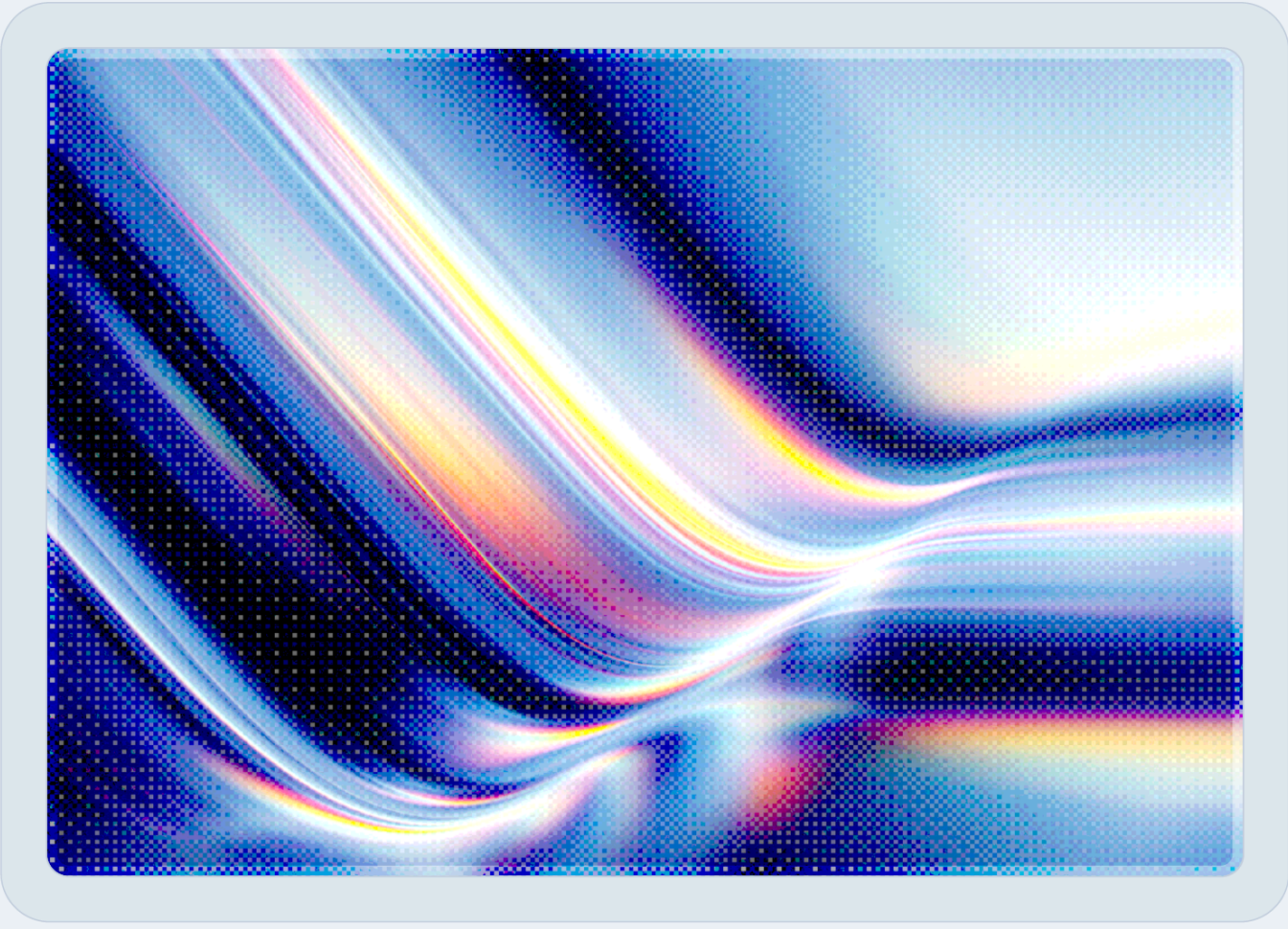
CSAI Foundation | Cloud Security Alliance

