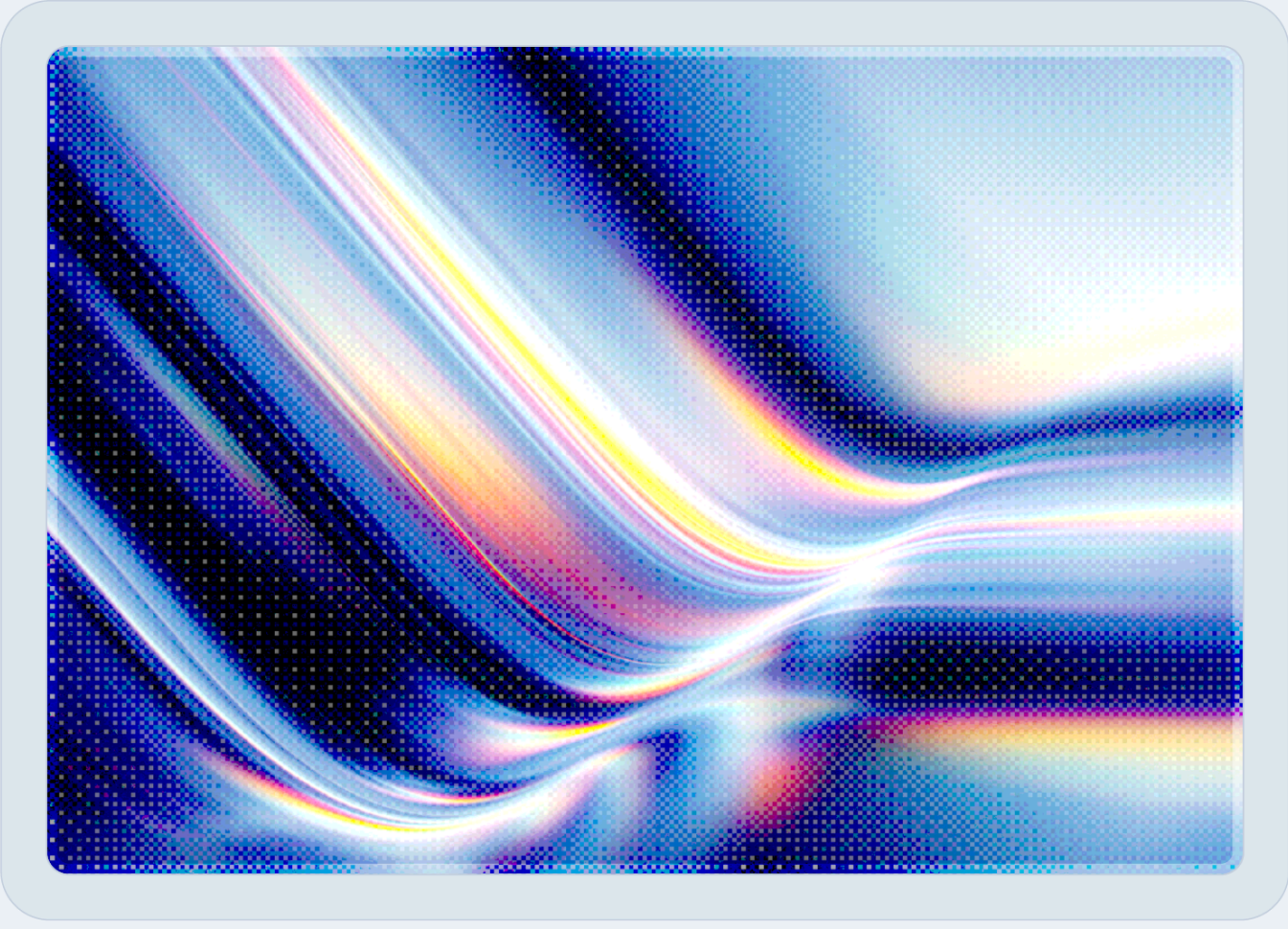
CSAI Foundation | Cloud Security Alliance

# Trojanized Claude Code Extension on Cursor's Marketplace

## Security Implications and Guidance for AI Developer Toolchains

2026-11-03

AI-assisted Rapid Research



CSAI

cloud security alliance®

**© 2026 Cloud Security Alliance. Some rights reserved.**

You may download, store, display, view, print, redistribute, and link to this document in its original, unmodified form, provided that attribution to the Cloud Security Alliance is maintained and all trademark and copyright notices remain intact.

