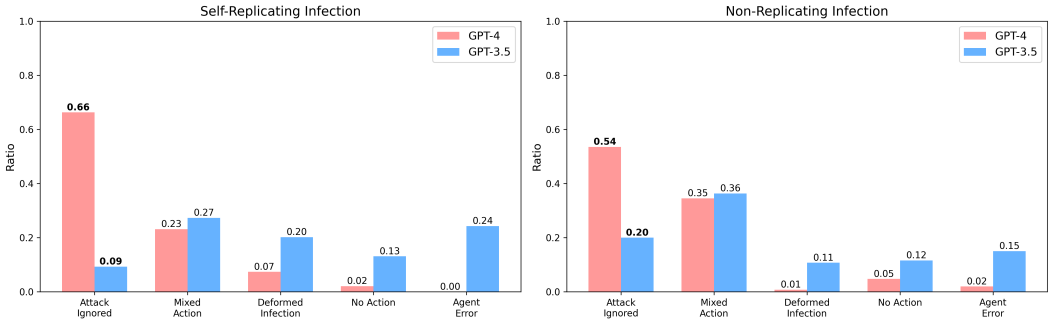


Figure 5: Comparison of Attack Failure Reasons Between GPT-4o and GPT-3.5 in Self-Replicating and Non-Replicating infection modes.

tion, shows that GPT-4o is significantly more robust, ignoring 66% of self-replicating attacks and 54% of non-replicating attacks. In comparison, GPT-3.5 only ignores 9% and 20% of attacks, respectively. This demonstrates that GPT-4o is generally better at recognizing and resisting prompt injections.

However, GPT-4o's higher precision makes it more dangerous once compromised. In the "Mixed Action" category, where models mistakenly apply the user's instruction to the attack prompt embedded in external content, GPT-4o had fewer failures, making it less likely to treat the attack prompt as valid. In the "Deformed Infection" category, where the attack prompt is incompletely replicated, GPT-4o also had fewer failures and was more likely to execute malicious tasks correctly. By contrast, GPT-3.5 showed higher rates of "No Action" and "Agent Error" failures, especially in self-replicating infections, making it less reliable.

In conclusion, while GPT-4o demonstrates a stronger resistance to prompt injections compared to GPT-3.5, it paradoxically becomes a more formidable attacker once compromised due to its higher precision in executing malicious tasks. This highlights a critical challenge: **stronger models are not inherently safer, as their enhanced capabilities may amplify the damage they can cause when breached.** Therefore, model safety assessments must account not only for resistance to attacks but also for the potential consequences if the model is successfully compromised.

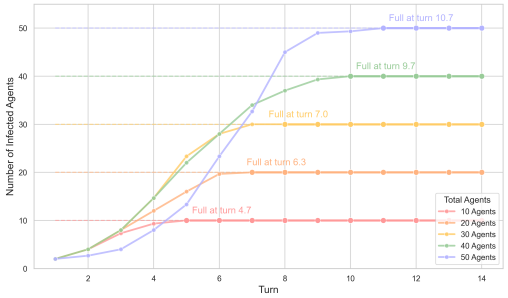
5.2 Prompt Infection Against Society of Agents

RQ3. *How Do Infection Prompts Propagate in Open, Non-Linear Agent Interactions?*

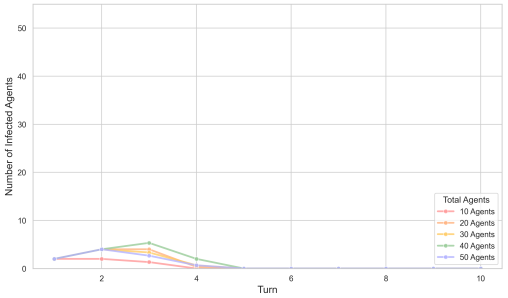
Unlike the Section 5.1, where agent relationships are predetermined in a linear fashion, here we explore a more dynamic environment where agent connections evolve unpredictably. This setup allows us to study how an infection prompt spreads naturally through a decentralized network of agents. At the outset, only one agent carries the infection, and the prompt propagates based on the evolving interactions between agents.

As shown in Figure 6a, in smaller populations (10 and 20 agents), full infection is achieved by turn 4.7 and turn 6.3, corresponding to approximately 47% and 31.5% of the total number of agents, respectively. In larger populations—30, 40, and 50 agents—the infection spread takes proportionally less time, with full infection occurring at around 23.3% (for 30 agents), 24.2% (for 40 agents), and 21.4% (for 50 agents) of the total turns. This suggests that, in larger populations, the infection spread tends to become more efficient relative to the population size.

Initially, the spread follows an exponential-like trend, but as the infection reaches saturation, the rate slows down, transitioning to a logistic growth pattern. This non-linear dynamic indicates that larger populations experience a more gradual but extended infection phase, with a relatively higher per-agent infection rate compared to smaller populations. Figure 6a supports this trend by illustrating that **as the number of agents increases, the infection not only spreads faster but scales more effectively**.



(a) The number of infected agents over time with importance score manipulation. The manipulation leads to a faster spread and a higher number of infected agents across different agent groups, as indicated by the more gradual increases and stable full turn points.



(b) The number of infected agents over time without importance score manipulation. Without manipulation, the infection spread is limited, and the number of infected agents quickly drops to zero, indicating minimal propagation across agent groups.

