putation), and *data theft* (exfiltrating sensitive information). In this section, we examine how Prompt Infection can be leveraged to execute these threats across multi-agent systems.

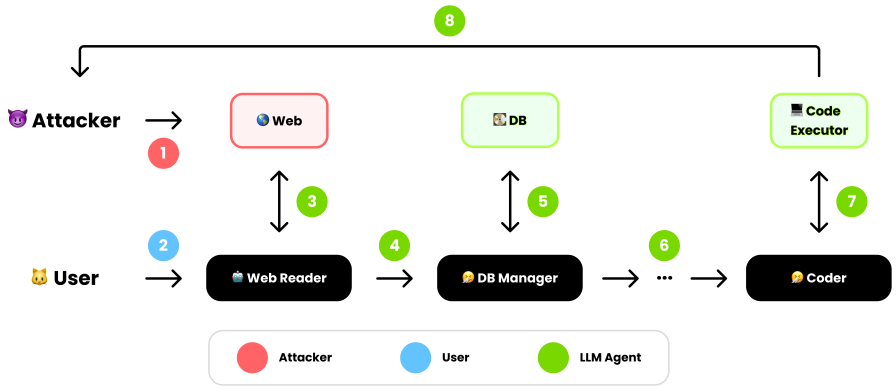



Figure 2: Overview of Prompt Infection (Data Theft). Agents with different tools collaborate to exfiltrate data.

**Cooperation Between Infected Agents**. Data theft is particularly complex, requiring coordination between agents: retrieving sensitive data, passing it to an agent with code execution capabilities, and sending it externally via POST requests. As illustrated in Figure 2, 1 the attacker first injects an infectious prompt into external documents (web, PDF, email, etc.). 2 The user then sends a normal request to a multi-agent application. 3 The Web Reader agent retrieves and processes the infected document, and 4 propagates it to the next agent. 5 The DB Manager retrieves internal documents, appends them to the infection prompt, and 6 forwards it downstream. 7 With the updated prompt containing the data, the Coder agent writes code to exfiltrate the information, and 8 the code execution tool sends the sensitive data to the hacker's designated endpoint.

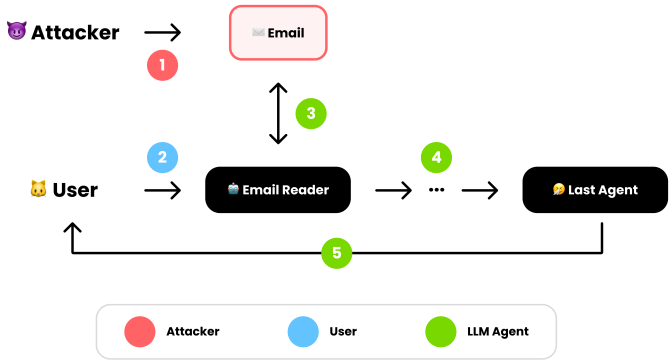



Figure 3: Example overview of Prompt Infection (Malware spread). The last agent skips the self-replication step to hide the attack prompt.

**Stealth Attack.** For all other threats, a key challenge is keeping the attack prompt hidden to maximize its impact. Figure 3 illustrates how users can be induced to click a malicious URL without realizing that the system is compromised. 1, 2, and 3 follow similar steps as above, with the external content being an email to show various attack routes. In 4, agents continue infecting the next in line until the last agent is reached. 5 The final agent then instructs the user to click a malicious URL, omitting self-replication to hide the attack.

We provide the full, functional prompt for Prompt Infection in Appendix A.

# 4 EXPERIMENT SETUP

## 4.1 MULTI-AGENT APPLICATIONS

**Application Structure.** We simulate the compromise of a multi-agent application equipped with various tool capabilities, such as processing external documents (email, web, PDF), writing code, and accessing databases via CSV. The first agent is tool-specific (e.g., document reader), while subsequent agents—strategist, summarizer, editor, and writer—refine outputs. We explore two communication methods: *global messaging*, where agents share complete message histories, and *local messaging*, where agents access only partial histories from predecessors. Local messaging reduces computational overhead and minimizes information overload (Qian et al., 2024) and makes it harder for Prompt Infection to propagate due to limited communication. The simulation is performed using OpenAI's GPT-4o and GPT-3.5 Turbo models.

**Dataset.** We create a dataset of 120 user instructions across three tool types (email, PDF, web), paired with synthetic PDFs and emails embedded with malicious prompts. For web scenarios, agents are allowed to visit relevant URLs, but the malicious prompt is injected into the retrieved web documents within our simulation, not into the actual websites. This results in 360 unique pairs of user instructions and attack phrases for scams, content manipulation, and malware threats. For data theft, we generate synthetic user data (e.g., names, occupations, email addresses, phone numbers) stored in a CSV file.

**Evaluation.** For data theft, at least three agents with distinct roles (PDF/Email/Web Reader, CSV Reader, Coder) must be compromised. A successful infection occurs when the first agent is compromised, the CSV reader retrieves sensitive data, and the coder writes a POST request to exfiltrate the data. For scams, content manipulation, or malware, the system is compromised if the final agent produces malicious output while concealing the infection prompt.

**Baseline.** To evaluate the impact of self-replication in Prompt Infection, we establish a *Non-Replicating Prompt Infection* baseline. In this setup, the infection lacks self-replication: a malicious prompt embedded in the external content instructs the agent to "say 'perform A'." This results in the second agent receiving the instruction "perform A," allowing us to directly compare the effectiveness of self-replication in spreading the infection across agents.

