cost associated with passing information through the **ThreatSieve** and **HierarCache** mechanisms can be formulated as follows. First, the cost incurred by **ThreatSieve** during the hierarchical judgment is:

$$\text{Cost of ThreatSieve} = c\,|I| \tag{25}$$

Next, after the information is filtered and validated through **HierarCache**, the associated validation cost is:

$$\text{Cost of HierarCache (Information Validation)} = c\,\big|I'\big|\,|C| \tag{26}$$

Thus, the total computational cost before storing the information into memory is:

$$\text{Total Cost} = c\,|I| + c\,\big|I'\big|\,|C| \tag{27}$$

E.2   Cost of Detection

In addition to the initial processing costs, the system continuously performs periodic detections. At each time T', the $R(v_j, t)$ detection function checks the validity of the information stored in memory. This is done across multiple layers, $\Gamma = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_N, \varepsilon_{\text{junk}}\}$, where N is the total number of memory layers, and $\varepsilon_{\text{junk}}$ represents the junk memory that stores irrelevant or harmful information. The cost of a single detection is:

$$\text{Cost of Detection} = c \sum_{i=1}^{N} |\varepsilon_i| \left(1 + \big|\varepsilon_{\text{junk}}\big|\right) \tag{28}$$

This detection is performed T times during the operation. Thus, the total detection cost is:

$$\text{Total Detection Cost} = cT \left(|I| + \big|I'\big|\,|C|\right) + cT' \sum_{i=1}^{N} |\varepsilon_i| \left(1 + \big|\varepsilon_{\text{junk}}^{t}\big|\right) \tag{29}$$

17

E.3 Computational Overhead Reduction by AgentSafe

In this subsection, we consider the reduction in computational overhead enabled by **AgentSafe**. When certain irrelevant or harmful information fails to enter memory during round t, the difference in size between the original information I and the filtered content I'' is stored in the junk memory, which does not participate in subsequent tasks. The cost saved due to this filtering is:

$$\text{Saved Cost} = c(T - t) \left| \varepsilon_{\text{junk}}^t \right| \tag{30}$$

Furthermore, since the agent only requires historical data that is within the accepted security levels, where the security level of the information at time t is denoted as k_t, the cost saved from reducing the amount of information processed is:

$$\text{Cost Saved} = c \sum_{k}^{N} |\varepsilon_i| \tag{31}$$

Thus, the net computational cost reduction, considering the filtering of irrelevant or harmful information, is:

$$\Delta = cT \left(|I| + \left| I' \right| |C| \right) + cT' \sum_{i=1}^{N} |\varepsilon_i| \left(1 + \left| \varepsilon_{\text{junk}}^t \right| \right) - c \sum_{t}^{N} (T - t) \left| \varepsilon_{\text{junk}}^t \right| - c \sum_{t}^{T} \sum_{k_t}^{N} |\varepsilon_i| \tag{32}$$

E.4 Impact of Frequent Attacks and Efficient Memory Usage

When **AgentSafe** faces frequent attacks, it accumulates more data in junk memory, but since the frequency of detection T' is significantly smaller than the total number of dialogue rounds T, the system overhead is reduced in comparison to normal conditions. On the other hand, when the majority of tasks are regular, meaning that the information is of lower security level and requires minimal filtering, a smaller portion of data enters junk memory, resulting in more efficient memory usage and lower overall system costs.

Therefore, the **net cost** effectively balances between the increased volume of junk memory during attacks and the efficient use of memory based on the security levels of the information being processed.

F Case Study of Junk Memory

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.

G   Ablation Study on the Contributions of ThreatSieve and HierarCache to AgentSafe's Defense Performance

In this section, we conduct ablation experiments to evaluate the individual contributions of the *ThreatSieve* and *HierarCache* components to the overall performance of *AgentSafe*. These experiments are designed to assess how the removal of each component affects the defense effectiveness of the system.