This document may not be modified or altered. You may quote portions of the document as permitted by the Fair Use provisions of the United States Copyright Act, provided that attribution is given to the Cloud Security Alliance.

This document may be shared on professional and social media platforms in its original form with attribution.

*This document was generated with AI assistance and has not undergone official CSA review and approval processes.*

## Key Takeaways

On October 29, 2026, Cursor's extension marketplace served tampered builds of Anthropic's Claude Code extension (`anthropic.claude-code`, versions 2.1.312 and 2.1.313) to macOS and Windows users for roughly eleven hours, between approximately 09:27 and 20:35 UTC [1][6]. Cursor estimates that up to 5,000 users may have installed the malicious builds [2][5]. Anthropic, Cursor, and Microsoft have each stated that Anthropic did not build the tampered code and that it was not distributed through Anthropic's official channels, the Microsoft Visual Studio Marketplace, Open VSX, or the npm package [1][5][6]. The incident therefore sits at the distribution layer: a downstream marketplace delivered code under a trusted publisher identity, and the publisher never approved that code.

The payload was a short, unobfuscated JavaScript loader prepended to an otherwise legitimate extension bundle. It ran automatically when the extension loaded and required no user interaction. It attempted to implant an attacker SSH key, enable the local SSH service, open a persistent reverse tunnel, and send a reconnaissance beacon [1][4]. Analysts found no second-stage download and no theft of credential values. The loader did, however, collect the *names* of environment variables matching key, token, secret, and cloud-provider patterns [1][6].

Security teams should keep several points in view as they respond. Uninstalling or updating the extension does not remove the backdoor, because the loader wrote its artifacts outside the extension directory [2][6]. Exposure depends on where and when the extension was downloaded, not on the version number alone, because the 2.1.312 build on Open VSX is clean [1]. As of this writing, no public account explains how the attacker gained publishing access to Cursor's marketplace [1][6]. More broadly, the event shows that IDE forks which rehost or mirror third-party extensions add a distribution tier with its own trust boundary. Current publisher-side controls, such as account security and verified-publisher badges, do not protect that tier.

## Background

Cursor is an AI-centric code editor built as a fork of Visual Studio Code. Microsoft's licensing terms restrict use of the Visual Studio Marketplace to Microsoft products, so Cursor moved its extension source to the Open VSX registry in mid-2025 [7]. Analysis published after the incident indicates that Cursor now runs an internal marketplace service that mirrors Open VSX and also hosts Cursor's own

extensions. For each request, the service decides whether to fetch a file from Open VSX or from internal storage, and it serves the result through a Cursor-operated content delivery network [4]. To a developer, an extension installed in Cursor looks identical to one installed in VS Code, with the same publisher name, extension identifier, and version string. The bytes, however, travel through a different pipeline with different controls.

Claude Code is Anthropic's agentic coding tool, and its IDE extension is widely used among AI developer integrations. AI coding tools are designed to read, write, and execute code, so they typically run with the developer's full local privileges. Wide adoption, implicit trust, and privileged placement together make an AI coding extension an attractive vehicle for supply chain compromise. An attacker who can ship code under a well-known AI vendor's identity gains execution inside the editor process. The machines involved typically hold source code, cloud credentials, and signing keys.

The incident unfolded over less than a day. The malicious builds became available at about 2:27 a.m. Pacific time on October 29, and Cursor removed them at about 1:35 p.m. Pacific. Cursor then took its entire marketplace offline, audited access, rotated keys, and rebuilt the service before restoring it [1][6]. Early reports named only version 2.1.313, and some initially suggested that Open VSX or npm were also affected. Cursor's revised notice added 2.1.312, and later analysis confirmed that Cursor was the only distribution point [1][4][5]. Anthropic skipped to version 2.1.314 for its official releases. The command-and-control (C2) server was taken offline on October 30 with the hosting provider's cooperation [1].

