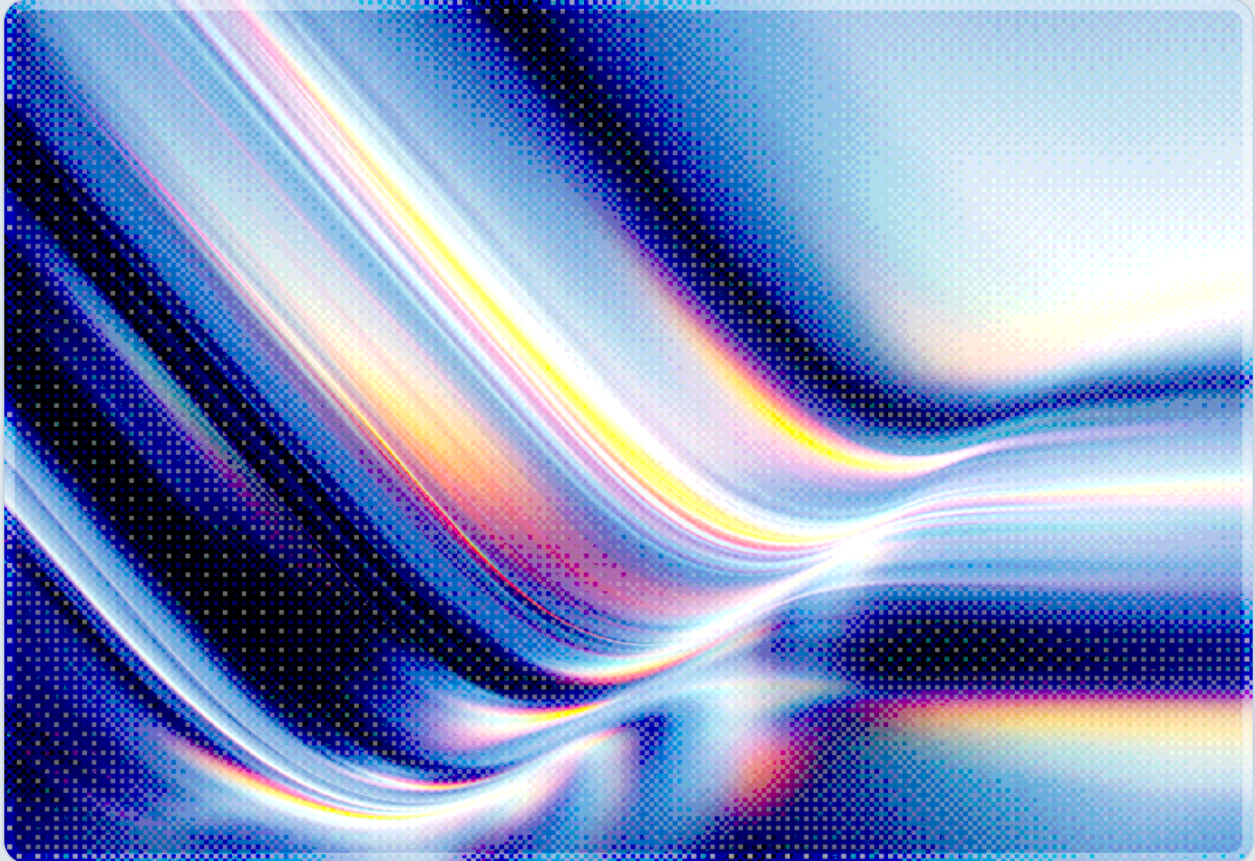


Gheist: Stolen GitHub Tokens Turn Actions Logs into Exfiltration Channels

Security Implications and Guidance

2026-10-29

 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

Gheist is an ongoing campaign that Wiz Research has tracked since September 15, 2026. Attackers use stolen GitHub tokens to plant a workflow named `sync-check` on new `ci-sync-` branches. The workflow prints the job's environment and its full `secrets` context to the Actions log in base32 encoding. The attacker downloads the log through the GitHub API, and the run is deleted seconds later [1][2]. Wiz reports more than 13,500 planted branches across more than 850 compromised accounts, ranging from personal projects to private repositories in enterprise organizations, and states that the activity was still running and changing as of October 24, 2026 [1]. Most of these attempts did not execute: GitHub's approval hold stopped the public-repository runs, and Wiz's own title says the campaign "stumbles." The confirmed impact is concentrated in private repositories, where Wiz observed completed runs, but Wiz's analysis does not say how many organizations had secrets exposed [1][2]. The campaign takes its name from the `gheist` user-agent string its tooling sends with GitHub API requests [1][2].

The technique needs no malware on victim infrastructure and no attacker-controlled server. Stolen secrets leave through GitHub's own API, so there is no attacker domain or outbound connection for network controls to block [1]. Base32 encoding defeats log masking because GitHub redacts secrets only when they appear verbatim, and GitHub's own documentation states that redaction "is not guaranteed" once a value is transformed [1][5]. In practice, any credential that can push workflow files to a repository (a classic token with the `workflow` scope, or a fine-grained token or app with write access to workflows) can expose every repository- and organization-level secret available to that repository's workflows, other than secrets held behind protected environments.

On July 28, 2026, GitHub introduced a control that holds potentially malicious workflow runs until a collaborator approves them. That control stopped every Gheist run Wiz examined in public repositories [1][3]. It does not currently extend to private repositories on github.com or to GitHub Enterprise Server [1][3]. Because enterprise application code is usually held in private or self-hosted repositories, those repositories are likely to hold most production credentials, and that is the scope the hold does not cover. Wiz observed runs completing in private repositories and found AWS CloudTrail records of organizations' own workflows assuming AWS roles on attacker branches [1].

