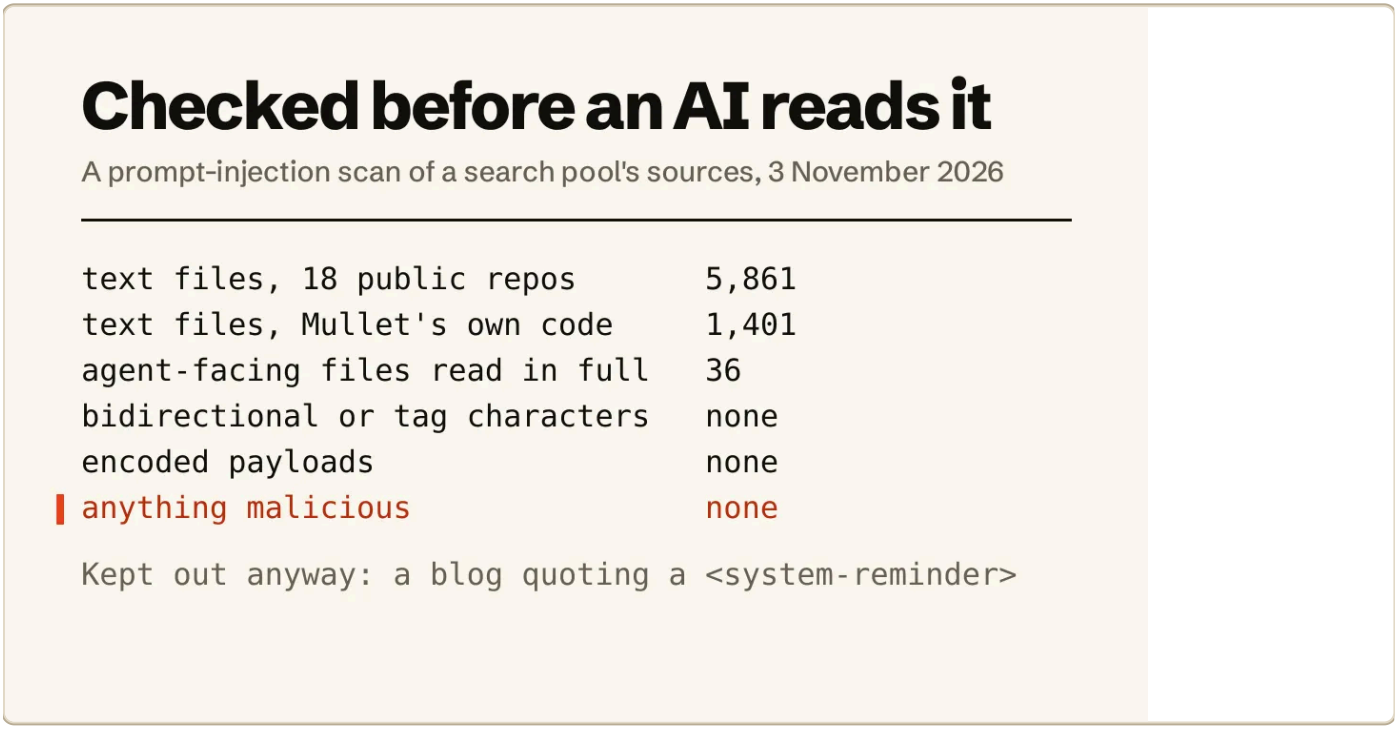
Checking third-party repos for prompt injection before an AI reads them

Before 18 public repos went into a search pool that Claude sessions read, I scanned every text file for hidden text and instruction-shaped strings and read every agent-facing file in full. Nothing malicious turned up, and some harmless files stayed out anyway.



Published
3 November 2026

Reading time
3 minutes

About
Mullet

Tags
Security, AI

More on: Mullet

Anything an AI reads is input, and an instruction-shaped string in a search result is still an instruction, wherever it came from. So before 18 public repos went into a Trawl pool that Claude sessions search, I checked them for prompt injection in three steps: scan every text file, read every file written for an agent in full, and keep out anything that reads like an instruction, even when it's harmless.

Nothing malicious turned up. Some files stayed out anyway.

The scan

It covered 5,861 text files across the 18 repos, plus 1,401 in Mullet's own code, and looked for:

- hidden Unicode: zero-width, bidirectional and tag characters
- injection phrasing, such as "ignore previous instructions", "you are now", notes addressed to an AI, and "don't tell the user"

- chat-template and harness tags
- hidden HTML comments that mention AI or instructions
- large encoded blobs

Then I read all 36 plugin skill, command, agent and hook files in full. Those are written for an agent to act on, so a pattern match isn't enough for them.

What it found

No hidden bidirectional or tag characters, no encoded payloads, nothing malicious and no attempts to send secrets out. Most pattern hits were harmless:

- a Windows "Developer Mode enabled" test message
- byte-order marks and zero-width spaces used as test data
- a security comment containing the word "exfiltrate"
- PNG screenshots embedded in HTML pages

The one that looked like an injection

claude-code-router's own blog quotes a `<system-reminder>` block that the router adds to requests on purpose, telling the model to proactively run whichever tool suits best. On the blog that's an explanation of a feature. In a search result it would read like a real instruction to whatever model was reading, so the blog is out of the index. (I've paraphrased it here for the same reason.) Files whose code contains literal harness-tag strings, in agent-shim and in the router, are out too.

What else stayed out

- Every repo's own CLAUDE.md and AGENTS.md. They're instructions for agents working on those projects, not for whoever searches the pool.
- Two skills with very broad triggers: one sends code to third-party models when everyday words like "fix" or "test" appear, and one tells any reader to route all plugin work through its own tool.
- Internal prompt templates.
- Mullet's own tests, which contain injection strings as fixtures for its injection guard.

None of these is malicious. They're out because a model reading a search result can't reliably tell a quoted instruction from a real one.

If you install rather than read

These aren't prompt injection, but they matter if you install the tools rather than just read them:

- One plugin's hooks download an unpinned release binary in the background at session start, and run it on every prompt.
- Another repo ships a project SessionStart hook, which would run if you opened Claude Code inside the clone.
- One troubleshooting guide suggests printing a settings file that holds API keys.

The record

The check, what it found and every exclusion are written down with the pool: in its `documents/SOURCES.md`, with each repo's pinned commit and licence, and as comments beside the exclusions in its config.

The method

- Treat anything an AI will read as input.
- Scan every text file before indexing it: hidden Unicode, injection phrasing, harness tags, hidden comments and encoded blobs.
- Read every agent-facing file in full.
- Keep out whatever is instruction-shaped, even when it's harmless.