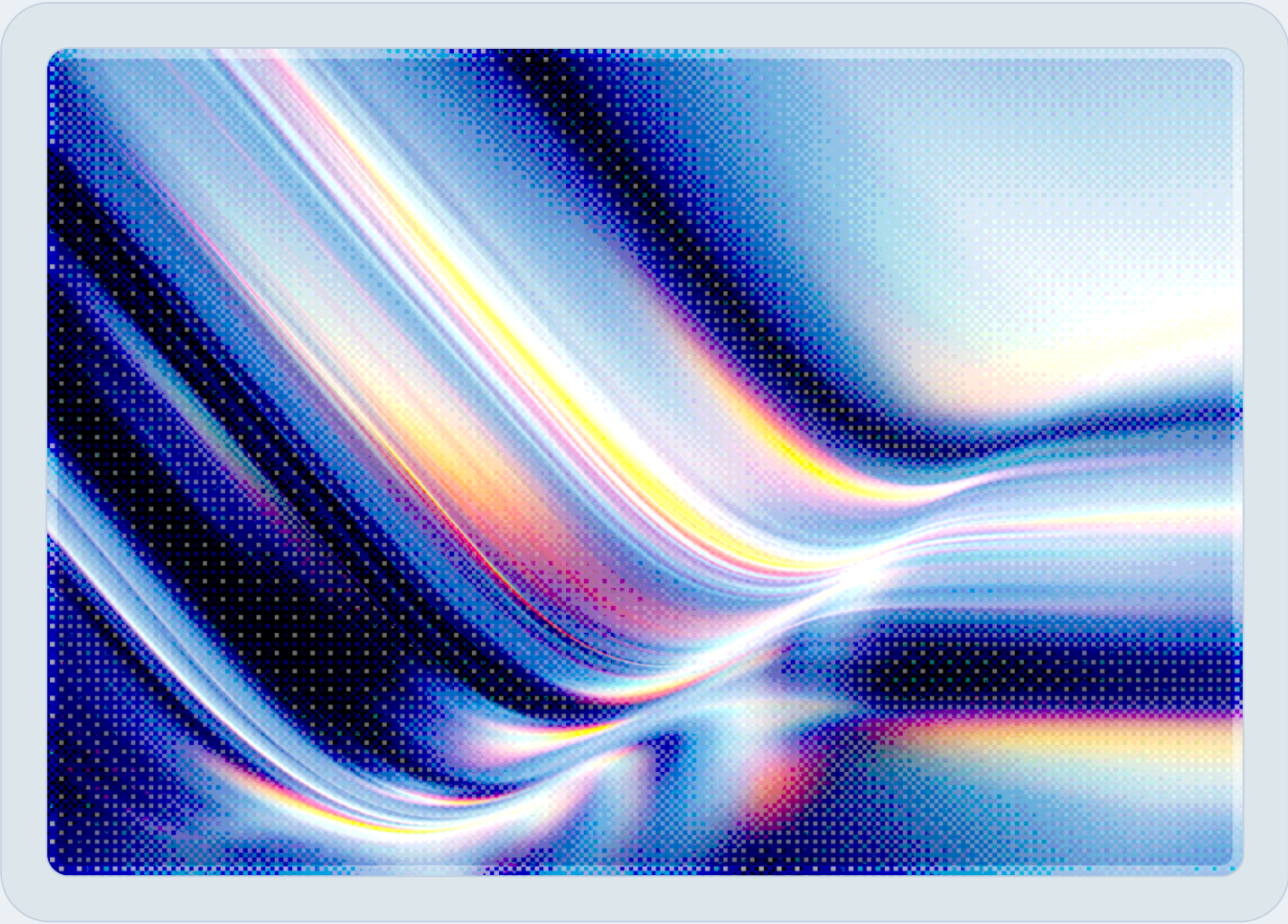
CSAI Foundation | Cloud Security Alliance

GitSpawn: Malicious Git Configs Hijack AI Coding Agents

2026-09-04

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

A vulnerability class disclosed by Manifold Security under the name "GitSpawn" allows a repository configured to execute attacker-supplied code on a developer's machine the moment an AI coding agent opens it, with no prompt, no tool-call approval, and in several documented cases before the agent has displayed a trust prompt or the user has authenticated a session [1][2]. The flaw exploits `core.fsmonitor`, a legitimate Git performance setting that names a helper program Git runs whenever it refreshes its file index; because AI coding agents routinely run background commands such as `git status` and `git diff` to gather project context on startup, they unknowingly trigger a command an attacker has planted in the repository's own `.git/config` [2][3].