Organizations should hunt for the published indicators, rotate every secret available to any affected repository (deleting the branch or run does not undo the exposure), revoke the pushing tokens, and treat any self-hosted runner that executed a `ci-sync` branch as compromised [1][2]. Over the longer term, three controls substantially reduce what a stolen token can reach, particularly production cloud credentials: required reviewers on deployment environments, OIDC trust policies scoped to protected branches or environments, and GitHub Apps or short-lived fine-grained tokens in place of long-lived personal access tokens [1][4]. Repository-level secrets and source code remain exposed to any token with sufficient scope.

Background

GitHub Actions workflows run with access to the secrets configured at the repository, organization, and environment levels, subject to whatever restrictions the owner has applied. GitHub's documentation describes automatic redaction of secret values printed to workflow logs, and it also cautions that "because there are multiple ways a secret value can be transformed, this redaction is not guaranteed" [5]. That caveat is what makes Gheist's exfiltration step work. A workflow that encodes the secrets context before printing it produces log output that no longer matches the stored secret strings, so masking does not apply and the values appear in recoverable form [1].

The campaign also exploits a trust boundary in how workflows are triggered. A workflow defined on a branch of the repository runs as first-party code, so a push trigger on that branch executes with the repository's secrets in reach. Code review and branch protection on the default branch do not prevent the creation of a new branch carrying a new workflow file. Gheist is one of several campaigns in 2026 to exploit that boundary at scale. The Megalodon campaign of May 18, 2026 pushed 5,718 malicious commits to 5,561 repositories in roughly six hours and harvested CI secrets through injected workflows [8][10]. On October 8, 2026, a revived GhostAction campaign used two compromised maintainer accounts to push credential-stealing workflows into 345 repositories [9]. Wiz reports that 57 accounts compromised in Gheist's first phase were also Megalodon victims in May. It describes Gheist as distinct from GhostAction because the two use different workflow files and GhostAction exfiltrates to a separate IP address [1].

