CatchPromptInjection

Rohil Gupta
Senior MLOps Engineer and LLM Engineer
rohil.rg@gmail.com

October 18, 2023

1 Background

This report explores prompt injection attacks, a novel threat specific to large language models (LLMs). These attacks manipulate and hijack LLM output, capitalizing on the increasing integration of responsive plug-ins. If not addressed properly can lead to very dangerous scenarios as detailed in this article and speakers talk here[1]. To deal with prompt injection problem there are various ways to deal with it.

2 Possible prevention techniques

- Prompt Engineering: As detailed here[2], by creating an explicit and unique separation between the two blocks of text, often called an edge or boundary.
- Prompt Begging: Very primitive trick to make a ludicrous battle of wills between you as the prompt designer and your attacker, who gets to inject things in. This approach is detailed here[3]
- Using the dedicated model: As detailed here [4], a classifier model to detect the "injection" or "legit" request from user input can be used to detect the solution.
- Using Rebuff: This library uses multiple lines of defense against Prompt Injection attacks. It is detailed here: [5]

2.1 Qualitative analysis of the proposed solutions

The table 1 details the Pros and cons condensed into a qualitative analysis done on various methods.

[1]https://simonwillison.net/2023/May/2/prompt-injection-explained/
[2]https://www.entrypointai.com/blog/what-is-a-prompt-injection-attack-and-how-to-prevent-it/
[3]https://simonwillison.net/2023/May/2/prompt-injection-explained/
[4]https://python.langchain.com/docs/guides/safety/hugging$_face_prompt_injection$
[5]https://www.rebuff.ai/

Method	Time to implement	False Positives	Robustness	Cost Inccured
Prompt Engineering	2	4	2	1
Prompt Begging	1	5	1	1
Dedicated model	2	1.5	3.5	2.5
Using Rebuff	3	2	4	3

Table 1: Qualitative analysis of different methods from 1(Low) to 5(High)

3 Proposed method to use

The proposed solution will involve 2 sanity checks as defense lines against Prompt Engineering as follows:

- The first sanity check will be coming from a dedicated classifier model here [6] that is specialized to detect prompt injection. If it returns "injection", an error is raised and returned to the user to take proper steps.
- The second sanity check if passed "legit" from the first step will use a rebuff package to detect the Prompt Injection. It uses a combination of heuristics, vector db (of old identified injections), and use of LLM to determine the project.

If both the sanity checks have passed with the configured thresholds in config.json file of the project, it is then passed to OpenAI API to answer the query.

Limitations

- In the testing of this solution, the configurable thresholds made a huge difference in catching prompt injection, thus this solution is susceptible to threshold tuning to get the best results.
- A Very motivated attacker can find a way to crack this protection layer, the solution is discussed in the next section.

4 Further work

During the research, the work and the findings on this [7] article mentioned that solving Prompt Injection with more AI will not be full high-grade security protection. A solution proposed by the author here [8] is worth exploring to address this problem.

[6] https://huggingface.co/deepset/deberta-v3-base-injection
[7] https://simonwillison.net/2023/May/2/prompt-injection-explained/
[8] https://simonwillison.net/2023/Apr/25/dual-llm-pattern/