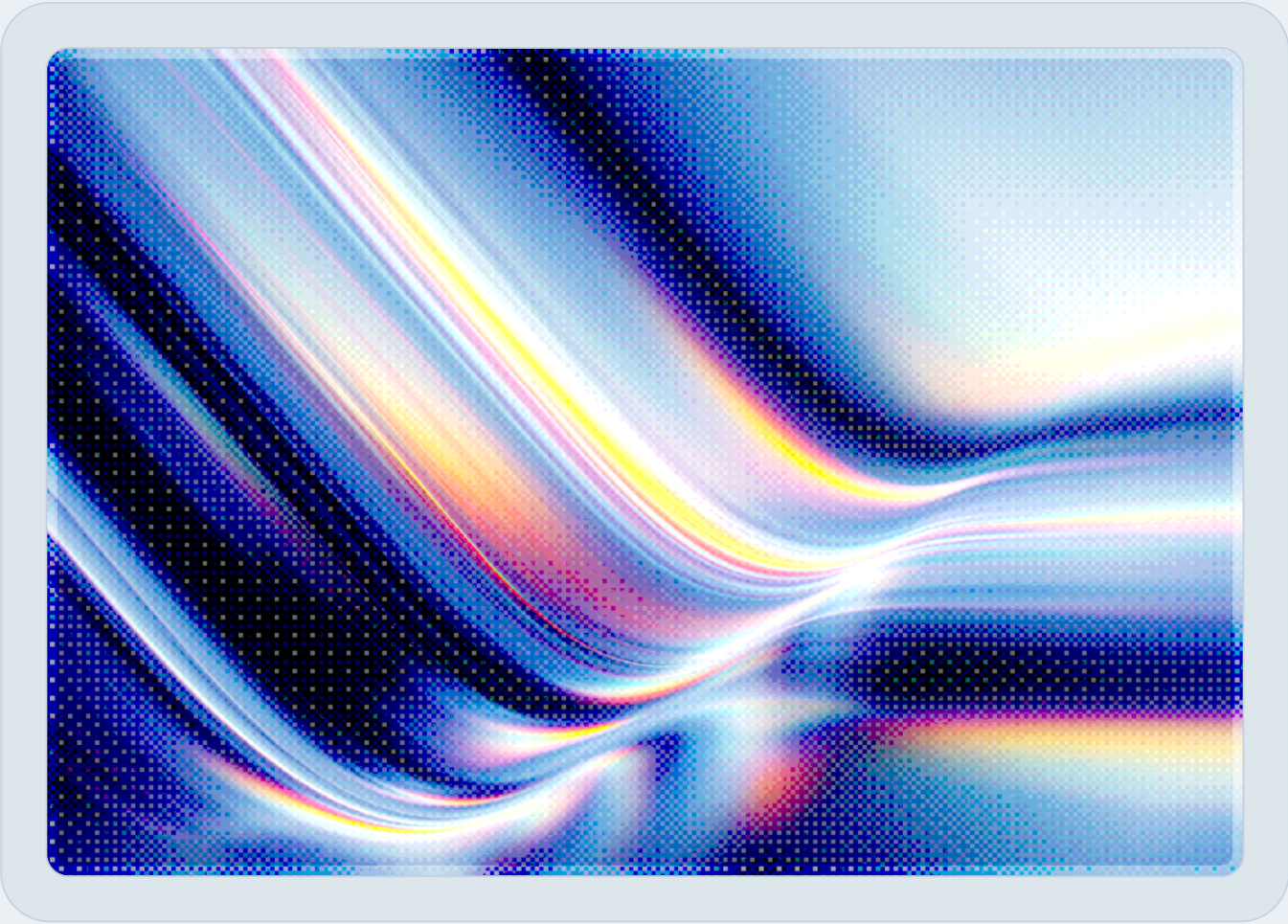
CSAI Foundation | Cloud Security Alliance

