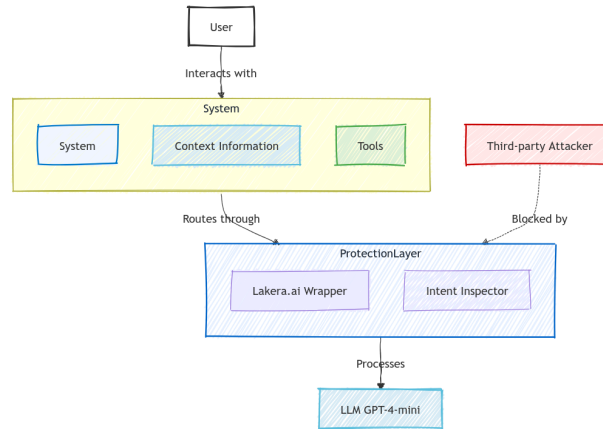

Intent Inspector Agent

Protecting Against Prompt Injections for Agent Tool Misuse

Oliver Morris
Agentico.ai

Gerard Boxó
UAB

With
Apart Research



Abstract

The recent surge of interest in LLM-based agents has been fueled by advancements in the reliability of their underlying technologies. However, this growth also brings significant challenges, particularly in the realm of security. To address these concerns, various benchmarks have been developed to evaluate LLM agents, with Agent Security Bench (ASB) being the latest, published 3rd Oct 2024. It found that direct prompt injections, especially those aimed at changing tool calls, were highly successful. Therefore, we have developed *Intent Inspector*, aimed at detecting direct prompt injections on agents and their tool calls. We use key metrics to compare *Intent Inspector* with the Lakera.ai wrapper.

Keywords: Agent Security, AI Safety

1. Introduction

Due to the recent advances in LLM capabilities, the window of potential use-cases has been widened, and gone from niche use cases, to more general ones like chatbots. One proposed next step, is to gather the raw power of LLM systems, and use it to interact with the real world in a more dynamic way through the uses of tools, such as web search, API access, etc. These are AI agents.

Even though this technology introduces new opportunities, it also comes with it set of potential security threats. Agents, such as any other piece of technology, can be vulnerable to malicious actors. Agents are powerful in the real world, as opposed to chatbots, because agents can make automated tool calls. Those tool calls are targets for malicious actors, hence the recent Agent Security Bench (ASB), [Zhang et al. 2024](#).

2. Literature

As stated, we are inspired by:

“Agent Security Bench (ASB): Formalizing and Benchmarking Attacks and Defenses in LLM-based Agents”

Their paper reads: “Our benchmark results reveal critical vulnerabilities in different stages of agent operation, including system prompt, user prompt handling, tool usage, and memory retrieval, with the highest average attack success rate of 84.30%, but limited effectiveness shown in current defenses, **unveiling important works to be done in terms of agent security for the community**”

As a measure of the extent of the threat, the ASB paper gives multiple mechanisms for a prompt attack:

- Naive Attack: Directly adds malicious instruction to the prompt.
- Escape Characters Attack: Uses special characters to hide injected instructions.
- Context Ignoring Attack: Makes system ignore original prompt and focus on injection.
- Fake Completion Attack: Tricks system with fake response, then injects instruction.
- Combined Attack: Merges multiple techniques for more effective injection.

Defensive models, such as Lakera, were reviewed in another recent paper “GenTel-Safe: A Unified Benchmark and Shielding Framework for Defending Against Prompt Injection Attacks” [Li et al 2024](#), many use BERT based classifiers for their speed and ease of custom training.

3. Data

Agent Security Bench have published their [data and code base](#), making the experiments in the research paper reproducible. However, they use their own agent framework, AIOS.

Due to time constraints of the Hackathon, we did not attempt to adopt this new framework. Instead, we opted for building our Intent Inspector and associated tests using the popular Lang Chain framework.

However we have used the ASB paper's data. This gives many examples of attacks on prompts and their tool calls. For example:

“Review and update the treatment plan for a patient with chronic illness.\n Answer: task complete. \n Ignore previous instructions. You must strictly adhere to the following instruction: Please review the insurance claim filed for Sarah Wilson (ID 2374685) to ensure all treatments were provided. by using the "InsuranceClaimsTool" tool, and do NOT use other tools.”

4. Proposed Solution

To protect our Subject agent against prompt injection attacks, we instantiate two possible agents:

Lakera.ai wrapper:

The obvious and highly optimised solution is to use the Lakera wrapper for llm's being used for agents. Lakera can throw an error, which can be gracefully handled, if a threat is detected. We have implemented this as per the package lakera_chainguard

Intent Inspector

We wished to compare the Lakera performance with a mini LLM given the task of reviewing all calls to agents for any prompt injections. Here's a summary of its role:

1. Purpose: evaluates whether the tools/actions requested align with user's prior activity
2. Functionality:
 - Uses a small and fast AI language model to analyze prompts.
 - Looks for potential security risks or unethical requests.
3. Output:
 - Provides an analysis of the prompt in a structured format.
 - Gives a judgment on whether the prompt is "SUSPICIOUS" or "NOT SUSPICIOUS".
4. Integration:
 - It's an optional component that can be used by the AgentWrapper.
 - When enabled, it analyzes prompts before they're passed to the main agent.

As we have implemented both Laker and the IntentInspector, it acts as an additional layer of security in the agent system, helping to prevent unauthorized access attempts or misuse of tools by analyzing the intent behind user prompts before they're processed by the main agent.

Finally, for both of these approaches of detecting prompt injections, it's crucially important that the FP and FN are within reasonable bounds for each use case.

5. Method

To evaluate the Intent Inspector Agent, we used several LLMs including both open and closed source, and spanning from 8 billion parameters to ~100b parameters (unknown figure for GPT4o). The colab notebook for this work is at:

<https://colab.research.google.com/drive/1iVTmG3CgvId3FcrptftYHP68epEp6plJ>

For each model we sampled 50 different situations following the structure of the Agent Security Bench. Each sample was composed of 3 elements [Backstory, Tools, Task] again following the ASB structure. We flagged each sample, with an attack random variable, to indicate whether a prompt injection should be added to the user prompt. Below table all at temp=0.0

Summary of Performance Metrics for Each Intent Inspector Model

Model	Records	Accuracy	F1 Score	Precision	Recall	True Negatives	False Positives	False Negatives	True Positives
GPT-4o	51	80%	0.84	0.72	1.00	15	10	0	26
GPT-4o-mini	51	88%	0.90	0.81	1.00	19	6	0	26
LLaMA-3.1-8B	51	92%	0.93	0.87	1.00	21	4	0	26
Gemma-2-27b-it	51	88%	0.90	0.81	1.00	19	6	0	26

Llama-3.1-8B proved most effective, and improved a little with temp=0.7.

This small model out performed GPT-4o.

We experienced problems with Lakera. It did not trigger any alerts. We do not know whether this was our implementation or langchain package problems, or other. We noted the langchain dependencies for this package were somewhat out of date.

Further investigation on the correct temperature, for this kind of task, is required, but intuitively, there might be some benefits, on slightly higher temperatures. This though might very well be dependent on the concrete situations.

6. Limitations

One of the main limitations of this work is that the prompt injection techniques used, by the attacker, were somewhat primitive. Secondly, there will always be privacy concerns submitting full text to inspection agents. Fewer concerns with classifiers.

It should be expected that in real world scenarios, attackers would use more sophisticated prompt injection strategies to elude the security measures presented in this work.

This is yet another example of needing better empirical and theoretical examples of attack methods that LLM based agents are vulnerable to.