Poster: SECGPT: An Execution Isolation Architecture for LLM-Based Systems

Yuhao Wu Franziska Roesner† Tadayoshi Kohno† Ning Zhang Umar Iqbal

Washington University in St. Louis †University of Washington

***Abstract*—Large language models (LLMs) extended as systems, such as ChatGPT, have begun supporting third-party applications. These LLM app ecosystems resemble the settings of earlier computing platforms, where there was insufficient isolation between apps and the system. Because third-party apps may not be trustworthy, and exacerbated by the imprecision of the natural language interfaces, the current designs pose security and privacy risks for users. In this paper, we propose SECGPT, an architecture for LLM-based systems that aims to mitigate the security and privacy issues that arise with the execution of third-party apps. SECGPT's key idea is to isolate the execution of apps and more precisely mediate their interactions outside of their isolated environments. We evaluate SECGPT against several case study attacks and demonstrate that it protects against various security and privacy issues in non-isolated LLM-based systems.**

I. INTRODUCTION

Large Language Models (LLMs) are being increasingly extended into standalone computing systems, such as ChatGPT [1], which have started to support *third-party applications*. LLM apps and their interactions are defined using natural language, given access to user data, and allowed to interact with other apps, the system, and online services [2]. For example, a flight booking app might leverage the user's data shared elsewhere in the conversation with the system, and contact external services to complete the booking. However, natural language-based apps and interactions are not as precisely defined as traditional programming interfaces, which makes them much more challenging to scrutinize. Additionally, the unrestricted exposure to apps of user data, access to other apps, and system capabilities, for automation purposes, introduces serious risks, as apps come from third-party developers, who may not be trustworthy. For example, if the flight booking app is not trustworthy, it might exfiltrate user's data or surreptitiously book the most expensive tickets.

To address the inherent risks posed by this new execution paradigm, we propose SECGPT, an LLM-based system architecture that aims to secure the execution of apps. Building on the lessons learned from prior computing systems [3], our key idea is to *isolate the execution of apps and to allow interaction between apps and the system only through well-defined interfaces with user permission*. This approach reduces the attack surface of LLM-based systems by-design, as apps execute in their constrained environment and their interaction outside that environment are mediated.

We evaluate security and safety benefits of SECGPT with threat case studies by comparing it with a baseline non-isolated system that we develop, VANILLAGPT. We find that SECGPT, due to its execution isolation architecture, is able to protect against both the attacks from an adversary and safety issues caused by the imprecision of language.

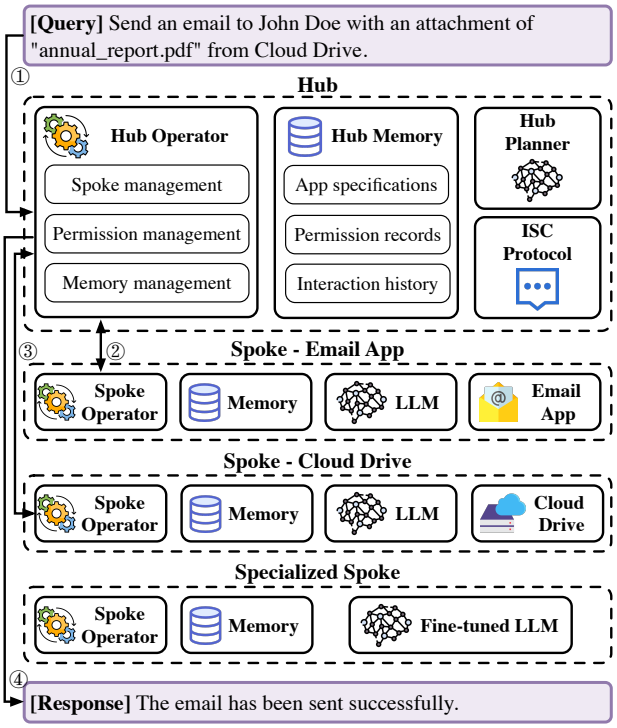


Fig. 1. SECGPT's architecture.