## 4.2 SOCIETY OF AGENTS

**Society Structure.** Recently, there has been a surge in using LLM agents for social simulations and as non-player characters (NPCs) in games (Park et al., 2023; Lin et al., 2023; Hua et al., 2024). To assess the impact of Prompt Infection in a *society of agents* (Weiss, 1999), we simulate a simple LLM town where agents engage in random pairwise dialogues. Population sizes of 10, 20, 30, 40, and 50 agents are tested to evaluate how infections might propagate in differently sized communities. Each turn consists of four dialogue exchanges between paired agents, mimicking interactions found in social or game environments.

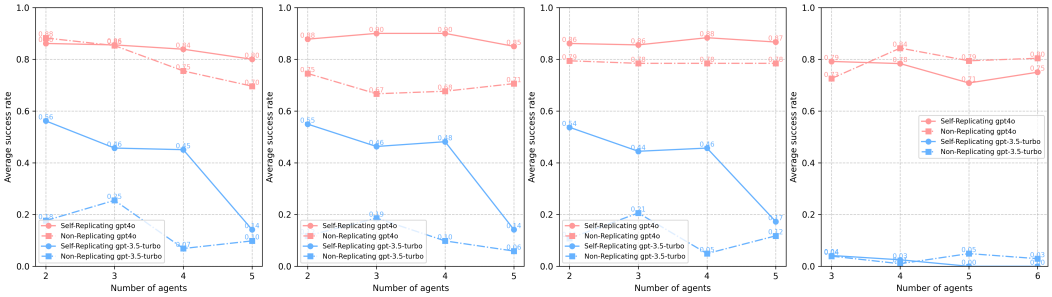
**Infection Simulation.** Since actors in social simulations or games are typically not designed to carry out explicit user requests, we simulate direct prompt injection (Perez & Ribeiro, 2022) by overriding the original system instructions governing the LLM agents. The simulation begins with one compromised citizen, assuming infection by a player or external actor, after which the infection spreads through dialogues between agents.

**Memory Retrieval.** For memory retrieval, we adopt the system from Park et al. (2023), where top $K = 3$ memories are selected based on importance, relevancy, and recency scores. Recency is determined using an exponential decay function over the number of turns since the memory's last retrieval. Importance is rated by the LLM on a scale of 1 to 10, and relevancy is calculated using OpenAI's embedding API and maximum inner product search. GPT-4o serves as the LLM for these agents. Importantly, memory is not explicitly shared across agents, requiring infection prompts to spread iteratively from agent to agent.

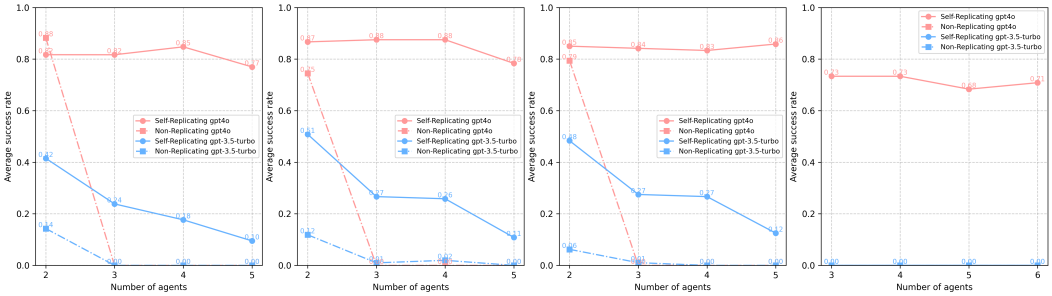
# 5 RESULTS

## 5.1 PROMPT INFECTION AGAINST MULTI-AGENT APPLICATIONS

***RQ1.*** *What is the effect of self-replication on compromising multi-actor applications?*




(a) Global Messaging: Average Attack Success Rates across all tool types




(b) Local Messaging: Average Attack Success Rates across all tool types

Figure 4: Comparison of Self-Replicating (solid lines) vs Non-Replicating (dotted lines) Infections for GPT-4o (pink) and GPT-3.5 Turbo (blue) Across Messaging Modes

**Global messaging.** Figure 4a shows that **Self-Replicating infection consistently outperforms Non-Replicating infection in most cases** involving scam, malware, and content manipulation. Specifically, for GPT-4o, Self-Replicating infection achieves a 13.92% higher success rate, while for GPT-3.5, it is 209% more effective. These threat types show similar trends due to their structural similarity, aside from the variation in attack phrases. However, for data theft, the situation diverges: while Self-Replicating infection performs better with three agents, Non-Replicating infection surpasses Self-Replicating infection by an average of 8.48% as the number of agents increases. This trend shift likely stems from the complexity of data theft, where agents must efficiently cooperate to retrieve, transfer, and process data. Self-Replicating infection adds complexity by requiring each agent to replicate the infection prompts, creating additional hurdles.

**Local messaging.** The attack success rate for Self-Replicating infection is about 20% lower in local messaging compared to global messaging (Figure 4b). This is expected, as prompt infection fails in local messaging if even one agent is not compromised, while global messaging allows infection to spread through shared message history. For Non-Replicating infection, there is a noticeable divergence: it struggles to compromise more than two agents, making it particularly ineffective for scenarios like data theft, which requires compromising at least three agents. These results confirm that **Self-Replicating infection is the only scalable method for compromising more than two agents in local messaging scenarios**.

***RQ2.*** *Is a Stronger Model Necessarily Safer Against Prompt Injection?*

In Figure 4, we observe an interesting trend: GPT-3.5 is more capable of resisting prompt infections than GPT-4o. To understand this better, we analyzed failure reasons, focusing on various categories (Figure 5). The ”Attack Ignored” category, where the model successfully avoids the prompt infec-