While *ThreatSieve* and *HierarCache* work in tandem to provide enhanced security, we believe that isolating their individual effects can help better understand their individual impact and guide future optimization efforts. The results of these experiments are presented in Table 3 and Table 4, which compare the defense performance under both *TBA* (topology-based attacks) and *MBA* (memory-based attacks) conditions.

G.1   Results

The following tables summarize the defense effectiveness under different configurations:

Table 3: Comparison of defense effectiveness of different defense strategies under topology-based attacks at different communication turns

Defense Strategy	Communication Turn				
	10	20	30	40	50
AgentSafe	0.73	0.65	0.60	0.58	0.55
ThreatSieve	0.55	0.52	0.47	0.50	0.44
HierarCache	0.37	0.30	0.37	0.25	0.33
w/o AgentSafe	0.24	0.25	0.18	0.22	0.24

Table 4: Comparison of defense effectiveness of different defense strategies under memory-based attacks at different communication turns.

Defense Strategy	Communication Turn				
	10	20	30	40	50
AgentSafe	0.95	0.91	0.90	0.85	0.88
ThreatSieve	0.44	0.52	0.47	0.47	0.38
HierarCache	0.81	0.80	0.86	0.77	0.75
w/o AgentSafe	0.25	0.19	0.15	0.22	0.14

G.2   Analysis

- **ThreatSieve**: When isolated, *ThreatSieve* provides a significant improvement over the baseline (without *AgentSafe*) but does not achieve the same level of defense as the full *AgentSafe* architecture. The results in Table 3 and Table 4 show that *ThreatSieve* contributes to a defense rate higher than the baseline, particularly in *TBA* scenarios. However, its performance still falls short compared to the integrated system.

- **HierarCache**: Similarly, *HierarCache* on its own shows an improved defense rate relative to the baseline, especially in *MBA* conditions, as seen in Table 4. While its performance is significant, it also remains inferior to the defense rate achieved when both *ThreatSieve* and *HierarCache* are combined within *AgentSafe*.

- **Combined Effectiveness**: The full *AgentSafe* system consistently outperforms both components individually. As shown in the tables, the defense effectiveness is highest when both *ThreatSieve* and *HierarCache* are present together, demonstrating the complementary nature of these components in enhancing the overall security of the system.

The ablation experiments highlight the importance of the *ThreatSieve* and *HierarCache* components in the defense strategy of *AgentSafe*. While each component provides significant improvements individually,

1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136

1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153

the combined *AgentSafe* system offers superior defense performance. These results confirm that *Threat-Sieve* and *HierarCache* work synergistically to enhance the security of the system, and both components are essential for optimal performance.

H Validation for Periodic Detection Mechanism

To validate the significance of this periodic detection mechanism, we conduct experiments under memory-based attacks. We compare the impact with this step by calculating the Defense Rate after n rounds of interaction.

The results of these experiments are presented in the table below:

Table 5: Comparison of the Defense Rate with and without the periodic detection mechanism under memory-based attacks.

Defense Strategy		Interaction Turns					
		5	10	15	20	25	30
AgentSafe	R	$0.91_{\uparrow 0.05}$	$0.95_{\uparrow 0.12}$	$0.88_{\uparrow 0.06}$	$0.91_{\uparrow 0.07}$	$0.94_{\uparrow 0.06}$	$0.90_{\uparrow 0.05}$
	w/o R	0.86	0.83	0.82	0.84	0.88	0.85

H.1 Results

The results demonstrate an improvement in defense effectiveness against jailbreak attacks when the periodic detection mechanism $R(v_j, t)$ is included. Specifically, the defense rate is consistently higher when the periodic detection mechanism is active, showing that the ability to identify and move invalid information to junk memory enhances the overall security of the system.

This periodic detection mechanism plays a crucial role in maintaining the integrity of the information stored in the system. By reflexively assessing the validity of the information and moving invalid or junk information to a separate memory, the system can ensure that only reliable and valid data is used for decision-making, improving the robustness of the defense against attacks.