Plugin4Shell: SHA-Pinning Bypass Enables AI Coding Agent RCE

2026-09-19

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

Air Security disclosed Plugin4Shell on September 18, 2026, a zero-click remote code execution flaw that breaks the SHA-pinning integrity guarantee relied on by four major AI coding agents — Anthropic's Claude Code, OpenAI's Codex, Microsoft's GitHub Copilot, and Google's Gemini CLI — when they install plugins from a marketplace [1][2]. Each agent locks an installed plugin to a specific, previously reviewed commit hash, but none of the four verified that the code actually checked out onto disk matched that hash; an attacker who controls the plugin's source repository can exploit ambiguities in how Git resolves references to substitute malicious code while the agent continues to report that the pin is intact [2]. Because plugin auto-update is enabled by default in Claude Code and Codex, the substituted code can execute with no user interaction at all, and because the malicious code inherits the full permission scope of the developer running the agent, a successful rug-pull can expose local files, stored credentials, and any system reachable from the compromised workstation [1][3]. Air Security privately reported the flaw to all four vendors in June 2026; as of the September 18 public disclosure, Anthropic and OpenAI have shipped fixes, while Microsoft has not patched Copilot and Google has declined to patch the now-deprecated Gemini CLI [2][3]. No CVE identifier had been assigned to Plugin4Shell as of the disclosure date, three months after vendors were first notified [1].

Background

SHA pinning is a mechanism developers and AI agent marketplaces have widely adopted to make plugin installation auditable: rather than trusting a mutable branch name or version tag, an integration is locked to an immutable 40-character Git commit hash, so that whatever passed security review is exactly what later gets installed and re-installed. This pattern mirrors long-established software supply chain practice — hash-pinned dependencies, signed container digests, and reproducible builds all rest on the same assumption, that a cryptographic identifier for a specific artifact cannot be silently redirected to a different one. CSA's AI Safety Initiative has previously argued that AI coding agents have become an unaudited node in that supply chain, noting that agents now select dependencies and execute build commands with a degree of implicit trust the industry spent the past decade learning not to extend to human contributors or third-party packages [4]. Plugin4Shell extends that pattern from packages and dependencies to the coding agents' own plugin and extension ecosystems, showing that the pinning mechanism marketplaces built specifically to prevent this class of substitution can itself be defeated.

The four affected agents are among the most widely used AI coding agents on the market: Claude Code and Codex are the leading commercial agentic coding tools from Anthropic and OpenAI respectively, GitHub Copilot is Microsoft's coding assistant embedded across the GitHub and Visual Studio ecosystem, and Gemini CLI was Google's command-line coding agent prior to the company's pivot toward its successor product, Antigravity [2][3]. Each of these tools supports an extensible plugin or skill model that lets developers and third-party publishers add capabilities — linting rules, framework-specific helpers, deployment integrations — that the agent can invoke during a coding session. Because plugins run with the same privileges as the agent itself, marketplace operators built commit-hash pinning specifically to give users confidence that a reviewed plugin version could not be quietly swapped for a malicious one after installation. Air Security's research shows that confidence was misplaced across all four implementations, each vulnerable to one of two closely related exploitation techniques [2].

Security Analysis

The root cause across all four agents is the same category of defect: each agent retrieves a pinned commit hash and instructs Git to check it out, but none of them verify after the fact that the resulting working tree actually landed on that hash. Air Security identified two distinct variants of the bypass, distinguished by which Git command the agent uses to perform the checkout [2].

In the first variant, which affects Claude Code, Codex, and GitHub Copilot, the agent clones the plugin repository and runs `git checkout <SHA>`. If an attacker who controls that repository creates a branch whose name is identical to the pinned 40-character hash and sets it as the repository's default branch, Git's reference resolution can favor the branch over the raw commit object when the two are ambiguous, so the checkout silently lands on attacker-controlled branch content instead of the immutable commit the pin was meant to enforce [2]. The second variant affects Gemini CLI, which instead runs `git fetch origin <SHA>` followed by `git checkout FETCH_HEAD`. If the attacker names the repository's default branch `FETCH_HEAD`, the same ambiguity arises in reverse: the checkout resolves to the branch rather than the commit that was just fetched, discarding the legitimate pinned code entirely [2]. Both variants require the attacker to control the plugin's source repository, which Air Security notes is achievable either by publishing a plugin that is benign at the time of initial review or by later compromising an already-trusted repository, both of which the firm had previously demonstrated against other AI agent supply chain targets [2].