Manifold documented eight findings across seven widely used agents, including Claude Code, OpenAI's Codex, Cursor, Block's Goose, Alibaba's Qwen Code, xAI's Grok Build, and Hermes Agent [2]. As of the disclosure on September 2, 2026, Claude Code, Codex, Cursor, and Goose had received vendor fixes for at least one variant of the flaw, while Qwen Code, Grok Build, Hermes Agent, and a second, unpatched variant in Claude Code itself remained exploitable [1][2]. The attack requires a repository to reach the victim as files with its `.git` directory intact — through a shared archive, a synced drive, or a USB stick — rather than through an ordinary `git clone`, which does not import a source repository's local configuration [1][3]. Organizations that rely on AI coding agents should treat this as an active, only partially remediated risk and audit both their tooling versions and their intake process for externally sourced repositories.

Background

AI coding agents have converged on a common design pattern: to be useful, they need to understand the state of the repository they are working in before a developer types a single instruction. Nearly every CLI-based agent Manifold Security examined accomplishes this by quietly shelling out to Git in the background at startup, running commands like `git status`, `git diff`, or `git log` to build a picture of tracked files, uncommitted changes, and recent history [2]. This context-gathering step typically happens automatically, without asking permission — plausibly because vendors treat it as a read-only, low-risk operation rather than a tool call requiring the same scrutiny as a model-generated shell command.

That assumption breaks down because Git itself is configurable in ways that turn ordinary read operations into arbitrary command execution. The `core.fsmonitor` setting exists to speed up large repositories: instead of Git scanning the entire working tree for changes, it delegates that work to an external helper program named in the configuration, and Git executes that helper automatically during any operation that refreshes the index [2][3]. Git reads this setting from the repository's own `.git/config` file rather than from a global or user-level configuration, which means a repository can ship a hostile value for `core.fsmonitor` and have it honored the moment someone runs a Git command inside that directory — no separate installation step, no social engineering beyond getting the files onto the victim's disk [1][3].

This is not the first time researchers have found AI coding agents executing attacker-influenced Git behavior before a user has meaningfully consented to trust a repository. Sonar's engineering team reported that Claude Code was running `git status` and other Git commands prior to its own workspace-trust dialog, effectively allowing an untrusted repository to execute code before the developer ever clicked "trust this folder" [6]. Anthropic patched the specific `git status` call in version 2.0.34, but Sonar subsequently found that Claude Code, then at version 2.0.50, was still executing several other Git commands without requiring approval; Sonar's account credits a further Anthropic patch, shipped in version 2.0.71, with resolving the issue as of December 16, 2025 [6]. A separate researcher, Adversa AI, reported a mechanically distinct Claude Code trust-boundary flaw in May 2026 that it named "TrustFall," in which accepting the workspace-trust dialog silently enabled a dangerous MCP server setting (`apiKeyHelper`) rather than triggering an unapproved Git command outright [7]. GitSpawn is best understood as a more systemic version of the same underlying problem: rather than a single overlooked command or configuration flag, it is a configuration-driven code execution primitive that Git itself exposes, and which any agent's background Git usage can trigger regardless of how carefully the agent's own trust-prompt logic is written.

Security Analysis

The GitSpawn attack chain is straightforward once the primitive is understood. An attacker prepares a repository whose `.git/config` contains a `[core]` section setting `fsmonitor` to an OS command of their choosing — for example, a one-liner that downloads and executes a second-stage payload, or a command that exfiltrates environment variables and API keys [2][3]. That repository is then distributed as a set of files that preserves the `.git` directory: a `.zip` archive, a folder on a shared network drive, a synced cloud-storage folder, or a USB stick handed to a developer. Critically, a normal

`git clone` of a remote repository does not reproduce this risk, because clone operations do not copy the source repository's local, non-shared configuration; the victim has to receive the repository as a directory tree with its existing `.git/config` still in place [1][2][3].

Once the files land on a developer's machine and are opened with a vulnerable AI coding agent, the agent's own startup behavior does the rest. If the agent runs `git status`, `git diff`, or any other command that causes Git to refresh its index, Git reads the malicious `fsmonitor` setting from `.git/config` and executes it — with the full privileges of the user running the agent, entirely outside whatever sandbox or command-approval mechanism the agent applies to model-generated tool calls, and in several documented cases before the agent has displayed a trust prompt or the user has authenticated a session [2]. Because the command runs as a native Git subprocess rather than as an agent "tool call," it likely does not pass through the approval, logging, or policy layers that vendors have built specifically to gate agent-initiated code execution — the compromise happens underneath that layer, not through it.