The stolen tokens appear to come from earlier compromises rather than from a new initial-access technique. Wiz names infostealer logs and tokens exposed in prior supply chain attacks as sources, including the March 2026 compromise of the Trivy vulnerability scanner [1][2]. In that incident, the TeamPCP group force-pushed malicious code to Trivy's GitHub Action release tags and harvested CI/CD

credentials from pipelines that consumed them [7][13]. Gheist therefore illustrates a compounding pattern: credentials stolen in one supply chain event are used later to harvest a wider set of credentials, which may in turn seed the next event.

GitHub responded to this class of attack on July 28, 2026 by holding workflow runs it flags as potentially malicious until a repository collaborator with write access approves them through an authenticated web session. The changelog entry states that the protection "currently applies to public repositories on github.com only" and that "GitHub Enterprise Server doesn't add this protection at this time" [3].

Security Analysis

Attack Mechanics

Wiz's analysis reconstructs a consistent operating pattern [1]. For each repository a stolen token can reach, the tool creates a branch named `ci-sync-` followed by six random characters. It commits a single unsigned file, `.github/workflows/sync-check.yml`, with the message `ci: sync config`, authored as the token's owner. The push triggers the workflow, which dumps the environment and `toJSON(secrets)` through `base32`. The workflow declares `permissions: {}`, which restricts the automatically issued `GITHUB_TOKEN` but does not restrict access to the secrets context. Because GitHub's masking matches only verbatim values, the encoded output prints in full. The operator retrieves the log through the API and deletes the run within seconds.

The tooling has changed during the campaign. Wiz distinguishes two variants observed in parallel, "Tool A" and "Tool B." Within hours of its first run on September 28, Tool B added a six-minute timeout, a job matrix that enumerates the repository's deployment environments, and per-repository payload changes [1]. The environment matrix matters because environment-scoped secrets are often production credentials, and the matrix reaches them whenever an environment lacks a protection rule [1]. On October 19 the tool added a `LAF=====` separator, which Wiz assesses was likely added for the operator's log parser, and began pushing a second `sync-host` job aimed at self-hosted runners [1]. That job searches the runner host for 31 token formats, including Anthropic, Hugging Face, Groq, and NVIDIA API keys, private keys, `.env` files, Terraform state, and shell history. It also probes cloud instance metadata, the Docker socket, runner credential files, and Kubernetes service account tokens. Wiz describes the job as buggy, and no successful run has been reported [1][2].

In enterprise organizations, Wiz observed the tool cycling through tokens belonging to several users and automation accounts within minutes, cloning private repositories before planting the workflow [1]. Hours later, cloud credentials from the same organization were used to enumerate AWS Secrets Manager. Wiz's analysis does not say where those cloud credentials came from [2], so the link between the Actions dump and the cloud activity should be treated as a likely sequence rather than an established one.

Why the Technique Is Effective

Several properties of the technique make it difficult for conventional controls to catch. Each one removes a signal or a checkpoint that defenders would normally rely on, and together they leave little for network or code-review controls to observe. The table below summarizes the main ones.

Property	Effect on defenders
Exfiltration through the GitHub API	No attacker domain, IP, or outbound connection exists for egress filtering or DNS controls to block [1].
Base32 encoding of the secrets context	Defeats verbatim log masking, a display safeguard that GitHub states is not guaranteed for transformed values [1][5].
Run deletion seconds after completion	Removes the log from the repository UI. Evidence survives mainly in audit logs and cloud-side records, and only where those are collected [1].
New branch rather than default-branch change	Avoids pull request review and default-branch protection. The commit appears under a legitimate user's identity [1].
Push trigger in first-party context	The workflow runs with access to repository and organization secrets without any fork or pull request approval gate [1].
Hold limited to public github.com repositories	The platform's main automated control does not cover private or Enterprise Server repositories, which are likely to hold most production credentials [1][3].