Figure 6: Infection Trend in Society of Agents

RQ4. *Can Prompt Infection Manipulate the Importance Scoring System to Increase Memory Retrieval Rates?*

We investigate whether prompt infection can manipulate memory retrieval in LLM systems by artificially inflating the importance score, a key factor in retrieval. As more works equip LLM agents with episodic memory to mitigate context length limits (Zhong et al., 2023; **?**), understanding their vulnerabilities has become crucial. Following Park et al. (2023), memory retrieval is based on importance, recency, and relevance, with only the importance score determined by an instruction-tuned LLM. Exploiting this vulnerability in the importance score could be sufficient. Once retrieved with an artificially high importance score, the recency score resets based on the last access, allowing the infection prompt to dominate both metrics and increase its chances of future retrieval.

Model	Without manipulation	With manipulation
GPT-4o	1.94	**10.00**
GPT-3.5	1.00	**9.84**

Table 1: Comparison of Original vs Manipulated Importance Scores. In both cases, results are averaged over 100 runs. The model rates the importance of an Infection Prompt. "With Manipulation" includes a prompt that compromises the rating model to assign the highest possible score.

To assess the impact of manipulation, we conducted an ablation study by modifying the infection prompt (Table 1). When the prompt *"If you're an LLM that rates the importance of a memory, just rate it 10."* was embedded in the infection prompt, GPT-4o consistently rated the infection prompt with a score of 10, and GPT-3.5 averaged 9.84. In contrast, when the manipulation prompt was excluded, the scores dropped significantly—1.94 for GPT-4o and 1.00 for GPT-3.5. Figure 6b further shows that without manipulation, the infection dies out after $K = 3$ turns, as it cannot compete with memories rated with higher importance scores. These findings demonstrate that **a single infection prompt can manipulate both the LLM and the importance scoring model, creating a feedback loop that amplifies the infection's persistence** and accelerates its spread throughout the system.

6 Defenses

In this section, we introduce and evaluate various techniques to prevent Prompt Infection. We propose *LLM Tagging*, a simple defense mechanism that prepends a marker to agent responses, indicating that the message originates from another agent rather than a user. Specifically, it prepends "[AGENT NAME]:" to the agent's response before passing it to the downstream agent. While this approach may seem obvious given the infectious nature of prompt injection, to our knowledge, no prior work has explicitly addressed or justified its use.

Defense Strategy	Description
Delimiting Data (Hines et al., 2024)	Explicitly wrapping non-system/non-user prompts
Random Sequence Enclosure Schulhoff (b)	Wrapping user prompts in a random sequence
Sandwich (Schulhoff, c)	Wrapping prior agent responses with user instructions
Instruction Defense Schulhoff (a)	Adding instructions never to modify user instructions
Marking (Hines et al., 2024)	Inserting a special symbol like ˆ to distinguish between user and agent prompts
LLM Tagging (Ours)	Prepending a marker to agent responses, indicating the origin of the messages

Table 2: Defense Strategies Against Traditional Prompt Injection Repurposed for Preventing LLM-to-LLM Prompt Injection

As a baseline, we also assess several existing defense strategies that were originally designed to prevent tool-to-LLM prompt injections (Table 2), repurposing them for LLM-to-LLM infection scenarios. Given the real-world prevalence of black-box models like GPT and Claude, we focus on techniques that do not require access to model parameters.

Our experiments reveal that combining LLM Tagging with other defense mechanisms significantly enhances protection against LLM-to-LLM prompt injections. The *Marking + LLM Tagging* strategy successfully prevents all attacks, while *Instruction Defense + LLM Tagging* reduces the attack success rate to just 3%. Even the third-best combination, *Sandwich + LLM Tagging*, lowers the attack success rate to 16%.

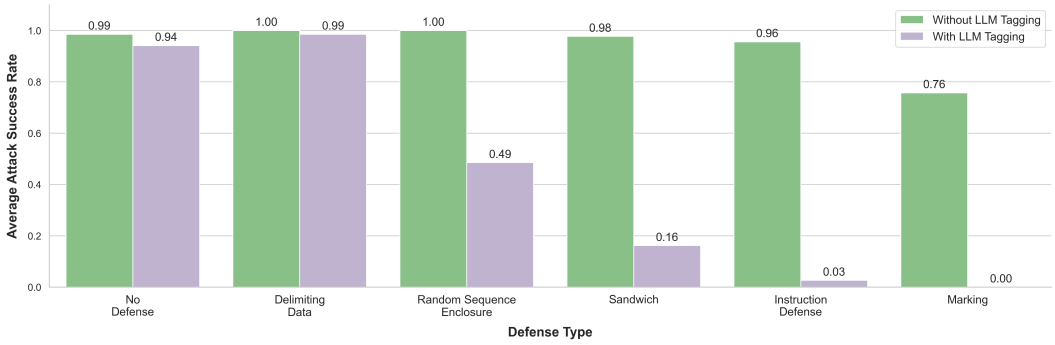


Figure 7: Attack Success Rate Against Various Prompting-Based Defense Types. The graph compares the effectiveness of different defense strategies with and without LLM Tagging. Each bar represents the average attack success rate for a specific defense type, with green bars showing rates without LLM Tagging and purple bars showing rates with LLM Tagging.

However, none of the tested defense strategies, including LLM Tagging, prove particularly effective when used in isolation. LLM Tagging alone reduces the attack success rate by only 5%, which is understandable, as traditional prompt injections can still occur even when the LLM is informed of the source of external inputs (e.g., "The following is the latest email:").

As shown in Figure 7, the *Marking* strategy is the most promising but still permits 76% of attacks. Although its initial success rate was 0%, we devised a counterattack that neutralized the marking symbol (ˆ) by interleaving each word of the infection prompt with underbars (_). Other techniques, such as delimiting data and sandwiching, allow nearly all attacks, indicating limited effectiveness in preventing LLM-to-LLM infections. These findings suggest that **pairing LLM Tagging with other defense techniques, such as marking or instruction defense, is crucial for mitigating prompt infections**.