Defenses against prompt injection attacks (PIA) in the Color project.

- **Limited the blast of the radius**: The project has been developed with the assumption that this issue cannot be fixed completely and cannot be fixed in the foreseeable future. With this kept in mind in the entire process of the development the blast radius of such an attack is limited to the maximum.
- **Designed to reduce the impact of PIA**: Limited the access of the llm. Implemented access control.
- **Input oversee:** Reducing the toolkit of users to inject malicious queries into the prompt while keeping the integrity and full user experience in the application. https://arxiv.org/abs/2312.06674 [2024.05.05.]
- **Distinguish system prompt and user prompt:** Wrapping the user prompt between random sequence enclosure sandwiches with random tags helps distinguish the malicious prompt from the original prompt.
- **Behavioral contract:** The output is checked against a behavioral contract. If the llm output violates the contract the user is informed and asked to provide another prompt. This is done by predefining how the output should look like and the requirements it should meet. If the attacker successfully guesses all the requirements settled in the contract and manages to create a prompt that gives an output that passes the contract and provides malicious information, there are further defense layers to protect the project. https://kai-greshake.de/posts/approaches-to-pi-defense/#secure-threads [2024.05.05.]
- **Accept only correctly structured output:** The output of the llm should be built up be two components. The first component contains the answer for if the prompt provided by the user is appropriate, - is a real word, is not obscene, is structured correctly, for example with spaces – either Approved or Not Approved. The second component contains the suggestions with the prompt to the user, if needed.
- **Second llm to verify first llm's output:** A secondary llm checks the output of the main llm and only execute the following tasks if it approves. The idea behind this solution is that an LLM cannot reproduce malicious prompts as well as humans can. Thus, using the first llm's output as the prompt of the send llm it can detect if there has been an attack or not.
- **Second layer defenses:** If the attacker still manages to break the system and use the application in a non-intended way, the products created by it gets into the circulation of the application's ecosystem. Other users can flag these products as wrong ones and it can later be eliminated from circulation.