AI vs. AI: An Autonomous Agent Exploits a Snowflake Flaw

How a Red-Teaming Agent Beat AI Code Review to a CI/CD Vulnerability

2026-08-25

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

An autonomous red-teaming agent built by Wiz independently discovered and exploited a script-injection flaw in a Snowflake GitHub Actions workflow, exfiltrating an internal Jira access token less than a week after the vulnerable code merged [1]. The flaw sat in a workflow that interpolated an untrusted GitHub issue title directly into a shell command. Per Wiz, GitHub Advanced Security's static analysis scanned the vulnerable workflow file without flagging the injection, and Copilot Autofix — listed as a co-author on the merged pull request — is described by Wiz as having reviewed the change and missed it as well, a characterization GitHub disputes, stating Copilot never reviewed the specific vulnerable lines [1][2][3]. When its first exploitation attempt failed on a bash syntax error, the agent, called Red Agent, autonomously diagnosed the failure and rewrote its payload to succeed, illustrating — on Wiz's account — an adaptive exploitation capability that static, signature-based scanning typically does not replicate [1]. A subsequent dispute between Wiz and GitHub over whether Copilot Autofix had actually authored the vulnerable lines — it had not; a human Snowflake engineer had, in a separate commit — underscores a harder problem than the vulnerability itself: as more AI systems touch a single pull request, attributing responsibility for a defect between human authors, AI co-authors, and AI reviewers is becoming genuinely difficult [3][4]. Snowflake investigated the incident, found no evidence that any other party exploited the flaw during its exposure window, and rotated the affected credential within a day of disclosure [1]. The episode is best read not as an indictment of any single tool but as an early, concrete data point consistent with a trend CSA has been tracking since June 2026: in at least this and one prior documented case, a commercial autonomous offensive agent found a real production vulnerability before the AI-assisted review process in place caught it [5].

Background

Snowflake maintains `snowflake-connector-net`, an open-source .NET driver, on GitHub. Like many engineering organizations, Snowflake uses GitHub Actions workflows to automate routine maintenance tasks, including a workflow that creates a corresponding Jira ticket whenever someone opens a GitHub issue against the repository. On June 18, 2026, a pull request modified this workflow's handling of the issue title, and Wiz's account of the incident states that GitHub Copilot Autofix was listed as a co-author on the merged commit, which merges all contributors to a squashed pull request into a single attribution line [1]. The modified code took the raw text of the issue title and interpolated it into a

shell command using `echo`, then piped the result through further shell processing. Because GitHub issues can be opened by any user with a GitHub account, and because the workflow's `if:` condition — intended to gate execution — checked a pull-request-only field that evaluates to null (and therefore effectively true) on issue-triggered events, the workflow would run its shell command in response to any issue title, from any account, with no repository privileges required [1].

Five days later, on June 23, 2026, Wiz's autonomous security research tool, Red Agent, identified the pattern while scanning Snowflake's public GitHub organization for exploitable weaknesses [1]. Red Agent crafted an issue title containing a single quote designed to break out of the `echo` string and inject an arbitrary command. Per Wiz's account, its first attempt produced a bash syntax error rather than command execution; rather than stopping, the agent analyzed the error output, adjusted its payload to properly close and reopen the quoted string, and resubmitted [1]. The revised payload executed successfully inside the GitHub Actions runner, retrieved the `JIRA_API_TOKEN`, `JIRA_USER_EMAIL`, and `JIRA_BASE_URL` environment variables, base64-encoded them, and exfiltrated them via a DNS-style callback to an out-of-band listener the agent controlled. The stolen token authenticated as a Snowflake service account with read access to internal engineering, security, and bug-bounty Jira projects. Wiz reported the finding to Snowflake through HackerOne the same day it was discovered; Snowflake patched the workflow and rotated the exposed credential within 24 hours, and the two companies coordinated public disclosure for August 17–19, 2026 [1].

The incident's second act concerned attribution rather than the vulnerability itself. Wiz's initial blog post framed the finding as an AI-versus-AI story: an autonomous offensive agent exploiting a flaw that an AI code-review co-author had missed. GitHub disputed the framing, stating that "a human wrote the contributions that led to the vulnerability, and that Copilot Autofix neither reviewed nor contributed to them" [3]. The underlying issue was mechanical: when a pull request is squashed into a single commit, every participant who touched any file in that PR is listed as a co-author on the final commit message, regardless of which specific lines they wrote. GitHub's analysis found that Copilot Autofix had worked on a different file within the same pull request, while the vulnerable shell-interpolation code traced to a separate commit from August 25, 2025, attributed to a named Snowflake engineer [3][4]. Wiz updated its post to acknowledge that "it's unclear whether the code-change was AI-assisted," while maintaining that Copilot's review of the merged PR had not flagged the injection risk regardless of who wrote it [2] [3]. Wiz co-founder and CTO Ami Luttwak summarized the broader lesson: "co-authors of the PR is not enough" to establish who is actually responsible for a given line of code, a problem he said is becoming more acute "in a world where multiple agents run on every PR, scan it, and update it" [2][4].