The practical attack chain follows a pattern security researchers have started calling a rug-pull: an attacker publishes a genuinely benign plugin, which passes marketplace review and is pinned to its initial commit; once the plugin has accumulated an installed user base, the attacker ships a routine-looking update, prompting the marketplace to re-pin installations to the new commit hash; the attacker then

creates a branch named after that new hash (or, for Gemini CLI, named `FETCH_HEAD`) and points it at malicious code; the next time an installed agent's background auto-update process re-checks the plugin, it executes the substituted code with zero user interaction and no error surfaced to the user [2]. In CSA's assessment, this sequencing is significant: it means the compromise is not limited to careless users who install unreviewed plugins, but specifically targets users who did everything the security model asked of them, installing only marketplace-reviewed, hash-pinned plugins.

The table below summarizes patch status by agent as of the September 18, 2026 disclosure.

Agent	Vendor	Checkout mechanism exploited	Patch status
Claude Code	Anthropic	`git checkout <SHA>` / branch-name collision	Fixed in version 2.1.179 [1][2]
Codex	OpenAI	`git checkout <SHA>` / branch-name collision	Fixed in version 0.146.0 [1][2]
GitHub Copilot	Microsoft	`git checkout <SHA>` / branch-name collision	No client fix at disclosure [1][3]
Gemini CLI	Google	`git fetch` + `git checkout FETCH_HEAD`	Not fixed; product deprecated in favor of Antigravity [1][3]

In CSA's assessment, marketplace operators have limited ability to close this gap on their own. Restricting plugin hosting to platforms that reject SHA-shaped branch names would blunt the first variant, but doing so effectively limits plugins to GitHub-hosted repositories, excluding self-hosted Git servers and platforms such as Bitbucket that the affected agents officially support, and it does nothing to address Gemini CLI's `FETCH_HEAD` variant [2][3]. Air Security's own assessment is that the fix has to be enforced by the agent itself, immediately after checkout, by asserting that the resolved `HEAD` — not the ref that was originally requested — matches the pinned SHA exactly, aborting installation if it does not [2].

Recommendations

Immediate Actions. Security teams should inventory every AI coding agent deployed across developer workstations and CI environments and confirm each is running a patched version — Claude Code 2.1.179 or later and Codex 0.146.0 or later close the vulnerability, while GitHub Copilot and Gemini CLI currently have no vendor fix to verify against [1][2]. Where an agent cannot yet be confirmed patched, disabling automatic plugin updates removes the zero-click element of the attack, forcing any future plugin substitution to at least require a manual re-installation step that a vigilant user or endpoint control could catch.

Short-Term Mitigations. Organizations that operate or curate internal plugin marketplaces for these agents should treat every installed plugin as a software supply chain dependency subject to the same scrutiny CSA has recommended for AI-generated code and its dependencies more broadly, including maintaining a record of exactly which commit each installed plugin currently resolves to and re-verifying that resolution independently of the agent's own self-reported pin status [4]. Teams that cannot immediately retire GitHub Copilot or Gemini CLI plugin usage should, at minimum, restrict plugin sourcing to repositories hosted on platforms that reject commit-hash-shaped branch names, recognizing this narrows but does not eliminate exposure and does not address the Gemini CLI variant at all [2].

Strategic Considerations. The durable fix is architectural rather than procedural: any agent, marketplace, or package manager that advertises commit-hash pinning as an integrity control needs to assert the resolved working-tree state after checkout, not merely the reference it requested, and vendors should be pressed to publish this verification behavior explicitly rather than leaving it as an undocumented implementation detail. More broadly, Plugin4Shell is evidence that the ambient trust and privilege AI coding agents extend to their own extensions deserves the same least-privilege sandboxing and identity controls CSA has recommended for MCP servers and other agent tooling, so that even a successful pin bypass is contained to a narrow, monitored blast radius rather than the full permission set of the developer running the agent [5].

