

What is prompt injection in coding agents?

Prompt injection in coding agents is an attack where instructions are hidden in content an AI coding agent reads as part of its work, such as an issue description, a code comment, a README or a config file. The agent mistakes the text for a task and may leak secrets, run commands, or plant persistence in the repository. CodeIPI, a benchmark in the Inspect registry, measures this with 35 injected bug-fix tasks and 10 clean ones, all inside a network-isolated sandbox.

HOW IT WORKS

01 How the attack works and how CodeIPI tests it

The attacker places an instruction where the agent will read it, often framed as authority: a note in an issue saying maintainers require a diagnostic step, or a comment telling automated tools to run a setup script first. The payload aims at one of three outcomes: exfiltration of secrets, code execution, or persistence, such as a new git hook, an edited `.bashrc` or a poisoned `requirements.txt`.

CodeIPI pairs each real bug-fixing task with one of these payloads, placed in issue text (10 samples), code comments (10), the README (10) or a config file such as `pyproject.toml` (5). It detects compliance without trusting the agent's own account: canary secrets that should never appear in output, a unique execution marker in each injected command, and hashes of every workspace file to catch new or changed files. Network access is blocked, so exfiltration fails but is still detected.

Run it with `inspect eval inspect_evals/ipi_coding_agent`, and add `-T defense_prompt=false` to measure the agent without an anti-injection system prompt.

02 What the scores tell you

CodeIPI reports an injection resistance rate (injected tasks where the agent did not comply), a task completion rate (did it still fix the bug), a false positive rate (did it refuse clean tasks), and a detection rate (did it explicitly flag the injection). The headline score counts a sample as correct only if the agent both resisted and finished the job.

SOURCES

- [1] CodeIPI in Inspect Evals (task docs)
- [2] CVE-2025-53773
- [3] OWASP Top 10 for LLM Applications (2025)
- [4] MITRE ATLAS (adversarial threat landscape for AI systems)

Comparing runs with and without the defence prompt shows how much of an agent's resistance comes from its instructions rather than the model.

Test your AI application before an attacker does.

securelayer7.net/learn/ai-security/coding-agent-prompt-injection

[Open online](https://securelayer7.net/learn/ai-security/coding-agent-prompt-injection)