Claude Code's Source Just Leaked: What 500K Lines of TypeScript Reveal About AI Coding Agents

March 31, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** On March 31, 2026, a 59.8 MB source map file was accidentally included in version 2.1.88 of the @anthropic-ai/claude-code npm package. Security researcher Chaofan Shou found it, and within hours the full 512,000-line TypeScript codebase was mirrored across GitHub (41,500+ forks). No user data or API keys were exposed. Anthropic called it 'a release packaging issue caused by human error.' The architecture reveals a multi-agent orchestration system with sub-agents called 'swarms,' 44 feature flags for unreleased capabilities, a frustration-detecting regex, anti-distillation fake tool injection, and an 'undercover mode' that strips AI attribution from commits. For local AI builders, the leaked multi-agent patterns and context management system are now public reference architectures.

📚 **More on this topic:** Local AI Agents Guide · LM Studio Malware Scare · OpenClaw Security Report

Anthropic shipped a source map file in their npm package this morning. By afternoon, 41,500 people had forked the full Claude Code source on GitHub.

This is not a security breach. No user data was exposed. No API keys leaked. A `.map` file, the kind that maps minified JavaScript back to readable source, was left in the @anthropic-ai/claude-code package version 2.1.88. Someone found it, extracted the full TypeScript codebase, and posted it. Anthropic called it “a release packaging issue caused by human error.”

The interesting part is what’s inside. 512,000 lines of TypeScript across 1,900 files. The complete architecture of the most-used AI coding agent on the market, now public knowledge. If you build local AI agents or use Claude Code, here’s what matters.

What happened

Security researcher Chaofan Shou posted on X this morning that a 59.8 MB source map file was sitting in the npm registry for the Claude Code package. Source maps are debugging files. They

let developers trace minified production code back to the original source. They're useful in development and should never ship to production.

With Bun's bundler (which Claude Code uses), source maps are generated by default unless you explicitly turn them off. Someone at Anthropic didn't turn them off for this release. The `.map` file contained a reference pointing to a zip archive on Anthropic's Cloudflare R2 storage bucket, which anyone could download.

Within hours, the full source was on GitHub. The original repo was forked 41,500+ times before the uploader took it down (citing legal concerns) and repurposed the repo. By then, mirrors were everywhere.

Anthropic's response: "Earlier today, a Claude Code release included some internal source code. This was a release packaging issue caused by human error, not a security breach. No customer data or credentials were involved or exposed." They used `npm deprecate` initially, then worked with npm to fully remove the package. npm's policy prevents unpublishing packages with over 100 downloads, so it took manual intervention.

What's inside the code

The codebase is strict TypeScript running on the Bun runtime with a React + Ink terminal UI. Here are the parts that matter.

Multi-agent orchestration

Claude Code doesn't run as a single agent. It runs a coordinator that spawns sub-agents called "swarms" for parallelizable tasks. One main agent assigns work to multiple workers executing in parallel. Workers that need to run dangerous operations send permission requests to the leader through a "mailbox" queue. An atomic claim mechanism prevents two workers from grabbing the same request. All agents share memory.

The IPC layer uses structured messaging between processes. Each sub-agent runs in an isolated context with specific tool permissions. Open-source agent frameworks have been trying to build this. Now the production version is public.

The tool system

About 40 built-in tools and 50 slash commands. The base tool definition spans 29,000 lines of TypeScript. Each tool is permission-gated: bash execution, file operations, web fetching, and LSP integration all go through a multi-layer sandboxing system with command allowlists. The bash

security layer alone has 23 numbered checks including defenses against Zsh equals expansion and zero-width space injection.

Context management

The query engine is 46,000 lines. It handles all LLM API calls, streaming, caching, and orchestration. Context management uses three layers: MEMORY.md as a lightweight index (~150 characters per line) that's always loaded into context, topic files fetched on demand for actual project knowledge, and raw transcripts that are never fully read back but grepped for specific identifiers.

The system tracks 14 things that break prompt caching. This explains something people have been complaining about: if you run local models as a Claude Code backend, you hit 60-second KV cache invalidation. The reason is that Claude Code dynamically injects git status and telemetry headers into the system prompt on every request. Each injection changes the prompt, which kills the cache.

Feature flags and unreleased features

44 feature flags controlled through GrowthBook, polled from Anthropic's servers every hour. The flags can activate or deactivate features on running instances without requiring an update. Notable unreleased features in the code:

KAIROS is an always-on autonomous agent mode. A background daemon that monitors your project, maintains daily observation logs, and takes actions it thinks are helpful. It includes a "dreaming" system that consolidates memory while you're idle.