Security Analysis

Two separate failures compounded to make this incident possible, and neither is unique to Snowflake or to Copilot. The first is a familiar class of defect: untrusted user input — a GitHub issue title, fully controllable by any account with no repository access — flowed into a shell command without sanitization, a pattern security teams have flagged in GitHub Actions workflows for years. What made this instance notable is that the flaw survived GitHub Advanced Security's static analysis — which, per Wiz, scanned the vulnerable workflow file directly — and, on Wiz's account (though GitHub disputes that Copilot Autofix specifically reviewed the vulnerable lines), a Copilot Autofix review as well, for five days in a public repository, before an external actor found it. That gap matters because GitHub Actions injection is a well-documented, well-understood vulnerability class; a review process — human, AI, or hybrid — that misses a known pattern in a public repository provides limited assurance against novel ones.

The second failure is organizational rather than technical: the confusion over whether Copilot wrote the vulnerable code was resolvable only through a manual, after-the-fact forensic review of commit-level file attribution, because the readily visible signal — the co-author line on a squashed merge commit — does not distinguish "touched this pull request" from "wrote this vulnerability." As AI coding assistants, AI code-review tools, and AI red-teaming agents all become routine participants in the same development pipeline, the metadata that development platforms currently expose is not granular enough to answer a question that matters immediately after an incident: who, or what, introduced the flaw, and did any automated reviewer have a genuine opportunity to catch it. Wiz's own correction illustrates the risk of getting this wrong quickly and publicly: an initial framing that overstated AI authorship drew a public rebuttal from GitHub, generating a second news cycle largely about the dispute itself rather than the underlying vulnerability.

The exploitation side of the incident reinforces a pattern CSA has tracked since Wiz Red Agent's April 2026 public preview: adaptive, multi-step reasoning by an offensive agent finds and chains context-dependent flaws — authorization gaps, injection points, workflow logic errors — that static, signature-based scanning has missed in documented cases, and does so at a pace that compresses the window between a vulnerability shipping and its discovery [5]. In this case, that discovery window was five days, and the discovering party was a security vendor conducting internet-wide reconnaissance rather than an internal audit. Had it been an adversarial actor rather than Wiz, the outcome — read access to internal engineering and security Jira projects via a supply-chain-adjacent CI/CD credential — would have looked identical up to the point of disclosure. The broader implication is a preview of a landscape in which organizations should expect adversarial agents to operate at comparable speed against CI/CD workflows in public repositories, even though this incident involved only a defensive one; an organization's own review pipeline, whether AI-assisted or not, is competing against machine-speed reconnaissance rather than against a periodic external audit.

Recommendations

Immediate Actions

Security and engineering teams should audit GitHub Actions (and equivalent CI/CD) workflows that trigger on `issues`, `issue_comment`, or `pull_request_target` events for any pattern that interpolates event-derived text — issue titles, comment bodies, branch names — directly into a shell command rather than passing it through an environment variable with proper quoting. Teams should also verify that any conditional gate intended to restrict a workflow to trusted triggers actually evaluates correctly for every event type the workflow listens on; a null-safe check written for pull request events, as in this case, can silently evaluate to true on issue events. Any secret accessible to a CI/CD runner that has ever executed unvalidated user input should be treated as potentially exposed and rotated.

Short-Term Mitigations

Organizations relying on AI-assisted code review — whether GitHub Copilot Autofix or comparable tools — should treat a clean AI review as one input among several rather than a sufficient control, particularly for well-known vulnerability classes like injection in CI/CD trigger handling, where a dedicated static analysis rule or linter is likely to be more reliable than general-purpose AI review. Teams should also establish, before an incident forces the question, how they will determine authorship and review responsibility when a vulnerability surfaces in code touched by multiple human and AI contributors within a single pull request; commit co-author metadata alone, as this incident demonstrated, is not sufficient evidence. Piloting or expanding continuous, agent-based red teaming against public-facing repositories and CI/CD configurations — rather than relying solely on periodic manual review — will narrow the window between a vulnerable workflow merging and its discovery.

