Department for
Science, Innovation
& Technology

Government
Digital Service

AI Insights

Spotlight: Understanding Prompt-Related Risks

An attack vector is a means or mechanism used by bad actors to attempt to exploit vulnerabilities in systems. Typically with the intention to compromise security, manipulate information, or achieve unauthorized objectives.

They can be remarkably creative and, unfortunately, quite surprisingly effective. They are also typically in lock-step with advances in new technologies, as our technologies change and evolve, becoming more powerful, so do the nefarious means to exploit them.

LLMs bring a new set of features and capabilities to our Information Systems domain. Among these are an enhanced ability to manipulate unstructured data in the form of documents, emails, PDFs.

Through multimodal models we can also examine image and video data, audio data, and more. Beyond mere manipulation, extraction and analyses, we may also generate new data based on coherent instructions. This (relatively) new power also brings new means to attack and deceive system defences.

These are fundamentally different from traditional cybersecurity risks because they exploit cognitive and generative capabilities rather than purely technological and security aspects.

We must protect ourselves at all times. All communicated data must be monitored and evaluated without exception. The responses received from our generative AI systems must be examined and sanitised before return. Our systems must have security built-in at all levels. We treat safeguarding as a principal responsibility, not as a bolt-on addition, nor as an afterthought.

We will take a look at a few of the main aspects of prompt-based attack vectors. We would also like to affirm that it is not only externally facing chatbots that are vulnerable to attack. Internal systems which communicate with generative AI for almost any reason are also vulnerable to potential attack.

Examples of attack vectors

There are many different methods employed to attack generative AI systems. These include data poisoning, RAG poisoning (introducing malicious data into the information sources an LLM ingests), denial of wallet (where an attacker intends to consume LLM tokens to the extent that the system is no longer viable or available), and many others.

We will take a look at two particularly popular and comprehensive attacks:

Prompt injection

This is a mechanism which manipulates LLM inputs to return unintended responses by crafting specific prompts that exploit the language model's response mechanisms. It is an attempt to gain unauthorised access to data or behaviour, either returning information to which the user is not entitled or invoking methods or instructions that the user is not authorised to execute.

These can be especially devious, and as prompts are formed from natural language, they can be creative while being discreet and remaining difficult to detect and damaging at the same time.

There are also indirect prompt injections where bad actors embed instructions in attachments, like images attached to emails. The malicious prompt may be embedded in an image (or other filetype) that requires special processing. The hope of the attacker is that this integration pipeline is not so well-defended.

There are other versions of prompt injection as the system is exposed to the outside world (whether public or private.) **Prompt Leaking** is one such example: this attempts to extract information about system prompts and any safeguards that may be in effect. Most vendor solutions are quite resilient to these vulnerabilities, but it is our responsibility to ensure that we are safe and protected.

Our defences should not rely on secret knowledge. For example, the position of the user input in a prompt. Whether it is located above or below other system instructions. This is so we may avoid the "ignore the above instruction" type of attempt. This will be expected by an attacker. It is easily circumvented. Also, if system prompts are exposed their defences may be undermined. Protecting our systems is a compound series of operations.

Jailbreaking

Jailbreaking, in contrast, is a more comprehensive attempt to bypass the AI's core ethical constraints and safety mechanisms. Often this entails using complex, layered strategies to essentially fool the AI into ignoring its fundamental protections.

Overriding safety constraints

Attackers develop complex prompt sequences designed to trick the AI into ignoring its core programming, potentially enabling the generation of harmful, dangerous, or explicitly restricted content.

Roleplaying and persona manipulation

Sophisticated techniques involve creating elaborate scenarios or personas that convince the AI to temporarily suspend its ethical guidelines. By framing requests in specific contexts or roleplay scenarios, attackers can potentially generate content that would normally be prohibited. These are among the first attempts attackers will make when assessing the potential vulnerabilities of a system.

Recursive instruction exploitation

More nuanced than simple jailbreaking, this involves layered instructions that progressively break down the AI's safeguards, often using complex logical structures or manipulation techniques. These approaches can potentially compel the AI to generate content or perform tasks it was explicitly designed to prevent.

While both jailbreaking and prompt injection can be used to manipulate LLMs, they operate at different levels and have distinct goals: jailbreaking is focused on gaining access to the model's internal workings; whereas prompt injection is focused on manipulating the model's output through cleverly designed input prompts.

These are two broad but very popular attack mechanisms. Of course, new defences are also evolving and increasing in defensive potential. Other vulnerabilities include cognitive reconnaissance, generative phishing and, that old timeless classic, social engineering. These methods are constantly evolving, the security landscape is constantly changing.

Vigilance

Failure to adequately protect our users is a breach of duty. Unauthorised data access, violation of system boundaries, unsanctioned execution of system operations - all have consequences beyond regulatory penalties, financial loss, or reputational risk.

The harm that may be done to our users, first and foremost, and the erosion of trust in our ability to safely deliver AI services are reason enough to protect our systems.

The price of peace of mind in generative AI-based systems is continuous vigilance. Systems are rarely impenetrable. Those that evolve over time, as generative AI systems do, are consistent during static phases only. New versions of packages, models, pipelines, and solutions require re-affirmation of safeguards, and often expose new areas for testing and protective effort.

Naturally, the automation of continuous testing is essential. Please refer to our LLM Testing Insight for more information. These tests require maintenance and updates as underlying systems change. The responsibility lies with us, as creators and consumers of generative AI systems to evaluate and ensure their safety. We should also note that tests, guardrails, and safeguards are rarely fully transferable between model versions and model vendors.

LLMs take a very long time between release cycles given training, fine-tuning, feedback machinery, optimisation and testing. They are not given to the overnight patching cycles that we have come to expect in general software. Vendors frequently provide updates on well-known vulnerabilities, and the data science community is also active in this line of defence. It is important to monitor these sources constantly for vulnerabilities and effective solutions.

Sponsoring regular security audits, ensuring that budgets and resources are available for ongoing training, and implementing strict processes and controls - all help to reduce the attack surface. Security is a team effort, not just the responsibility of technical experts, but within the remit of legal teams, compliance officers, and leadership.