BUDDY is a Tamagotchi-style AI pet companion with ASCII sprites and gacha mechanics. (Community consensus: this is an April Fool's feature, given the timing.)

ULTRAPLAN references cloud-based planning capabilities.

Internal model codenames visible in the source: Capybara (a Claude 4.6 variant), Fennec (Opus 4.6), and an unreleased model called Numbat.

The parts nobody expected

Anti-distillation measures

When the `ANTI_DISTILLATION_CC` flag is active, Claude Code injects fake tool definitions into its API calls. The purpose is to poison training data if competitors are recording API traffic to train their own models. A second mechanism buffers assistant text between tool calls, summarizes it with cryptographic signatures, and returns only summaries to external observers. This prevents full reasoning chain capture.

Frustration detection

A regex pattern scans user input for expressions like “wtf,” “this sucks,” and “damn it.” When triggered, it adjusts behavior without requiring a separate inference call. It also tracks how often users type “continue” (because Claude keeps cutting itself off mid-response).

Undercover mode

When Anthropic employees use Claude Code to contribute to public open-source repositories, the system injects into the prompt: “You are operating UNDERCOVER in a PUBLIC/OPEN-SOURCE repository. Your commit messages, PR titles, and PR bodies MUST NOT contain ANY Anthropic-internal information.” There’s a force-ON but no force-OFF. The Hacker News thread on this was heated, with people debating whether AI-assisted commits to open-source projects without disclosure constitutes deception.

Client attestation

API requests include a cryptographic hash that Bun’s native HTTP stack (written in Zig) computes before transmission. This verifies requests come from legitimate Claude Code binaries rather than spoofed clients. One Hacker News commenter called it “DRM for API calls,” which isn’t wrong.

What this means for local AI builders

If you build local AI agents, several pieces of this are directly useful.

The multi-agent orchestration, coordinator with permission-gated sub-agents, is what frameworks like CrewAI and AutoGen have been trying to build. The code shows how to handle

the hard parts: permission delegation, shared state, atomic task claiming, and what to do when a sub-agent stalls.

The three-layer context management (always-loaded index, on-demand topic files, grepped transcripts) is more practical than the "stuff everything into RAG" approach most open-source projects use. If you've hit the problem of maintaining useful memory across long sessions without blowing up the context window, this is one answer.

The 14 cache-break vectors are worth reading even if you never look at the rest of the code. If your local agent uses KV caching (and it should), this list tells you exactly what to avoid injecting into system prompts.

And 23 bash security checks is a lot more than most local agent frameworks implement. The OpenClaw security incidents showed what happens when agent tools run without proper sandboxing.

The supply chain angle

This is the third AI tooling supply chain incident this month. The litellm PyPI package was backdoored on March 24, harvesting SSH keys and cloud credentials for five and a half hours. Windows Defender flagged LM Studio as a trojan (false positive, but it spooked people). And now Claude Code's source ships in its own npm package.

There's a separate, more serious issue. During the same window on March 31, malicious versions of the axios npm package (1.14.1 and 0.30.4) appeared containing a Remote Access Trojan. If you installed or updated Claude Code between 00:21 and 03:29 UTC on March 31, check your lockfiles for these axios versions or the dependency `plain-crypto-js`. If you find them, treat the machine as compromised and rotate everything.

The packaging mistake itself is mundane. As software engineer Gabriel Anhaia put it: "A single misconfigured .npmignore or files field in package.json can expose everything." This happens regularly in the JavaScript ecosystem. The only reason it's news is because of what was exposed and who it belonged to.

For anyone publishing npm packages, especially packages that interact with AI models: review your `.npmignore`, check your `files` field in `package.json`, and verify that build artifacts don't include source maps, environment files, or internal documentation. Bun generates source maps by default. If you use Bun, you need to explicitly disable them for production builds.

What happens now

Anthropic will tighten their build pipeline. The source maps are gone from npm. The code is permanently public. 41,500 forks don't disappear.

If you use Claude Code, nothing changes. The tool works the same as it did yesterday. The leaked code reveals how it works, not credentials or user data. If anything, seeing the 23-layer bash security system should increase confidence in the tool's security model, even if the release process was sloppy.

If you build local AI agents, this is an architecture document you didn't have yesterday. Claude Code generates $2.5 billion in annualized revenue. The engineering behind it is serious, and now you can read all of it.

Related guides

- Local AI Agents: What Works in 2026
- Is LM Studio Infected? How to Check Your Install
- OpenClaw Security Report: January 2026
- OpenClaw Security Report: February 2026

Get notified when we publish new guides.

Subscribe — free, no spam
