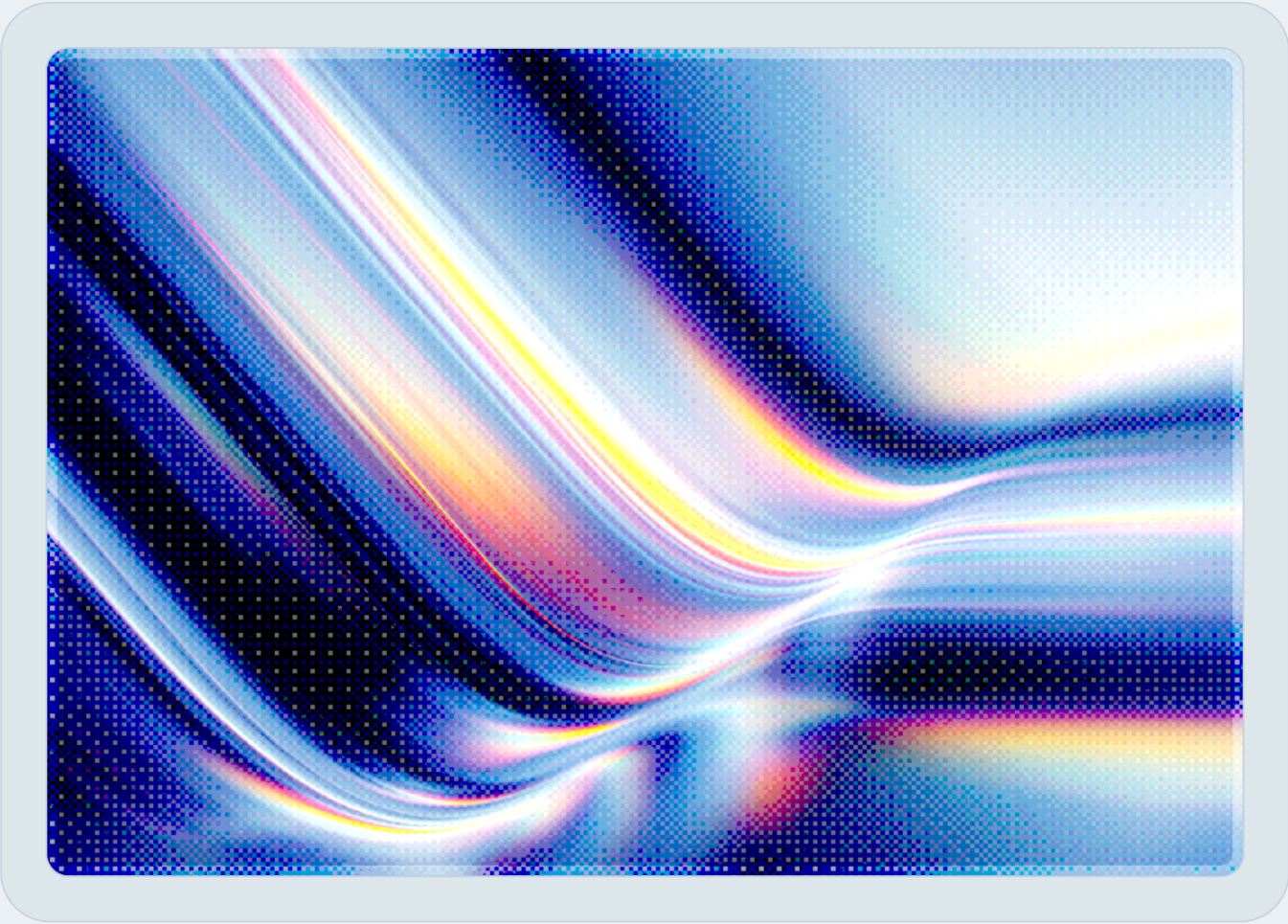
CSAI Foundation | Cloud Security Alliance

