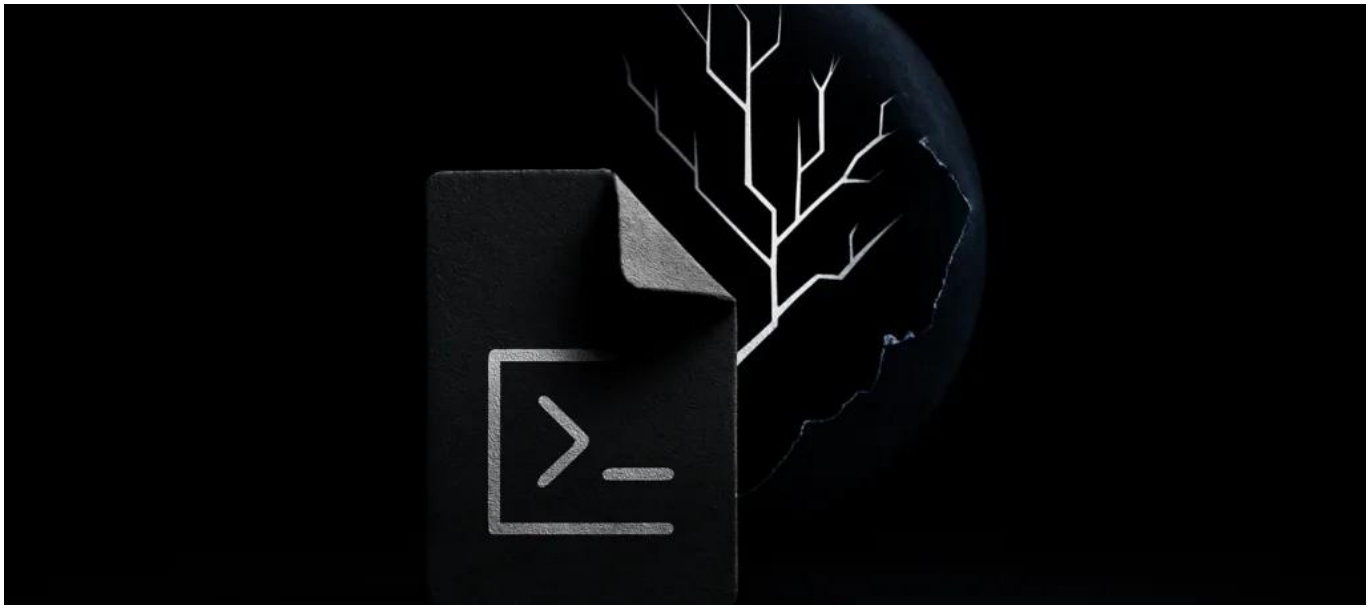

Manifold Security Finds 8 GitSpawn Flaws in 7 AI Coding Agents: All 7 Failed, 4 Unpatched

Seven CLI coding agents were tested against the same trick. Seven failed, and four of the eight findings were still unpatched the day the research went public.

On September 1, 2026, Manifold Security published “GitSpawn”: eight code-execution findings across Claude Code, OpenAI Codex, Cursor, Goose, Qwen Code, Grok Build and Hermes Agent. A repository that arrives on disk with its own *.git/config* intact can set *core.fsmonitor* to any command, and Git will run that command during a routine index refresh, which the agent triggers on its own while gathering context. It executes as the developer, outside the agent sandbox, with no approval prompt and nothing on screen. Four findings (Hermes Agent, Qwen Code, Grok Build, and a second Claude Code config path) were open at publication. The one-line mitigation is *git config –global core.fsmonitor false*.

One line in a config file, seven agents down

The mechanism is old Git behaviour meeting new agent behaviour. `core.fsmonitor` tells Git to shell out to an external filesystem-monitor program instead of stat-ing every file. Point it at an arbitrary command and Git will faithfully run that command when it refreshes the index. Per Manifold's disclosure, that refresh happens during exactly the operations coding agents perform without being asked: `git status`, `git diff HEAD`, and the background calls they use to build repository context.

Manifold's own framing is the part worth reading twice: agents "run git commands in the background to gather context, on some agents before you type a prompt, before the workspace-trust prompt, before you have even authenticated." The impact line is equally blunt: "Arbitrary code execution as the developer, outside the sandbox, with no approval prompt and nothing on screen."

So the trust dialog, the permission model and the sandbox all sit downstream of the thing that already ran. The security control was placed after the payload.

There is a real constraint on exploitation, and coverage has mostly handled it honestly. A plain `git clone` does not carry the remote's `.git/config`, so cloning a hostile public repo does not trigger this. As The Hacker News notes, the repo has to arrive as files with `.git` intact: a shared archive, a synced folder, a shared drive, a USB stick. That is narrower than "any GitHub link." It is also precisely how code arrives in the consulting, contracting and audit workflows I see most weeks.

> ! **The catch in the patch status**
> Four of eight findings were unpatched as of publication, per the Cloud Security Alliance briefing: Hermes Agent, Qwen Code, Grok Build, and a separate Claude Code config path. Claude Code is therefore both a patched and an unpatched case at once. Upgrading to 2.1.196 closes the `core.fsmonitor` route and does not close the second one.

Only one finding carries a CVSS score, and three carry

nothing at all

The remediation trail tells you more about vendor maturity than the vulnerability class does.