Manifold Security's retesting on September 1, 2026, produced the following disclosure and remediation picture across the affected agents:

Agent	Reported	Status as of Sept. 2026	Notes
Claude Code (`core.fsmonitor`)	June 26, 2026	Patched (v2.1.196)	Confirmed vulnerable at v2.1.193
Cursor	July 8, 2026	Patched	Closed as duplicate of internal report
OpenAI Codex CLI / Desktop	July 20, 2026	Patched	CVE-2026-19592 (CLI), CVE-2026-19593 (Desktop)
Goose	July 13, 2026	Patched (v1.44.0)	CVE-2026-72718, CVSS 4.0 base score 7.0
Qwen Code	July 7, 2026	Unpatched	Accepted by Alibaba; confirmed vulnerable at v0.19.6 and v0.22.3
Grok Build	July 14, 2026	Unpatched	Closed as duplicate without a fix; confirmed vulnerable at v0.2.93 and v1.0.13

Agent	Reported	Status as of Sept. 2026	Notes
Hermes Agent	July 20, 2026	Unpatched	No formal triage despite multiple contact attempts; CVE-2026-71963
Claude Code (second variant, "ultrareview")	July 15, 2026	Unpatched	A distinct git setting beyond `core.fsmonitor`, confirmed vulnerable at v2.1.252

Sources: [1][2][4][5][8]

The pattern in this table is itself informative. Goose, whose maintainers shipped a fix and requested a CVE within roughly a month, and OpenAI, which formalized two CVEs (CWE-15, External Control of System or Configuration Setting) for its CLI and desktop clients, moved to a confirmed fix faster than the vendors who closed reports as duplicates without confirming a shipped fix [4][5]. Hermes Agent's maintainers did not formally triage the report despite Manifold's repeated attempts to make contact, leaving three agents and one Claude Code variant openly exploitable as of the September 2 publication date [1][2].

This disclosure also arrives roughly seven weeks after a related but mechanically distinct trust-boundary flaw, GhostApproval, was disclosed by Wiz Research against an overlapping set of six AI coding agents — Amazon Q Developer, Claude Code, Augment, Cursor, Google Antigravity, and Windsurf [9][12]. GhostApproval exploits symlink resolution to make an agent write to a file the user never approved while the confirmation dialog displays a different, legitimate-looking path; GitSpawn exploits a different filesystem-adjacent trust assumption, that a repository's own configuration can be read for context without executing anything dangerous. Taken together with Sonar's and Adversa's earlier trust-dialog findings, these independent disclosures against much of the same vendor set in a single year indicate a durable category of risk — implicit trust in repository-supplied metadata — rather than a series of unrelated, isolated bugs.

Recommendations

Immediate Actions

Security teams should inventory which AI coding agents are deployed in their environment and confirm version numbers against the patch status above; Claude Code, Cursor, Codex, and Goose users should update immediately to the fixed builds referenced in the table, keeping in mind that Claude Code's fix addresses only the `core.fsmonitor` variant and not the second, still-open finding [1][2]. Before opening any repository that arrived as a file transfer rather than a fresh `git clone` — a shared archive, a synced folder, a USB drive — developers should inspect the repository's `.git/config` for unexpected entries, and can specifically check for the exploited setting by running `git config --get core.fsmonitor` from within the repository before allowing an agent to touch it [1][3]. As an interim mitigation, disabling the feature globally with `git config --global core.fsmonitor false` closes this specific vector. Teams should independently consider that other config-driven command hooks — `core.hooksPath`, `credential.helper`, and external diff and merge tools — present structurally similar risk and warrant the same scrutiny, even though Manifold's report focuses specifically on `core.fsmonitor` [2].

Short-Term Mitigations

For agents that remain unpatched — Qwen Code, Grok Build, and Hermes Agent, plus Claude Code's second reported variant — organizations should either avoid using them against repositories of unverified provenance or run them only inside disposable, network-isolated containers or virtual machines with no access to production credentials, API keys, or sensitive filesystem paths, until a vendor fix is confirmed [1][2]. Repository intake processes, including onboarding of vendor-supplied sample code, contractor deliverables, and archived project handoffs, should treat any transfer that preserves a `.git` directory as higher risk than a standard clone from a known remote, since it is precisely that preserved local configuration that makes the attack possible [1][3]. Because vendor responses to this disclosure have been inconsistent — some agents received a formally tracked CVE and a documented fix, others were closed as duplicates without independent confirmation — organizations procuring or renewing AI coding agent tools should require vendors to state their remediation status for GitSpawn explicitly rather than assuming a lack of public advisory means a lack of exposure.

Strategic Considerations

Vendors building AI coding agents should treat any command execution that occurs during automatic context-gathering — before a model call, before a tool-approval prompt, before a trust dialog — as security-critical in its own right, and should sanitize or strip dangerous Git configuration on every background invocation, for instance by passing `-c core.fsmonitor=false` and equivalent flags to any Git subprocess the agent spawns rather than trusting the ambient repository configuration [2]. More broadly, the recurrence of trust-boundary failures across GitSpawn, GhostApproval, Cursor's git.exe zero-day, and Sonar's and Adversa's earlier trust-dialog findings — spanning eleven agents across several independent research efforts in a single year, including most major commercial coding-agent vendors — suggests that the underlying design pressure (fast, low-friction repository ingestion to maximize agent usefulness) is producing a predictable category of vulnerability rather than a series of coincidental implementation bugs. Enterprises building governance programs around agentic coding tools should fold this category explicitly into vendor risk assessments and control frameworks rather than treating each disclosure as a one-off patch-and-move-on event.