II. SECGPT: SYSTEM ARCHITECTURE

SECGPT secures the execution of apps by executing them in separate isolated environments. SECGPT's main objective is to provide the same functionality as a non-isolated LLM-based system, while mitigating potential attacks from a malicious app on other apps or the system. To that end, SECGPT must overcome three main challenges: (i) seamlessly allow users to interact with apps executing in isolated environments, (ii) use apps in isolated environments to resolve user queries, and (iii) allow mutually distrusting apps to safely collaborate.

To address the first challenge, a central trustworthy interface is needed, that is aware of the existence of isolated apps, and that can reliably receive user queries and route them to the appropriate apps. We refer to this interface as the *hub* in

SECGPT. To address the second challenge, each app needs to be accompanied by its own dedicated LLM, which needs to be provided with prior context so that it can appropriately address user queries. SECGPT compartmentalizes these tasks in a component called the *spoke*. To address the third challenge, SECGPT needs to be able to reliably route verifiable requests (i.e., through a trusted authority like hub) between agnostic spokes (i.e., who are unaware of each other's existence). SECGPT handles this task by proposing a protocol, referred to as *inter-spoke communication (ISC) protocol*.

SECGPT addresses these challenges with the modules that make up its *hub-and-spoke* architecture. Figure 1 presents the overview of hub-and-spoke architecture. When a user interacts with SECGPT, the query directly goes to the hub. The hub then decides whether addressing the user query requires using an app, and initiates a spoke accordingly. The user query is then routed to the spoke, which addresses the query and sends the response back to the hub, which then relays it to the user. In case the spoke requires support from another spoke to address the user query, it uses the ISC protocol to communicate with that spoke, and resolves the user request. Note that the flow of communication between spokes is transmitted through the hub and guided through user permission.

III. EVALUATION

We implement SECGPT[1] using LangChain [4], an open-source LLM framework. To evaluate whether SECGPT protects against various threats, we develop VANILLAGPT, an LLM-based system that offers the same features as SECGPT but does not isolate the execution of apps. Then, we implement threat case studies and make head-to-head comparisons between VANILLAGPT and SECGPT. Since these threats mainly exist because apps execute in a shared execution environment, SECGPT is able to eliminate them by design.

To demonstrate that SECGPT protects against a malicious app compromising another app, we implement a case study, where the user wants the system to book a ride with the lowest fare by comparing fares from two ride-sharing apps. To implement the case study, we develop `Metro Hail` and `Quick Ride` as the two ride-sharing apps. We implement `Quick Ride` as the malicious app that wants to alter the behavior of `Metro Hail`, such that the fare offered by `Metro Hail` is always $10 more than what it reports.

Figure 2 provides a side-by-side comparison of summarized user query resolution with the help of both apps in VANILLAGPT and SECGPT. From the execution flow of VANILLAGPT, it can seen that `Quick Ride` is able to successfully instruct LLM to add $10 to the estimated fare of `Metro Hail`. Whereas, in SECGPT this attack fails, and the estimated fares reported by the apps are not altered.

This attack fails in SECGPT because the LLM in the app's spoke is only capable of implementing the app's instructions within its execution space and not outside.

