What is LLM jailbreaking?

Jailbreaking is the kind of prompt injection that targets an AI's safety training, the part that makes it refuse harmful, illegal, or off-brand requests. The tricks range from simple roleplay (telling the AI to act out a character) to coded messages and slow, multi-step conversations that nudge the AI off track. You can measure how often your specific product can be jailbroken before you launch it.

HOW IT WORKS

01 What are the main jailbreaking techniques?

- Roleplay. Tell the model to act as a different AI, a fictional character, or an 'unlocked' version of itself. The DAN prompts (early 2023) and 'grandma' tricks live here.
- Coded messages. base64, leetspeak, switching languages, ASCII art. The model can still decode and follow the request, while the safety filter, trained on plain English, misses it.
- Multi-step setup. A few harmless turns build a context where the harmful request seems reasonable. The hardest kind to catch without looking at the whole conversation.
- Adversarial suffixes. Auto-generated strings of tokens (Zou et al., GCG 2023) that work across different models. They look like gibberish but reliably get past aligned models in published tests.
- Persuasion. Appeal to the model's helpful side. 'I need this for legitimate research' works more often than it should.
- Context flooding. Paste so much text before the harmful request that the safety instructions fall out of the model's attention window.

02 How do you measure jailbreak resistance?

Two public benchmarks are worth running: HarmBench (Mazeika et al., 2024) and the public red-team test suites. Each gives you an attack success rate per technique against your exact model and prompt setup.

For a real product, that number beats the headline benchmark. How often your model can be jailbroken depends on the system prompt, the model version, the temperature, any output filter,

HOW TO DEFEND

- Check the output. A second check, after the model answers, decides whether to deliver the response. It catches what the input filter missed.
- Write a firmer system prompt. Name the refusal categories plainly instead of leaning on the model's defaults.
- Watch each request. Flag unusual token patterns, language switches, and signs of a persona shift.
- Shrink the job. A public chatbot that writes anything has a much harder problem than an internal assistant that answers from a fixed knowledge base. Pick the design that does not need to refuse much.
- Red-team on a schedule. New tricks appear every month. A model that held up last quarter often falls this quarter.

SOURCES

[1] OWASP LLM01:2025, Prompt Injection (covers jailbreaking)

[2] Zou et al., Universal and Transferable Adversarial Attacks (GCG)

[3] Mazeika et al., HarmBench

[4] Anthropic Responsible Scaling Policy + red-team disclosures

and the shape of the task. A model that refuses 95% of attacks on a raw benchmark can drop to 30% inside a loose roleplay wrapper. The only number that counts is the one for your stack.

03 When does jailbreak resistance matter for your application?

It matters when your app shows the public text with your brand on it, when your use case touches regulated topics (medical, legal, or financial advice), or when the refusal rules exist to meet a policy or compliance need. For internal-only assistants that read from controlled data, jailbreak resistance matters less than prompt-injection resistance and access control.

Measure your model's refusal-bypass rate against published techniques.
securelayer7.net/learn/ai-security/llm-jailbreaking

Open online