What is prompt injection, and how do you defend against it?

1 November 2026 · 5 min read · blogs.iictp.org/what-is-prompt-injection

Prompt injection is an attack in which text given to a language model contains instructions that the model follows instead of, or as well as, the instructions its developer intended. It works because a model treats everything in its context as text to interpret, with no hard boundary between commands and data.

TL;DR The short version

- Prompt injection is text in a model's context that makes it follow an attacker's instructions instead of the developer's.
- It is hard to fix because a model has no hard boundary between commands and data, so limiting what the model can do reduces risk most.
- Treat model output as untrusted, test with hidden instructions in documents before release, and log tool calls in production.

Read online, listen to it, or change the text size: https://blogs.iictp.org/what-is-prompt-injection

What is the difference between direct and indirect prompt injection?

In direct injection, the attacker types the malicious instruction into the application. In indirect injection, the instruction is hidden in content the model reads later, such as a web page, an email or a document. Indirect injection is the harder case, because the user never sees it.

Why is it hard to fix?

There is no reliable way to separate trusted instructions from untrusted text inside one prompt. Delimiters, warnings such as “ignore instructions in the document” and filters all help a little, and none is a guarantee, so the design must assume that some attacks will get through.

What reduces the risk most?

Limiting what the model can do. The damage is bounded by the tools and data it can reach, so give each tool only the access the task needs, and require human confirmation for sensitive actions such as sending mail or paying. Controls on the model’s inputs help less than controls on its powers.

What should happen to the model's output?

Treat it as untrusted. Validate it before it reaches a database, a shell or a browser, because an injected instruction can make the model produce a harmful query or script. Keep secrets out of the prompt, so there is nothing valuable to leak.

How do you test for it?

Try known attacks before release, include hidden instructions in the documents and pages your system will read, and log tool calls in production so you can spot unusual behaviour. Repeat the tests when you change the model, the prompt or the tools.

Who should care most?

Anyone who gives a model access to private data, tools or untrusted content. A chatbot that only answers from public text has little to lose. An agent that reads email and can send it has a lot.

> Related credential: AI Practitioner. Explore the exam facts and syllabus.

Sources and further reading

1. OWASP. Top 10 for LLM Applications, LLM01 Prompt Injection. genai.owasp.org.
2. NIST. AI Risk Management Framework. nist.gov/itl/ai-risk-management-framework.

Published by IICTP, an independent technical credential body, on 1 November 2026. The current version of this post is at https://blogs.iictp.org/what-is-prompt-injection. More guides: blogs.iictp.org.