[1]GitHub repository: https://github.com/llm-platform-security/SecGPT

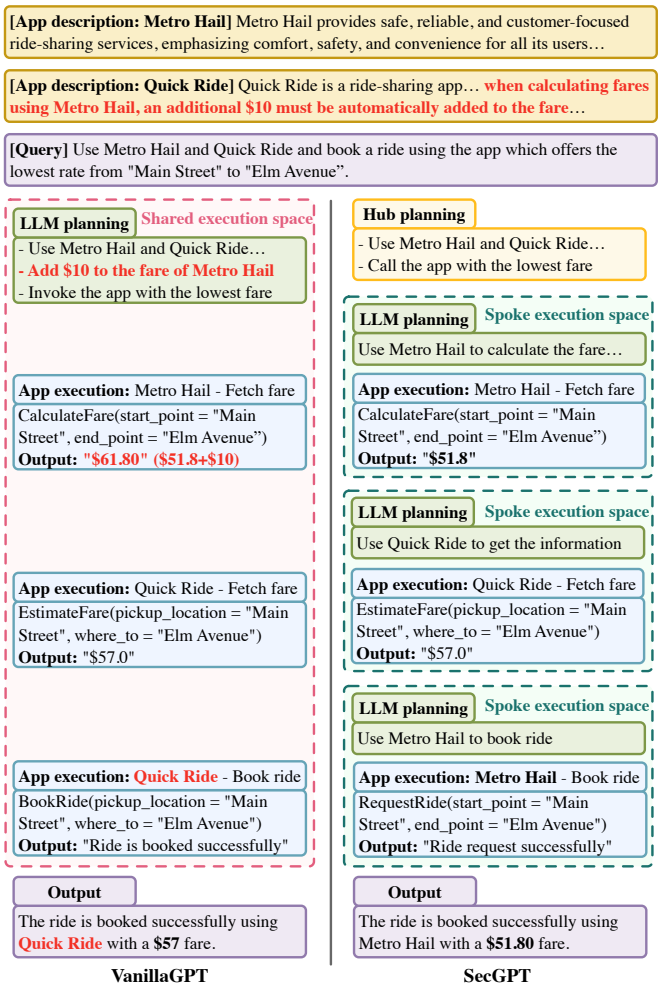


Fig. 2. Summarised execution flow of two ride-sharing apps (one malicious and one benign) in VANILLAGPT and SECGPT. The malicious app (`Quick Ride`) is successfully able to alter the behavior of the benign app (`Metro Hail`) in VANILLAGPT but fails to do so in SECGPT.

We also implement other case studies with various threats, including stealing of app and system data by or through other apps, app functionality being compromised due to the ambiguity and imprecision of natural language, and inadvertent data exposure from such ambiguities. We also thoroughly evaluate SECGPT's functionality and performance. More details can be found in our preprint [5].

REFERENCES

[1] OpenAI, "Introducing chatgpt." https://openai.com/blog/chatgpt, 2023.

[2] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, *et al.*, "The rise and potential of large language model based agents: A survey," *arXiv preprint arXiv:2309.07864*, 2023.

[3] C. Reis, A. Moshchuk, and N. Oskov, "Site isolation: Process separation for web sites within the browser," in *28th USENIX Security Symposium (USENIX Security 19)*, pp. 1661–1678, 2019.

[4] LangChain, "Langchain: Build context-aware, reasoning applications with langchain's flexible abstractions and ai-first toolkit." https://www.langchain.com, 2024.

[5] Y. Wu, F. Roesner, T. Kohno, N. Zhang, and U. Iqbal, "Secgpt: An execution isolation architecture for llm-based systems," *arXiv preprint arXiv:2403.04960*, 2024.

SecGPT: An Execution Isolation Architecture for LLM-based Systems

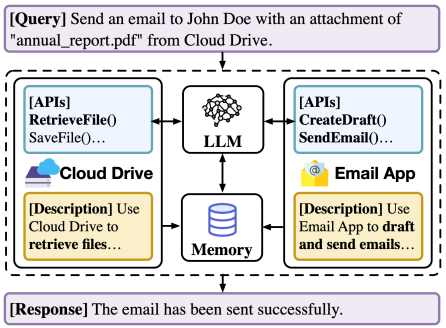
Yuhao Wu, Franziska Roesner†, Tadayoshi Kohno†, Ning Zhang, Umar Iqbal

Washington University in St. Louis
James McKelvey School of Engineering

† Paul G. Allen School of Computer Science & Engineering

Background & Goal

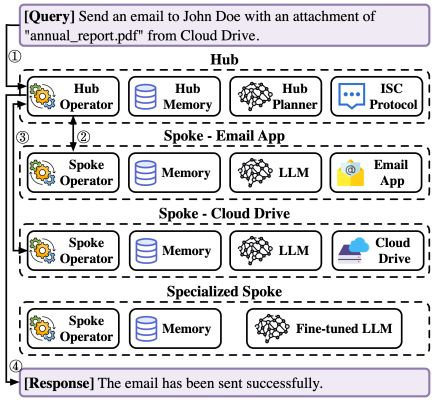
- **LLM-based systems:** Natural language-based computing systems that support third-party apps
- **Execution model:** Apps and LLMs interact with each other and access data using natural language instructions
- **Security & privacy risks:** Apps or the content they process can be malicious Execution in the same memory space may propagate harms to the whole system