# Security Analysis

The analysis below covers four questions in turn: how the payload operated, how well it worked, where the trust failure occurred, and why the extension's category raises the stakes. Together they explain why the incident matters more for its distribution path than for its loader.

## Payload Mechanics

StepSecurity's static analysis describes the tampering as a prepend. An approximately 7 KB loader was placed at the top of an otherwise unmodified 2.1.312 `extension.js` bundle [1]. OpenSourceMalware characterized the same loader as roughly 66 lines of plain JavaScript that executes before the legitimate code [4][5]. The editor's extension host loads extension modules using Node.js. The loader therefore ran with the full privileges of the user's session as soon as Cursor activated the extension, with no prompt, Claude conversation, or package-manager hook involved. The loader proceeded through six stages. Each stage was wrapped in its own error handler, so a failure in one stage would not stop the others [1]. The following table summarizes each stage's behavior and platform-specific details.

| Stage | Behavior | Platform notes |
|---|---|---|
| Reconnaissance | Collects hostname, username, OS, architecture, home and working directories, global git email, public IP, and the names of up to 40 environment variables matching KEY, TOKEN, SECRET, PASS, AWS, or GCP patterns | Values of the variables were not collected [1] [6] |
| Key implant | Appends an Ed25519 key labeled `svc-telemetry` to `~/.ssh/authorized_keys` | All platforms [1][3] |
| SSH enablement | Attempts to start sshd | macOS only where passwordless sudo exists; installs OpenSSH.Server on Windows [1] |
| Reverse tunnel | Writes an embedded private key to a temp file and retries `ssh -N -R '*:0:localhost:22'` (preferring `autossh`) to `134.209.82[.]225` every 60 seconds | Exposes the local SSH port to the attacker's server [1] |
| Persistence | Cron `@reboot` on Linux; `com.apple.sysmond` LaunchAgent on macOS; `SysMon` scheduled task or Startup-folder script on Windows | Names chosen to resemble system components [1][3] |
| Beacon | Sends recon data and per-stage status by unencrypted HTTP to port 8099 on the C2 server | Also posts to localhost; the purpose of the local post has not been publicly explained [1][4] |

The loader's code includes Linux paths, including the cron `@reboot` persistence entry and a `systemctl` call to enable sshd. However, the sources agree that Cursor served the tampered builds only to macOS and Windows clients, specifically the darwin-arm64 and win32-x64 packages [1][3][5]. The Linux logic is therefore dormant code in builds that were not delivered to Linux hosts. Linux machines enter scope only if a tampered macOS or Windows package was copied onto them by some other route.

The design is notable for what it omits. The loader did not exfiltrate credentials, did not access Claude or editor data, and did not download follow-on tooling [1]. Its purpose appears to have been establishing durable interactive access. It also gave the operator a triage list of hosts worth visiting, based on hostnames, git identities, and the presence of cloud credential variables. This inference fits the evidence, but it remains an interpretation of intent rather than an established fact.

## Effectiveness and Its Limits

Researchers differ in emphasis about how well the payload worked. Sophos and CSO Online reported Phorion's analysis, which found coding errors that caused SSH enablement to fail on most macOS and Windows hosts [2][6]. On macOS, a logic flaw meant the loader skipped enablement when it lacked administrator privileges. On Windows, an error could prevent service setup. OX Security similarly concluded that the SSH-enablement step likely failed because it requires administrative rights [3]. These defects reduce the risk that an attacker actually logged in over the tunnel, but they do not undo the other stages. Where the reconnaissance and persistence stages succeeded, the operator still received a profile of the developer's environment. The persistence mechanisms also kept trying to connect the tunnel until the C2 server was taken down. Some hosts already ran an SSH server, such as macOS machines with Remote Login enabled or Windows machines with OpenSSH Server installed for other purposes. Those hosts would have been exposed to the tunnel whether or not the enablement stage worked. The share of affected developer workstations in that state is not known.