Three AI Coding Agents, One GitHub Issue: CI/CD Secrets Exposed

2026-08-08

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

Security researcher Elad Meged of Novee Security disclosed, at Black Hat USA 2026 on August 5, that a GitHub issue opened by an account with no repository privileges was enough to reach CI runner secrets in the vendors' own repositories for Claude Code, Gemini CLI, and OpenAI Codex [1][2]. In Anthropic's Claude Code, a command validator that strips single-quoted text before running its checks let a payload smuggled inside a `git push --receive-pack` flag reach the runner untouched, and after two further patch-and-bypass rounds the final variant, tracked as CVE-2026-54316, abused Claude Code's pre-approved access to Hugging Face to exfiltrate an API key one character at a time using public download counters as a covert channel [1][3]. In Google's Gemini CLI, automatic workspace trust in headless mode combined with an allowlist that was checked at registration but never enforced at execution let an attacker read the parent process's environment through Linux's `/proc` filesystem, a flaw tracked as CVE-2026-12537 with a maximum CVSS v4 score of 10.0 [1][4]. OpenAI's Codex received no CVE because the company characterized the underlying multi-pass architecture, in which one agent invocation can write an instruction file that a later invocation loads and trusts, as the sandbox working as designed rather than as a patchable defect [3]. All three vendors have shipped fixes or workflow changes, and CISA's exploit-tracking data showed no confirmed active exploitation of the Claude Code or Gemini CLI CVEs as of the disclosure date, but the researchers found comparable default configurations running unmodified across well over a hundred other public repositories, indicating the underlying design pattern is common rather than vendor-specific [1][3].

Background

Over the past two years, AI coding agents have moved from developer-facing autocomplete tools into autonomous participants in the software delivery pipeline itself. Repositories increasingly wire these agents into GitHub Actions and similar CI/CD systems so that opening an issue, filing a pull request, or leaving a comment can automatically trigger an agent to read the request, reason about it, and take action, up to and including writing code, running shell commands, and merging changes. That automation is valuable precisely because it removes a human from the loop for routine triage and maintenance work, but removing the human also removes the checkpoint that would normally catch a request from an untrusted or anonymous source before it reaches a privileged execution environment. A GitHub issue is, by design, something anyone with an internet connection and no prior relationship to the

project can open, which means any automation that treats issue content as an instruction to an agent running inside the CI environment is implicitly trusting arbitrary strangers with a foothold inside that environment.

Novee Security's research, presented at Black Hat USA 2026, tested exactly this scenario against the three leading commercial vendors, Anthropic, Google, and OpenAI, evaluated against their own repositories running each vendor's own default, unmodified workflow configurations [2][3]. These are not obscure third-party integrations bolted onto AI coding agents by less careful downstream users; they are the reference configurations that the vendors themselves ship, and which other organizations plausibly copied or adapted, a pattern the researchers later confirmed by finding comparable configurations replicated across well over a hundred public repositories [1][3]. Across all three tools, the researchers found paths to remote code execution, credential theft, unauthorized repository modification, and in Codex's case, instructions that persisted across otherwise-unrelated agent runs [3]. The findings extend a pattern CSA's AI Safety Initiative has tracked since the disclosure of Clinejection in early 2026, in which prompt injection delivered through a single malicious GitHub issue title enabled CI/CD cache poisoning and the theft of npm credentials later used to publish a trojanized package [11], and more recently GhostCommit [10] and GuardFall [8], both of which showed that AI coding agents' safeguards frequently fail not because the underlying model is unsafe but because the surrounding harness, the code that mediates between the model's decisions and real system actions, was not built to withstand adversarial input.

Security Analysis

Claude Code: three rounds of patch and bypass

Anthropic's Claude Code Action, the GitHub Action that lets Claude respond to issues and pull requests inside a repository's CI pipeline, went through three distinct rounds of vulnerability and patch during Novee's engagement. The first round centered on a command-injection flaw in how the action validated bash commands before execution. Claude Code's validator approved or rejected commands using prefix matching against an allowlist, but the researchers found that the validator stripped single-quoted text out of a command string before running its 23 validation checks, a step that is correct behavior for how bash itself interprets quoting but that also meant a malicious payload placed inside single quotes, such as within the value passed to git's `--receive-pack` flag, reached the runner without ever being inspected [1][3]. That allowed an attacker to trigger remote code execution on the runner and read arbitrary files, including the environment variables holding the `GITHUB_TOKEN` and the project's Anthropic API key [2].

