Introduction to Prompt Engineering

As we have established in the Fundamentals of AI module, Large Language Models (LLMs) generate text based on an initial input. They can range from answers to questions and content creation to solving complex problems. The quality and specificity of the input prompt directly influence the relevance, accuracy, and creativity of the model's response. This input is typically called the `prompt`. A well-engineered prompt often includes clear instructions, contextual details, and constraints to guide the AI's behavior, ensuring the output aligns with the user's needs.

Prompt Engineering

Prompt Engineering refers to designing the LLM's input prompt so that the desired LLM output is generated. Since the prompt is an LLM's only text-based input, prompt engineering is the only way to steer the generated output in the desired direction and influence the model to behave as we want it to. Applying good prompt engineering techniques reduces misinformation and increases usability in an LLM response.

Prompt engineering comprises the instructions itself that are fed to the model. For instance, a prompt like `Write a short paragraph about HackTheBox Academy` will produce a vastly different response than `Write a short poem about HackTheBox Academy`. However, prompt engineering also includes many nuances of the prompt, such as phrasing, clarity, context, and tone. The LLM might generate an entirely different response depending on the nuances of the prompt. Depending on the quality of the responses, we can introduce subtle changes to these nuances in the prompt to nudge the model to generate the responses we want. On top of that, it is important to keep in mind that LLMs are not deterministic. As such, the same prompt may result in different responses each time.

While prompt engineering is typically very problem-specific, some general prompt engineering best practices should be followed when writing an LLM prompt:

- Clarity: Be as clear, unambiguous, and concise as possible to avoid the LLM misinterpreting the prompt or generating vague responses. Provide a sufficient level of detail. For instance, `How do I get all table names in a MySQL database` instead of `How do I get all table names in SQL`.
- Context and Constraints: Provide as much context as possible for the prompt. If you want to add constraints to the response, add them to the prompt and add examples if possible. For instance, `Provide a CSV-formatted list of OWASP Top 10 web vulnerabilities, including the columns 'position','name','description'` instead of `Provide a list of OWASP Top 10 web vulnerabilities`.
- Experimentation: As stated above, subtle changes can significantly affect response quality. Try experimenting with subtle changes in the prompt, note the resulting response quality, and stick with the prompt that produces the best quality.

Recap: OWASP LLM Top 10 & Google SAIF

Before diving into concrete attack techniques, let us take a moment and recap where security vulnerabilities resulting from improper prompt engineering are situated in OWASP's Top 10 for LLM Applications. In this module, we will explore attack techniques for `LLM01:2025 Prompt Injection` and `LLM02:2025 Sensitive Information Disclosure`. LLM02 refers to any security vulnerability resulting in the leakage of sensitive information. We will focus on types of information disclosure resulting from improper prompt engineering or manipulation of the input prompt. Furthermore, LLM01 more generally refers to security vulnerabilities arising from manipulating an LLM's input prompt, including forcing the LLM to behave unintendedly.

In Google's `Secure AI Framework (SAIF)`, which gives broader guidance on how to build secure AI systems resilient to threats, the attacks we will discuss in this module fall under the `Prompt Injection` and `Sensitive Data Disclosure` risks.

Next ➜

Mark Complete & Next