The payload's bugs are therefore not grounds to skip investigation. For any exposed host, the questions that matter are whether a tunnel was ever established and whether follow-on activity occurred over it. Only host and network logs can answer them.

## The Distribution-Tier Trust Gap

The more significant lesson concerns where the attack occurred. Anthropic has stated that its systems and employee devices were not compromised. It has also stated that its publishing credentials were not used to reach the Microsoft marketplace, Open VSX, or npm [5][6]. Independent analysis of those distribution channels is consistent with that account [1][5]. Instead, a malicious actor "used access to Cursor's marketplace" to publish the modified build under Anthropic's extension identifier [6]. Cursor responded by rotating keys and rebuilding the entire marketplace rather than removing a single package. That suggests the compromise extended beyond one extension listing, although a full rebuild is also a prudent precaution when the access vector is not yet understood. Cursor reported that no other extensions were modified, but no third party has independently audited that claim [4][5].

The precise access vector remains undisclosed. Public hypotheses include direct compromise of Cursor's internal marketplace storage. Another hypothesis is manipulation of the routing logic that decides whether to serve a file from Open VSX or from internal storage [4]. Until Cursor publishes a root-cause analysis, defenders should treat any mirror or proxy marketplace operated by an IDE vendor as an independent point of compromise.

This matters because most enterprise extension controls are anchored to publisher identity. Allowlists name `anthropic.claude-code`, and verified-publisher badges attest that a namespace belongs to Anthropic. Neither control helps when a downstream distributor serves different bytes under the correct identity. Existing verification mechanisms do not close this gap. Microsoft has pointed to extension signature verification, now enabled by default in VS Code, as the defense against package tampering. However, that verification establishes marketplace origin rather than attesting to the publisher's own build. VS Code verifies signatures only at install time, and Cursor has said that it does not verify extension signatures [8]. In CSA's assessment, the extension ecosystem therefore lacks a widely deployed way for an IDE client to verify end to end that a received package is byte-identical to the publisher's release. OpenSourceMalware argues that any marketplace distributing software under another publisher's identity takes on an obligation to authenticate that publisher [4].

### Why AI Coding Extensions Raise the Stakes

The attacker may or may not have chosen the extension because it was an AI coding integration, but its category amplified the impact. Vendors ship AI coding tools in rapid releases, and these tools are often deployed with automatic updates enabled. Automatic updates mean that a tampered build reaches installed bases within hours. That speed is consistent with Cursor's estimate of up to 5,000 affected users in an eleven-hour window [2][5]. These tools run on workstations that commonly hold much of a developer's access, including source repositories, cloud consoles, package registry tokens, and production SSH keys. A persistent reverse SSH tunnel from such a host gives an operator a foothold from which much of that access can be reached. That risk holds even though this loader did not read credential values itself.

## Recommendations

The recommendations below are ordered by time horizon. Immediate actions address scoping and remediation on potentially exposed hosts. Short-term mitigations bring IDE extensions under change control and improve detection. Strategic considerations address the structural gap between publisher identity and package integrity that this incident exposed.

## Immediate Actions

Scoping comes first. Identify every host running Cursor. On each one, determine whether the Claude Code extension was installed or updated from Cursor's marketplace between 09:27 and 20:35 UTC on October 29, 2026 [1][2]. The version number is not enough, because the 2.1.312 build distributed through Open VSX is clean [1]. Search extension directories named `anthropic.claude-code-2.1.312-*` or `anthropic.claude-code-2.1.313-*` for `extension.js` files containing the string `134.209.82.225` or `svc-telemetry`. In this search string, the IP address is intentionally left undefanged. StepSecurity published the SHA-256 of the tampered `extension.js` (3,692,922 bytes) as `ecb58cb29078d9646ad1dcdbe51526f2d9cfea06be5fb2ab183e8ecc3ab6bf15` [1]. The 2.1.313 bundle reportedly still self-identifies as 2.1.312, which suggests that both version numbers carry the same tampered file [1]. Even so, a non-matching hash does not rule out compromise. Responders should treat the string indicators as the primary check.