Anthropic's first patch closed that specific bypass, but Novee's second round found that the fix relied on hardcoded lists classifying certain commands as inherently "read-only" and therefore exempt from the same scrutiny applied to write operations. Commands such as `tac`, `rev`, `fold`, `expand`, and `unexpand`, all of which can read and reformat file contents without technically writing anything, were on that read-only list and so bypassed path validation entirely, letting an attacker use `tac` combined with empty-quote obfuscation to read `/etc/environ`, a file the validator had specifically tried to protect [3]. The third round, and the one that ultimately received a CVE, exploited Claude Code's pre-approved outbound access to Hugging Face through its WebFetch tool. Because Hugging Face was on the tool's allowed-domains list, an attacker who had otherwise lost the ability to exfiltrate data directly could instruct Claude to create up to 64 model repositories, one representing each possible character of an API key, and then monitor the public download counters on those repositories as an encoding channel: incrementing the counter for the repository matching the next character of the secret leaked that character one increment at a time, a technique the researchers described as turning a public metric into an exfiltration oracle [3]. This final finding was assigned CVE-2026-54316 and is rated inconsistently across sources: Anthropic scored it Moderate under CVSS v4 at 6.0, while the National Vulnerability Database's independent assessment under the older CVSS v3.1 scale rated it 9.1, a divergence that likely reflects both a scoring-methodology difference and differing views on how much precondition (getting untrusted content into a Claude Code context in the first place) should discount the score [1][6]. Anthropic's published security guidance for the action acknowledges the underlying tension directly, warning that its content sanitizer can be bypassed and that new bypass techniques may emerge, and recommending that operators review the raw content of untrusted input from external contributors before letting Claude process it rather than relying on the sanitizer as a sole control [7]. The vulnerability affected Claude Code Action versions from 0.2.54 through 2.1.162 and was fixed in version 2.1.163 [1][3].

Gemini CLI: a perfect CVSS score from two collapsed assumptions

Google's Gemini CLI and the associated `run-gemini-cli` GitHub Action carried a vulnerability that researchers scored at the maximum possible CVSS v4 value of 10.0, tracked as CVE-2026-12537 and documented in advisory GHSA-wpqr-6v78-jr5g [4]. The flaw rested on two assumptions that each held in isolation but collapsed together. First, Gemini CLI automatically extended workspace trust to any folder it operated on when running in headless, non-interactive mode, the mode CI/CD pipelines use by default, which meant it would load configuration files, including a `.env` file inside a repository's `.gemini/` directory, without the explicit authorization step that interactive use requires [4]. Second, the tool's `coreTools` allowlist, meant to restrict which shell commands the agent could execute using prefix matching such as permitting only `run_shell_command(echo)`, was parsed and matched during tool registration but never actually enforced at the point of execution, so the restriction existed in

name only [3][4]. Combined, an attacker could plant a malicious `.env` file that Gemini CLI would trust and load automatically in a CI context, then use the effectively unrestricted shell tool to execute arbitrary commands on the runner before any sandbox isolation had even initialized.

A separate design gap compounded the impact. Gemini CLI did sanitize the environment passed to child processes it spawned, stripping sensitive variables as intended, but the parent process's own environment remained readable through Linux's `/proc/$PPID/environ` interface because the parent and child shared a process ID namespace. That gap exposed the `GITHUB_TOKEN`, the `GEMINI_API_KEY`, and any other credentials injected into the CI job's environment, unfiltered, to any command the agent executed [3]. The combination gave an unprivileged, unauthenticated attacker a path from a crafted repository file to full shell access and credential theft on the CI host, sufficient in principle to publish backdoored packages to the roughly two million monthly installs that depend on the Gemini CLI package alone [3]. The vulnerability was independently discovered and reported through Google's Vulnerability Rewards Program by both Meged and Dan Lisichkin of Pillar Security, and Google fixed it in `@google/gemini-cli` 0.39.1 (and 0.40.0-preview.3) and in `run-gemini-cli` 0.1.22, describing the change as a breaking change to how non-interactive headless environments handle folder trust rather than a narrow patch [1][3][4].

Codex: a design decision, not a bug

OpenAI's Codex presented a materially different case because the company declined to treat the underlying issue as a vulnerability at all. Novee's researchers demonstrated that when a CI workflow runs two Codex passes within a single job sharing one checked-out workspace, the first pass can write to `AGENTS.md`, an instruction file Codex loads from disk and treats as authoritative context on every invocation. `AGENTS.md` is not protected the way `.git/`, `.agents/`, or `.codex/` directories are, so if the first pass's output fails schema validation and triggers a second pass, that second pass loads whatever instructions now sit in `AGENTS.md`, including attacker-controlled content injected into the model's system prompt by the first, compromised pass [3]. OpenAI's position was that this reflects documented, intended sandbox behavior rather than a defect, meaning organizations running comparable two-pass or multi-pass architectures with a shared workspace should not expect a vendor patch to close the gap and instead need to redesign their own workflow to separate passes into distinct jobs with clean checkouts, or enforce a read-only sandbox for any pass that consumes another pass's output. OpenAI reportedly hardened the `openai/codex` repository's own workflows within three days of the report, but because no CVE was assigned, other organizations running similar patterns have no advisory to alert them that the design itself, not a specific software version, is the exposure [3].

