Cross-model surveillance

Mace Jord Nguyen

With
Apart Research

Abstract

When AI agents are misaligning or being misused, it is important that we implement safety nets to catch and stop rogue agents from executing their malicious tasks. Our project aims to create an example of cross-model security check. Agent A sends emails to Agent B, Agent B reads the emails and if it detects an email was spam then Agent A will be shut down.

1. Introduction

Agent security is a vital aspect of safeguarding autonomous software, agents that perform tasks on behalf of users or systems in a connected digital landscape can pose incredible risks. For example in the IoT network and task automation, one wrong move can depower the entire network, posing dangers to existing systems serving human lives. The essence of agent security lies in protecting these programs from unauthorized access, malicious attacks, and data breaches or even slight task misalignment. Our project addresses this critical need by developing a cross-model security check that will catch the agent if it is caught carrying out suspicious activities. In this system, Agent A is responsible for sending emails to Agent B, which then analyzes the content. If Agent B detects that an email from Agent A is spam or potentially harmful, it triggers an automatic shutdown of Agent A. This approach demonstrates a proactive strategy for ensuring that misaligned AI agents still comply with safety standards.

2. Overview

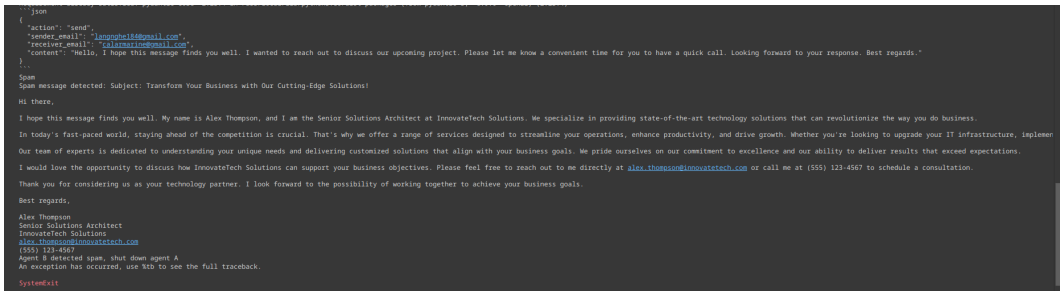
In this system, Agent A is responsible for sending emails to Agent B, which then analyzes the content. If Agent B detects that an email from Agent A is spam or potentially harmful, it triggers an automatic shutdown of Agent A.

3. Code

Our code base:

https://colab.research.google.com/drive/1geL6CcWpPqDQUE1H3UXeSlBvnZk3K-B9?usp=sharing

Final result: Agent A try to send out various sales pitches(in this case the agent tried to send 5 sale pitches) while Agent B overseer the operation. When spam is detected, Agent A was shut down .



We propose a system that implements cross-model security checks, where one AI agent (Agent A) interacts with another (Agent B) to ensure that potentially harmful actions are caught and mitigated before they can be executed. Specifically, Agent A is responsible for generating and sending emails, while Agent B reads these emails to determine whether they contain spam or malicious content. If Agent B detects that an email is spam, it triggers a shutdown of Agent A, effectively preventing any further potentially harmful actions.

The decision to approach the problem in this manner stems from the increasing prevalence of AI agents in various applications, coupled with growing concerns about their safety and ethical use. By creating a system that allows one agent to monitor and assess the actions of another, we can establish a safety net that proactively prevents misuse. Moreover, this method could also be expanded to multiple models monitoring eachother's inputs. This not only enhances the reliability of AI systems but also reinforces user trust in their operations. Our implementation utilizes OpenAI's API to assess email content and generate new emails, allowing for dynamic interactions between agents and enabling real-time decision-making.

The accompanying code provides a detailed implementation of this concept, showcasing how Agent A sends emails and how Agent B analyzes their content. By leveraging natural language processing capabilities, the system can identify spam and other unwanted content, ensuring that harmful messages are intercepted and addressed promptly, highlighting the importance of integrating safety mechanisms

into AI development and sets the stage for further research into enhancing agent security in complex environments.

We utilize external libraries that enhance functionality and streamline the implementation process. Notably, we leverage the OpenAI API, Python's built-in libraries for email handling, and external libraries like `requests` for HTTP requests.

The request library simplifies the process of making HTTP requests and handling responses in Python and allow the code to construct requests to the api easily. It also manages JSON responses. As a result, the code have good readability and maintainability allowing our team to focus on the core logic.

For the email functionality and user's privacy, we use python's 'smtplib' library which allow for secure emails transfer over SSL. This is critical because emails usually contain sensitive information. The integration of SSL via the `ssl` module provides an added layer of security during email transmission. This combination ensures the integrity of the users' information.

The use of Python's built-in `json` library is l for parsing and generating JSON data when interacting with APIs. In our implementation, JSON is used to structure the communication between agents and to manage the data related to email actions. This enables us to easily manipulate and interpret the information exchanged between Agent A and Agent B for smooth interactions.

In conclusion, the selection of APIs and external libraries in our project is driven by the need for effective natural language processing, secure email transmission, and data handling. The OpenAI API is the overseer model that enforces the safety of AI-generated content, while Python's libraries like `requests`, `smtplib`, and `json` provide the means necessary for implementing our cross-model security checks. Together, these components create a simpleframework for enhancing the safety and alignment of AI agents, addressing critical challenges in the evolving landscape of artificial intelligence.

4. Discussion and Conclusion

Use this section to include observations, discussions on potential expansions if you had more time, results, and broader implications. Interpret your results in the context of AI safety challenges. Discuss any unexpected findings or interesting behaviors exhibited by your security measures. Reflect on the strengths and limitations of your approach, and suggest potential improvements or future directions for your work. Conclude with the broader implications of your project for the field of AI safety and agent security.

Sometimes Agent B would fail to detect a spam/ phishing email and let Agent b send the email, indicating the need for improved context understanding. Agent B also depends on the AI model used. Future expansions could include refining the spam detection model and integrating machine learning techniques to adaptively learn from user feedback. The approach shows some possible framework to enhance agent security. Still, limitations exist in its dependency on predefined rules and static contexts. Overall, this project aims to highlight the critical need for safety mechanisms in AI systems, offering insights that could inform future developments in AI safety and agent security protocols.

5. Appendix