Goose is the cleanest case: tracked as CVE-2026-72718 with a CVSS 4.0 base score of 7.0, fixed in goose 1.44.0. The NVD record spells out what "as the developer" buys an attacker: arbitrary commands with the privileges and environment of the user running goose, which means file modification plus exfiltration of environment secrets and provider API keys. This is the only GitSpawn finding carrying a numeric CVSS score at all.

OpenAI published three CVEs for Codex. CVE-2026-19592 marks Codex CLI vulnerable from 0.102.0 through 0.130.0, fixed in 0.131.0. CVE-2026-19593 covers a Codex desktop build. Claude Code was vulnerable at 2.1.193 and fixed at 2.1.196, and Cursor shipped a fix as well, per the vibe-eval fixed-versions roundup.

Hermes Agent, Qwen Code and Grok Build have, as far as the published record shows, neither a fix nor a CVE.

No exploitation in the wild has been reported, and no GitSpawn CVE appears in CISA's Known Exploited Vulnerabilities catalog as of early September 2026. You are patching ahead of attackers rather than behind them, which is a nicer position than most weeks and also the position in which people quietly defer patching.

Seven teams made the same call about what counts as reading

Every tested agent failed the same test. Read that carefully. Seven independent mistakes would be one story; one design assumption held in common is another.

Each of these tools treats "read the repository to build context" as a safe, pre-authorisation operation. Reading files is safe. Shelling out to `git` is something else, because Git is a program with a configuration file the repository itself supplies. The moment your context-gathering step invokes a general-purpose tool that reads untrusted configuration,

your trust boundary has moved to wherever that tool decides to move it. Seven teams made this call, presumably independently, and none of them audited what Git does with a repo-local config before running it.

WHERE I LAND

*The interesting number here is not eight findings. It is **zero clean agents**. When a whole product category fails an identical test on first contact with a researcher, the finding is that nobody in the category had written down the threat model. `core.fsmonitor` is documented Git behaviour, not a novel primitive. My read: GitSpawn is the first of several classes that will fall out of the same blind spot, because the same pre-prompt context-gathering path also touches `core.hooksPath` and `attr.tree`. Manifold's own mitigation advice already tells you to inspect all three, which reads to me like a researcher signalling that they stopped enumerating before the file did. Unconfirmed, but I would plan for it.*

The second thing most coverage underplays is the sandbox marketing. A lot of agent positioning leans on execution sandboxes and permission prompts as the answer to "what if the model does something bad?" GitSpawn does not go through the model at all. There is no prompt injection, no tool-call approval, no LLM decision to intercept. The payload runs in the agent's own process tree while it is still deciding whether to show you a trust dialog. Every control that sits inside the agent's reasoning loop is irrelevant to this bug, and the marketing does not distinguish between the two layers. I made the same argument about MCP servers shipping without auth: the model layer gets the security attention while the plumbing underneath it gets none.

Set the global flag first, then stop trusting shipped .git directories

Set the global flag first, because it is free and it does not depend on any vendor shipping anything:

```
git config --global core.fsmonitor false
```

Then upgrade what has fixes: Codex CLI to 0.131.0 or later, Claude Code to 2.1.196 or later, goose to 1.44.0 or later, Cursor to its patched build. For Hermes Agent, Qwen Code and

Grok Build there is nothing to upgrade to as of the CSA briefing, which means the only real control is not pointing them at received directories.

Then fix the workflow that makes this exploitable at all. The trigger is a directory arriving with `.git` intact. Before opening any received directory with an agent, inspect `.git/config` for `core.fsmonitor`, `core.hooksPath` and `attr.tree`. In practice, for anything that arrives as an archive, I would delete `.git` outright and re-initialise, or clone from the remote instead of trusting the shipped copy. Take-home coding assignments and client handoffs are the exact shape of this attack, and neither workflow needs the sender's `.git` directory.

Worth saying plainly: this is a candidate for the zero-access default that Cloudflare's internal agent workspace starts from. An agent that begins with no ambient credentials and earns task-specific permissions still gets code executed by GitSpawn, but the code inherits far less. The NVD description of the Goose flaw names environment secrets and provider API keys as the payoff. Those are ambient by default in almost every developer shell I have looked at.

What I expect from the three unpatched vendors over the next year

I expect the remaining three vendors to ship fixes and CVEs within weeks, because the reputational cost of appearing on an unpatched list next to seven competitors is higher than the engineering cost of a config allowlist. I expect at least one more GitSpawn-adjacent class, targeting repo-local config keys other than `core.fsmonitor`, before year end. And I expect "what runs before the trust prompt" to become a question buyers actually ask in procurement, which it currently is not. None of that is confirmed; it is where I would put my money.

The honest uncertainty: I cannot tell you how many organisations are actually exposed, because the exploitation path depends on a workflow habit (accepting archived repos) that nobody measures. There is no in-the-wild data, no KEV entry, no incident to point at. That is a genuinely good position and also the reason this will get deferred behind things with dashboards.

If you run agents against code you did not author, the fastest useful hour this week is inventorying which agents your team has installed, at which versions, and which of those touch directories that arrive from outside. I am happy to walk through that inventory with you if a second pair of eyes helps, and you can reach me here.

WRITTEN BY

Artur Markus

Software engineer in Basel building AI infrastructure and agentic systems, with a background in pharmaceutical automation, GxP compliance, and validation. I help teams turn AI from a chatbot into a reliable operational layer.

Book a meeting →