Table 1 summarizes the three findings.

Agent	Root cause	Identifier / severity	Fixed in
Claude Code Action	Quote-stripping validator bypass; hardcoded "read-only" command exemptions; Hugging Face download-counter exfiltration channel	CVE-2026-54316 (CVSS v4 6.0 per Anthropic; CVSS v3.1 9.1 per NVD)	2.1.163
Gemini CLI / run-gemini-cli	Automatic headless workspace trust loading `.env`; tool allowlist unenforced at runtime; parent-process `/proc` environment leak	CVE-2026-12537, GHSA-wpqr-6v78-jr5g (CVSS v4 10.0)	gemini-cli 0.39.1 / 0.40.0-preview.3; run-gemini-cli 0.1.22
OpenAI Codex	Multi-pass architecture lets one pass's output become a later pass's trusted system-prompt context via unprotected `AGENTS.md`	No CVE assigned; classified by OpenAI as intended sandbox behavior	Workflow-level hardening in `openai/codex`; no version fix

A pattern, not three isolated bugs

What makes these findings significant collectively is less any individual bypass technique and more what they say about where the real attack surface in AI coding agents sits. As Meged put it in describing the research, the harness is the code between the model and the real world, and each of these three vulnerabilities lived in that harness rather than in the underlying language model's judgment [3]. Adversa AI's GuardFall research on the shell-injection class of flaws, which CSA's AI Safety Initiative has also tracked, reached the same structural conclusion from a different angle, finding that ten of eleven tested open-source coding agents used command guards that inspected commands as plain strings while the shell that actually executed those commands rewrote them first [8], a mismatch between what the guard sees and what the system runs that incremental denylist tuning has so far failed to close. The Novee findings and GuardFall together point toward pattern-based, string-level validation as a structurally inadequate control for agents that hand commands to a real shell, regardless of vendor. The scale of exposure reinforces the point: Novee found well over a hundred public repositories running configurations functionally identical to the vulnerable defaults in the vendors' own repositories, and within days of this disclosure Google separately removed three of its own Agent Development Kit

workflows, `issue-analyze.yml`, `issue-fix.yml`, and `pr-analyze.yml`, after researchers showed that a privilege-escalation gap let an attacker manipulate content posted by a trusted bot account to unlock a second, credentialed workflow, the same class of trusted-identity-without-provenance-checking failure playing out again in a related but distinct codebase [5]. Taken together, these incidents support the reframing CSA's AI Safety Initiative has argued for elsewhere [9]: AI coding agents function as an unaudited node in the software supply chain, making decisions and executing commands with a degree of implicit trust that the industry has spent the past decade learning not to extend to human contributors without review.

Recommendations

Immediate Actions

Organizations running Claude Code Action, Gemini CLI, or `run-gemini-cli` in CI/CD pipelines should upgrade immediately to Claude Code 2.1.163 or later, Gemini CLI 0.39.1 (or 0.40.0-preview.3) or later, and `run-gemini-cli` 0.1.22 or later [1]. Because Codex received no vendor patch, teams running multi-pass Codex workflows that share a single checkout across passes should treat this as an architecture defect requiring their own remediation: split passes into separate jobs with independent, clean checkouts, or restrict any pass that consumes another pass's output to a read-only sandbox that cannot write to `AGENTS.md` or equivalent instruction files. Every organization using any of the three tools should also audit which of their own workflows can be triggered by an outside contributor opening an issue, filing a pull request, or leaving a comment, since that trigger condition was the common entry point across all three vendors' vulnerable configurations, and any repository secrets, `GITHUB_TOKEN` scopes, or model API keys accessible to those workflows should be rotated as a precaution if the workflow ran in a vulnerable configuration before patching.

Short-Term Mitigations

Security teams should treat GitHub issue content, pull request descriptions, comments, and any repository file an agent loads automatically, including convention files like `AGENTS.md` or `CLAUDE.md` and configuration files like `.env`, as untrusted input regardless of the agent vendor's default trust settings. Long-lived personal access tokens granted to CI workflows should be replaced with short-lived, narrowly scoped credentials so that a successful exfiltration yields a token of limited value and duration rather than standing access. Where an agent's tool permissions include outbound network access to third-party services, as Claude Code's WebFetch access to Hugging Face illustrated,

that allowlist deserves the same scrutiny as a firewall rule, since an "approved domain" for legitimate use can double as an exfiltration channel once an attacker controls what the agent sends to it. Auto-execute or "YOLO" modes that skip human approval for agent-issued commands should be disabled in any workflow reachable by unauthenticated or low-privilege external input, and workflows should log and monitor for anomalous access patterns, such as processes reading `/proc/*/environ`, that would indicate an agent's execution environment is being probed for credentials.