OX Security and OpenSourceMalware recommend isolating any host where the malicious build loaded and preserving forensic evidence before removing artifacts [3][4]. Remediation must then address each persistence and access mechanism individually, because uninstalling the extension leaves them in place [2][6]. The following table maps each mechanism to the corresponding remediation step.

| Mechanism | Remediation |
|---|---|
| Implanted SSH key and service | Remove the `svc-telemetry` key from every user's `authorized_keys` and check whether sshd or Remote Login was enabled unexpectedly |
| Reverse tunnel and dropped files | Terminate long-running `ssh -N -R` or `autossh` processes and delete the dropped key and script files, including `~/.config/.smt`, `~/.config/.smtk`, and `$TMPDIR/.stk_*` [1] |
| Persistence entries | Remove the `com.apple.sysmond` LaunchAgent, the `SysMon` scheduled task, Startup-folder `sysmon.cmd` scripts, and any cron `@reboot` entries [1] |
| C2 communication | Block `134.209.82[.]225` on TCP 22 and 8099 and search network telemetry for historical connections [4] |

| Mechanism | Remediation |
| --- | --- |
| Credential exposure | Rotate credentials accessible from the host, prioritizing those whose environment-variable names were collected, and perform rotation from a known-clean machine [1][3][4] |

Where telemetry shows that a tunnel was established, OpenSourceMalware advises rebuilding the host rather than relying on artifact removal. It also advises reviewing SSH, shell, authentication, and network logs for follow-on activity [4]. Rebuilding is the more conservative course. It is warranted for machines with access to production systems or signing infrastructure.

## Short-Term Mitigations

Over the coming weeks, organizations should bring IDE extensions under the same change control they apply to other build-time dependencies. One option is version pinning combined with a quarantine window, which holds new extension releases for a defined period before they reach developer machines. A window of even 24 hours would likely have prevented most installations in this incident, because the malicious builds were withdrawn about eleven hours after release. That outcome assumes the fork's update path honors the policy and that held builds are re-checked before release. Enterprise policies that restrict which extensions can be installed should cover every VS Code fork in use, not only Microsoft's editor. The extension inventory should also record which marketplace each fork pulls from.

Detection engineering should also adapt. This campaign's persistence names imitated system components ( `sysmond` , `SysMon` , `svc-telemetry` ), and its beacon used unencrypted HTTP on a nonstandard port. Several endpoint rules apply well beyond this campaign: alerts on new `authorized_keys` entries, on reverse SSH forwarding initiated from user sessions, and on child processes spawned by editor extension hosts. Organizations can also reduce the value of any future foothold. Moving long-lived secrets out of developer environment variables and into short-lived, brokered credentials limits what an attacker can reach.

## Strategic Considerations

The structural problem is that publisher identity and package integrity have come apart across the extension ecosystem. Organizations that standardize on an IDE fork should ask the vendor how its marketplace authenticates publishers and whether it verifies that served packages match the upstream

publisher's artifacts. They should also ask how the vendor protects its internal storage and routing, and what its incident disclosure commitments are. These questions belong in vendor risk assessments, alongside the questions already asked of package registries and CI/CD providers.

Publishers of widely used AI developer tools also have a role. A publisher can sign its release artifacts and publish their hashes through an independent channel. Downstream marketplaces and enterprise tooling could then detect when a served package diverges from the official build, which is the exact failure observed here. Over the longer term, the industry would benefit from extending provenance frameworks such as SLSA attestations to IDE extensions. A client could then verify end to end that an installed package came from the publisher's build system, regardless of which marketplace delivered it. Finally, organizations should treat AI coding extensions and agents as non-human identities with defined privileges. That means running them with least privilege and, where feasible, separating them from the credentials a developer uses for production access.

## CSA Resource Alignment

This incident connects to several existing CSA publications and frameworks. They provide context for the attack pattern, governance arguments for the affected infrastructure, and control vocabulary for putting the recommendations above into practice.