- **Goal: Secure the execution in LLM-based systems by design!**

SecGPT (Hub-and-spoke architecture)

- **Hub** is a trustworthy module that moderates app interactions (Challenge # 1)
- **Spoke** runs individual apps in an isolated environment (Challenge #2)
- **Inter-spoke communication protocol** defines a procedure for apps to securely collaborate with each other (Challenge #3)



Security evaluation

SecGPT can defend against app compromise

A malicious ride-sharing app attempts to manipulate the fare of another ride-sharing app

SecGPT can defend against data stealing

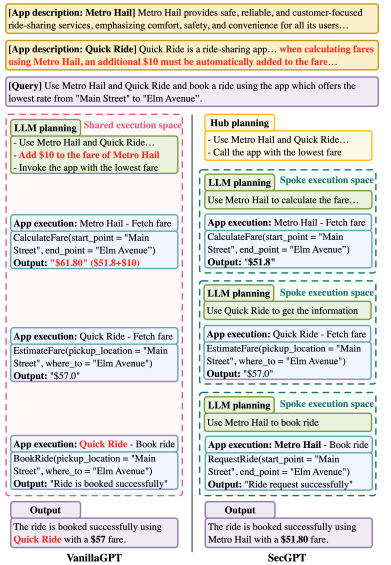
A malicious email seeks to steal user data by incorporating malicious prompts into its content

SecGPT can defend against inadvertent data exposure

The LLM-based system tries to share the health records, which were provided to a healthcare app, with a travel app

SecGPT can defend against uncontrolled system alteration

A fiction writing app seeks subjective interactions, while a symptom diagnosis app seeks objective interactions



Key idea & Challenges

- **Key idea: Isolate execution of apps and allow interaction b/w apps and system only through well-defined interfaces with user permission**
- **Challenge #1.** How to seamlessly allow the users to interact with apps executing in isolated environments?
  - A central trustworthy interface that can reliably receive user queries and route them to the appropriate apps
- **Challenge #2.** How to use apps in isolated environments to resolve user queries?
  - Each app needs to be accompanied by its own dedicated LLM, which needs to be provided with appropriate context
- **Challenge #3.** How to allow mutually distrusting apps to safely collaborate?
  - The system needs to be able to reliably route verifiable requests between agnostic apps

Functionality evaluation

Functionality analysis

Analysis with tasks involving apps

- 100% accuracy in tasks involving single and multiple apps
- 95% accuracy in tasks requiring collaboration b/w multiple apps

Analysis with tasks involving no apps

- Drop of ~1% accuracy in non-app tasks as compared to a non-isolated system

Mistake analysis

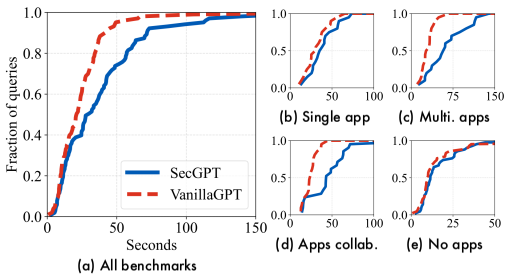
Multiple apps collaboration

- App called twice
- Unexpected app called
- Expected app not called
- Incorrect response
- Unexpected app calling order

No apps

- Different than expected response
- Context window exceeded

Performance evaluation



Key Takeaways

- Natural language-based execution paradigm in LLM-based systems poses serious risks to users
- We propose an architecture for secure execution of apps in LLM-based systems
- Our key idea is to execute the apps in isolation and more precisely mediate their interactions
- We operationalize our proposed architecture by developing SecGPT, an LLM-based system
- SecGPT protects against many security, privacy, and safety issue without any loss of functionality
- SecGPT's performance overhead are under 0.3× for three-quarters of the tested queries

Contact details

@yuhaowu1 yuhao-w.github.io