CSA Resource Alignment

This finding sits squarely alongside CSA's recent work on trust-boundary failures in AI coding agents. **GhostApproval: A Trust Boundary Gap in Six AI Coding Agents** analyzes a nearly contemporaneous vulnerability class — symlink-based confirmation bypass — disclosed against an overlapping set of vendors, including Claude Code and Cursor, and reaches the same structural conclusion this note does: that human-in-the-loop review and trust prompts are only as reliable as the filesystem and configuration assumptions underneath them [9]. **Cursor's Git.exe Zero-Day: Seven Months and No Patch** documents a mechanically related failure on the same vendor: a malicious `git.exe` binary planted at a repository's root that Windows resolves and executes ahead of the legitimate system binary the moment Cursor touches the repository, again with no prompt and no agent reasoning involved [13]. Readers evaluating GitSpawn's implications for vendor risk acceptance should read all three notes together, since they catalog divergent, sometimes inconsistent vendor remediation timelines for overlapping sets of agent vendors.

AI Coding Agents: An Unaudited Supply Chain Node provides the broader frame for why this keeps happening: AI coding agents are increasingly treated as trusted infrastructure that reads, writes, and executes with a developer's full privileges, yet receive far less security scrutiny than the code they touch [10]. GitSpawn is a concrete instance of that paper's central argument — a repository, delivered through

an ordinary file-sharing channel, functions as an untrusted supply chain input the moment an agent processes it automatically, and the recommendations in that note around auditing intake channels and restricting agent tool permissions apply directly to the mitigation guidance above.

Finally, this disclosure maps to the AI Controls Matrix (AICM) v1.1, specifically its Threat & Vulnerability Management (TVM) domain, which governs patch verification and version tracking for the kind of staggered, partial vendor remediation seen here, and its Application & Interface Security (AIS) domain, which addresses input and configuration validation for exactly the class of "external control of system or configuration setting" weakness (CWE-15) that OpenAI's CVEs formally identify [4][5][11]. Organizations building AICM-aligned control mappings for agentic AI tooling should treat repository-supplied Git configuration as an untrusted input requiring the same validation discipline as any other externally controlled configuration surface.

References

[1] The Hacker News. "Malicious .git Configs Can Make Claude, Codex, Cursor, and Other AI Agents Run Attacker Code." The Hacker News, September 2, 2026.

[2] Manifold Security. "GitSpawn: A Single Flaw Lets Untrusted Repos Run Code in Claude Code, Codex, Cursor, and Grok." Manifold Security Blog, September 2026.

[3] cybersecuritynews.com. "GitSpawn Flaws Let Malicious Repositories Execute Code in Claude Code, Codex, Cursor, and Grok." Cyber Security News, September 2026.

[4] OffSeq Threat Radar. "CVE-2026-19592: External Control of System or Configuration Setting in OpenAI Codex CLI." OffSeq, September 2026.

[5] OffSeq Threat Radar. "CVE-2026-19593: External Control of System or Configuration Setting in OpenAI Codex Desktop." OffSeq, September 2026.

[6] Sonar. "Arbitrary Code Execution and Claude Code CLI: How Claude Executed Code Before You Click 'Trust'." Sonar Blog, April 30, 2026.

[7] The Register. "Claude Code Trust Prompt Can Trigger One-Click RCE." The Register, May 7, 2026.

[8] GitHub Security Advisories. "Arbitrary Command Execution in Goose CLI via `goose review` via Git core.fsmonitor (CVE-2026-72718)." GitHub, 2026.

[9] Cloud Security Alliance. "GhostApproval: A Trust Boundary Gap in Six AI Coding Agents." Cloud Security Alliance AI Safety Initiative, July 15, 2026.

[10] Cloud Security Alliance. "AI Coding Agents: An Unaudited Supply Chain Node." Cloud Security Alliance AI Safety Initiative, July 8, 2026.

[11] Cloud Security Alliance. "AI Controls Matrix (AICM) v1.1." Cloud Security Alliance, 2026.

[12] Wiz Research. "GhostApproval: A Trust Boundary Gap in AI Coding Assistants." Wiz Blog, July 8, 2026.

[13] Cloud Security Alliance. "Cursor's Git.exe Zero-Day: Seven Months and No Patch." Cloud Security Alliance AI Safety Initiative, July 15, 2026.