Strategic Considerations

The recurrence of harness-level failures across three independently engineered products from three different vendors, and the reappearance of a related failure mode in Google's own ADK repository within days, suggests that pattern-based command validation and prefix-matched tool allowlists are not adequate long-term controls for agents that execute real shell commands on privileged infrastructure. Organizations should push vendors of AI coding agents to disclose whether command validation operates on the raw string a model produces or on the fully normalized, post-expansion command the shell will actually execute, since only the latter can reliably catch the class of bypass Novee and Adversa AI's GuardFall research both documented. CI/CD governance programs should begin treating AI coding agents as privileged automated identities subject to the same Zero Trust principles, least-privilege scoping, and audit logging applied to service accounts and human administrators, rather than as developer productivity tools sitting outside the security perimeter. Procurement and vendor-risk processes for AI coding tools should incorporate specific questions about harness design, sandboxing, and credential isolation architecture, since this research demonstrated that base-model safety training did not prevent any of these three outcomes; the exploitable gap in each of the three cases examined here sat in the surrounding integration code.

CSA Resource Alignment

This research connects most directly to Adversa AI's GuardFall research, which CSA's AI Safety Initiative has also tracked, and which documented the identical structural failure, command guards validating a string that the shell subsequently rewrites before execution, across ten of eleven tested open-source coding agents [8]. The Novee findings against Claude Code, Gemini CLI, and Codex extend that same root-cause pattern to three of the most widely deployed commercial and vendor-maintained agents, reinforcing GuardFall's conclusion that shell-aware, post-expansion command validation is a more durable approach than incremental denylist entries. CSA's AI Agent Prompt Injection: The New CI/CD Supply Chain Threat provides the broader supply-chain framing this incident confirms: agents that execute commands and access credentials in CI/CD pipelines are privileged actors in the software

delivery chain, and this disclosure shows that framing applying to the reference configurations of the very vendors building these tools, not just to downstream adopters [9]. The GhostCommit disclosure is relevant for a related reason: it demonstrated that Claude Code's model-level refusals held where other agents' did not, yet this research shows that harness-level flaws beneath those refusals still permitted compromise, underscoring that model behavior and harness security must be evaluated as separate, independent controls rather than treated as substitutes for one another [10]. Finally, CSA's AI Controls Matrix (AICM) v1.1 offers the control structure organizations should use to formalize these fixes [12]: its Application and Interface Security and Threat and Vulnerability Management domains map directly onto the credential scoping, tool-permission auditing, and secure-configuration practices this note recommends, giving security teams a framework against which to assess and document their AI coding agent deployments rather than relying on ad hoc, incident-driven remediation.

References

[1] The Hacker News. "Claude Code and Gemini CLI Flaws Let a GitHub Issue Reach CI Workflow Secrets." The Hacker News, August 2026.

[2] Hackread. "Black Hat USA 2026: One GitHub Issue Could Compromise Major AI Coding Workflows." Hackread, August 2026.

[3] Novee Security. "If You Run These Automations, You're Exposed Too: Critical Flaws in Anthropic, Google, and OpenAI's Coding Agents." Novee Security Blog, August 2026.

[4] GitHub Advisory Database. "Gemini CLI: Remote Code Execution via Workspace Trust and Tool Allowlisting Bypasses (GHSA-wpqr-6v78-jr5g)." GitHub, 2026.

[5] The Hacker News. "Google Deletes 3 ADK AI Workflows After Malicious GitHub Issue Could Trigger Privileged Agent." The Hacker News, August 2026.

[6] National Vulnerability Database. "CVE-2026-54316 Detail." NIST, 2026.

[7] Anthropic. "Claude Code Action: Security." GitHub, 2026.

[8] Adversa AI. "GuardFall: Open-Source AI Coding Agents Shell Injection Vulnerability." Adversa AI Blog, June 2026.

[9] Cloud Security Alliance. "AI Agent Prompt Injection: The New CI/CD Supply Chain Threat." Cloud Security Alliance, 2026.

[10] Cybersecurity News. "New GhostCommit Attack Hides Prompt Injection Inside Images to Bypass AI Code Reviewers." Cybersecurity News, July 2026.

[11] Cloud Security Alliance. "Clinejection: Prompt Injection Enables CI/CD Cache Poisoning." Cloud Security Alliance, 2026.

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