CSA's paper *Poisoned Foundations: The AI Developer Toolchain Attack Surface* [9] examines how AI coding assistants such as Claude Code and Cursor, together with their plugin and extension ecosystems, widen the software supply chain attack surface. The Cursor incident is a concrete instance of the risk that paper describes. A widely trusted AI coding integration became the delivery mechanism for a backdoor on developer workstations.

CSA's research note *GlassWorm Takedown: Developer Supply Chain Attack Campaigns* [10] documented how adversaries shifted from attacking hardened production systems to compromising the developers who build them. One route was malicious extensions distributed through Open VSX to VS Code forks. The Cursor incident continues that pattern with a meaningful variation. GlassWorm relied on publisher accounts that attackers controlled or hijacked, whereas this attack appears to have bypassed the publisher entirely by compromising the downstream marketplace. The note's workstation-focused detection and containment practices apply directly to remediation here. These include monitoring for unexpected persistence and treating developer credentials as rotation candidates after exposure.

CSA's paper *Developer Ecosystems as Critical Infrastructure* [11] argues that package registries and extension marketplaces should be governed as shared infrastructure, because their compromise creates systemic enterprise risk. A fork-operated mirror serving thousands of developers under trusted publisher

identities fits that description. The incident illustrates the paper's argument for treating marketplace operators as critical suppliers subject to assurance requirements, rather than as passive conduits.

The AI Controls Matrix (AICM) v1.1 [12] provides the control vocabulary for putting these recommendations into practice. Its Supply Chain Management, Transparency, and Accountability domain addresses supplier assurance for marketplace operators and IDE vendors. Its Threat and Vulnerability Management domain covers detection and response for compromised developer tooling. Its Identity and Access Management domain supports governing AI coding agents as privileged non-human identities. CSA's MAESTRO framework [13] offers a layered method for teams threat-modeling AI coding agents and their extension surfaces. It helps identify where agent ecosystems, including the distribution of agent integrations, introduce trust boundaries that conventional application security models do not capture.

# References


[1] StepSecurity (Rohan Prabhu). "Trojanized Claude Code Extension Delivered Persistent SSH Backdoor to Developer Machines." StepSecurity Blog, October 30, 2026 (updated November 1, 2026).

[2] Sophos X-Ops. "Malicious Claude Code extension distributed through Cursor marketplace." Sophos Blog, October 2026.

[3] OX Security (Nir Zadok). "Compromised Claude Code extensions (versions 2.1.312 & 2.1.313) in Cursor try to install a backdoor." OX Security Blog, October 30, 2026.

[4] OpenSourceMalware. "Trojanized Claude Code Extension Plants an SSH Backdoor." OpenSourceMalware Blog, October 30, 2026.

[5] Ravie Lakshmanan. "Official Claude Code Extension on Cursor Marketplace Trojanized with Persistent SSH Backdoor." The Hacker News, October 30, 2026.

[6] Gyana Swain. "Cursor removes malicious Claude Code extension after SSH backdoor attempt." CSO Online, November 2, 2026.

[7] Cursor. "Extension Marketplace Changes – Transition to OpenVSX." Cursor Community Forum, June 25, 2025.

[8] OX Security (Nir Zadok and Moshe Siman Tov Bustan). "Can You Trust that Verified Symbol? Exploiting IDE Extensions is Easier Than it Should Be." OX Security Blog, July 1, 2025.

[9] Cloud Security Alliance AI Safety Initiative. "Poisoned Foundations: The AI Developer Toolchain Attack Surface." CSA Labs, June 30, 2026.

[10] Cloud Security Alliance AI Safety Initiative. "GlassWorm Takedown: Developer Supply Chain Attack Campaigns." CSA Labs, May 28, 2026.

[11] Cloud Security Alliance. "Developer Ecosystems as Critical Infrastructure." CSA Labs, 2026.

[12] Cloud Security Alliance. "AI Controls Matrix (AICM) v1.1." Cloud Security Alliance, 2025.

[13] Ken Huang. "Agentic AI Threat Modeling Framework: MAESTRO." Cloud Security Alliance Blog, February 6, 2025.