Strategic Considerations

This incident is a data point in a broader shift toward machine-speed offense and defense operating on the same software delivery pipeline simultaneously. Security leaders should assume that any public repository is subject to continuous, automated reconnaissance by both defensive researchers and potential adversaries, and that the relevant benchmark for review-process effectiveness is no longer "did a human or AI reviewer eventually catch this" but "how many days of exposure existed before either side found it." Governance processes for AI-assisted development also need an attribution model that

survives contact with an incident: distinguishing an AI system that authored code from one that reviewed it, and documenting review scope precisely enough to state, after the fact, whether a given tool had a genuine opportunity to catch a specific defect.

CSA Resource Alignment

This incident sits directly at the intersection of two lines of CSA research published earlier in 2026. CSA's "Three AI Coding Agents, One GitHub Issue: CI/CD Secrets Exposed" [6] documented an almost identical exploit pattern across Claude Code, Gemini CLI, and OpenAI Codex: a GitHub issue opened by an account with no repository privileges reaching CI runner secrets, in each case because a validation or trust boundary failed to account for how AI coding agents process untrusted event data inside automated pipelines. The Snowflake case extends that finding to a workflow with no AI coding agent in the execution path at all — it required only an AI code-review participant that failed to catch a familiar injection pattern — indicating that the underlying risk is broader than any single vendor's agent architecture and applies wherever GitHub event data reaches a shell command unsanitized. CSA's "Autonomous AI Red Teams: Security Implications and Guidance" [5] profiled Wiz Red Agent's April 2026 public preview and its rapid accumulation of findings — approximately 17,000 unique findings across roughly 1,000 customer environments in its first month — characterizing exactly the adaptive, multi-step exploitation behavior — reasoning over failures and adjusting payloads dynamically — that this incident's technical account describes; that note's recommendation to treat continuous, agent-based validation as a near-term necessity rather than a future aspiration is consistent with a five-day exposure window closing only because a vendor's own red-teaming agent found the flaw first. CSA's "The Vibe Coding Governance Gap" [7] further observed that none of the major AI security frameworks in active use provide dedicated guidance for attributing responsibility when AI systems participate in code authorship and review alongside humans, a gap this incident's Copilot-attribution dispute illustrates concretely even outside the citizen-developer context that note primarily addresses.

Organizations seeking a structured control baseline for the CI/CD and AI-code-review practices implicated here should consult CSA's AI Controls Matrix v1.1 [8], which spans application security, secure software development lifecycle, and identity and access management domains relevant to both the injection vulnerability and the exposed CI/CD credential. For organizations building or evaluating agent-based offensive or defensive tooling of the kind Wiz deployed here, CSA's MAESTRO agentic AI threat modeling framework [9] provides a layered reference architecture for reasoning about how an autonomous agent's planning, tool-use, and execution layers each introduce distinct attack surface and defensive requirements.

References

[1] Wiz Research. "Red Agent Exploits Snowflake Vuln Missed by GitHub Copilot." Wiz Blog, August 17, 2026.

[2] Bell, Lily Hay. "Snowflake Flaw Slips Past AI Checks, Gets Exploited by Another AI." CSO Online, August 19, 2026.

[3] "GitHub Disputes Wiz's Claim That Copilot Autofix Wrote a Snowflake Flaw." The Next Web, August 2026.

[4] "Wiz CTO Speaks Out Amid Confusion Over Snowflake-GitHub Copilot Flaw." IT Pro, August 2026.

[5] Cloud Security Alliance. "Autonomous AI Red Teams: Security Implications and Guidance." CSA AI Safety Initiative, 2026.

[6] Cloud Security Alliance. "Three AI Coding Agents, One GitHub Issue: CI/CD Secrets Exposed." CSA AI Safety Initiative, August 8, 2026.

[7] Cloud Security Alliance. "The Vibe Coding Governance Gap." CSA AI Safety Initiative, June 2, 2026.

[8] Cloud Security Alliance. "AI Controls Matrix v1.1." CSA, 2026.

[9] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." CSA Blog, February 6, 2025.