CSA Resource Alignment

Plugin4Shell is the latest entry in a pattern CSA's AI Safety Initiative has been tracking since mid-2026: AI coding agents functioning as unaudited nodes in the software supply chain, making trust decisions about code provenance with less scrutiny than the industry now applies to human-authored dependencies. CSA's research note "AI Coding Agents: An Unaudited Supply Chain Node" [4]

documented this pattern through incidents including hallucinated-package slopsquatting and prompt-injection-driven CI/CD compromise, and recommended extending existing SBOM, provenance, and hash-verification practices to cover agent-installed code; Plugin4Shell shows that even where a hash-verification control was explicitly built for this purpose, an unverified post-checkout state let it fail across four independent implementations without surfacing any error to the user.

CSA's "Sandboxing Agentic AI: Least-Privilege Patterns for MCP and Coding Agents" [5] is directly applicable to containing the consequences of a Plugin4Shell-style compromise even where a vendor patch has not yet shipped. That note's central finding — that a recurring architecture problem in 2026 agentic AI incidents is an agent or its extensions holding more ambient authority than a given task requires — describes exactly the condition that turns a plugin rug-pull into a zero-click RCE with the full reach of the developer's credentials and file system, and its recommended isolation stack (process, container, or microVM sandboxing matched to the trust level of installed extensions) would substantially reduce Plugin4Shell's blast radius regardless of which agent or plugin source triggered it.

Plugin4Shell also belongs alongside CSA's coverage of "GuardFall: Shell Injection Bypass Defeats AI Coding Agent Guardrails" [6], which found that ten of eleven popular open-source coding agents could be tricked into executing arbitrary shell commands despite command-safety guardrails, because the guardrails inspected commands before the shell's own expansion and resolution logic ran. Plugin4Shell exhibits the identical structural pattern one layer up the stack: the pinning control inspects and records the requested reference before Git's own reference-resolution logic runs, and never checks the two actually agree. A related concern is documented in CSA's "Miasma and IronWorm: Self-Replicating Worms Targeting AI Credentials" [7], which examined how self-replicating worms exploit AI coding agents' plugin and configuration trust boundaries to propagate across the same agent ecosystem — Claude Code, GitHub Copilot, and Gemini CLI among them — and to harvest credentials once a foothold is established; Plugin4Shell provides one more concrete mechanism by which that foothold can be obtained without any user interaction. Together, these cases argue for the same governance response, which CSA's AI Controls Matrix (AICM) v1.1 formalizes across its Application and Interface Security and AI Supply Chain domains: verification of an artifact's actual, resolved state, rather than the identifier used to request it, must be treated as the control, not an implementation detail left to each vendor's discretion [8].

References

[1] The Hacker News. "Plugin4Shell Lets Repository Owners Swap Pinned Plugin Code Across Four AI Coding Agents." The Hacker News, September 18, 2026.

[2] Air Security. "Plugin4Shell: Zero-Click RCE Vulnerability Found in Top 4 Most Popular Coding Agents." Air Security, September 18, 2026.

[3] Help Net Security. "Zero-Click RCE Vulnerability Hit Four Major AI Coding Agents, Two Remain Unpatched." Help Net Security, September 18, 2026.

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

[5] Cloud Security Alliance AI Safety Initiative. "Sandboxing Agentic AI: Least-Privilege Patterns for MCP and Coding Agents." Cloud Security Alliance, May 30, 2026.

[6] Cloud Security Alliance AI Safety Initiative. "GuardFall: Shell Injection Bypass Defeats AI Coding Agent Guardrails." Cloud Security Alliance, July 1, 2026.

[7] Cloud Security Alliance AI Safety Initiative. "Miasma and IronWorm: Self-Replicating Worms Targeting AI Credentials." Cloud Security Alliance, June 9, 2026.

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