The final row deserves emphasis. GitHub's approval hold was effective where it applied: every public-repository Gheist run that Wiz checked was still awaiting approval, and Wiz concluded that "the control worked, and the malicious payloads never ran" [1]. That result is encouraging evidence that platform-level

gating of suspicious workflows can neutralize unadapted attacks of this kind. However, the hold covers only runs that GitHub flags [2], and its current scope does not cover the repositories most likely to hold production credentials.

Detection Opportunities and Their Limits

The campaign leaves distinctive artifacts. Wiz publishes the user agent `gheist`, the branch pattern `^ci-sync-[0-9a-z]{6}$`, the commit message `ci: sync config` adding only `sync-check.yml`, the payload strings `toJSON(secrets) } }' | base32 -w 120` and `====gdmp-pivot:`, the self-hosted runner files `.gdmp_pat`, `.gdmp_patg`, and `.gdmp_sweep`, and four workflow blob hashes [1]. Environment-matrix variants differ per repository and have no fixed hash [2]. Because the operator has changed the tooling several times in six weeks, defenders should expect these literal indicators to lose value over time. Behavioral signals are likely to last longer, such as a workflow run that is deleted within a minute of completing, or a newly created branch whose only change is a workflow file.

Cloud-side telemetry provides an independent detection path that does not depend on GitHub audit log retention. Wiz recommends searching cloud audit logs for role assumptions or token exchanges whose OIDC subject includes a `ci-sync-` branch [1]. One caveat applies: by default the OIDC subject contains the branch name only when the job does not use a deployment environment [2]. Jobs bound to an environment present an environment-based subject, as GitHub's OIDC documentation illustrates with `repo:octo-org/octo-repo:environment:prod` [6]. A search on branch names alone may therefore miss environment-scoped role assumptions, and defenders should correlate on repository, workflow, and time window as well.

Relevance to AI Workloads

Gheist is not an AI-specific attack, but its target list reflects how much value AI provider credentials now carry in development pipelines. The self-hosted runner job explicitly hunts for Anthropic, Hugging Face, Groq, and NVIDIA keys [1], and the October GhostAction activity targeted Anthropic, OpenAI, and OpenRouter keys [9]. Depending on the provider and key scope, stolen model API keys may allow an attacker to consume paid inference at the victim's expense, to reach fine-tuned or private models, or to query logs and stored prompts that may contain sensitive data. Organizations building AI applications should therefore inventory model provider keys among their CI/CD secrets and rotate them with the same urgency as cloud credentials.

Recommendations

Immediate Actions

Organizations with GitHub repositories, especially private repositories on github.com and GitHub Enterprise Server instances, should first establish whether they are affected. That means searching all repositories for branches matching `^ci-sync-[0-9a-z]{6}$` and for commits adding `.github/workflows/sync-check.yml` with the message `ci: sync config`. It also means reviewing GitHub audit logs for the `gheist` user agent and for workflow runs on `ci-sync-*` branches that were deleted within a minute of completion [1]. Cloud audit logs should be searched in parallel for OIDC role assumptions tied to those branches, bearing in mind the environment-subject caveat described above [1][2].

For any affected repository, the response should assume full secret exposure. Wiz advises rotating every repository, organization, and environment secret the repository could use, because deleting the branch or the run does not reverse the disclosure [1][2]. Model provider keys deserve the same priority as cloud credentials. The tokens that pushed the malicious commits should be revoked, and the devices of the token owners should be examined for infostealer infections, since infostealer logs are a reported token source [1]. Any self-hosted runner that executed a `ci-sync` branch, together with its cloud instance role, should be treated as compromised and rebuilt rather than cleaned [1]. Finally, administrators should instruct collaborators not to approve held `sync-check` runs and not to execute code from `ci-sync` branches [1].

Short-Term Mitigations

CSA considers the most effective near-term control to be making production secrets unreachable from an arbitrary branch. GitHub's documentation states that jobs referencing an environment "can only access these secrets after any configured rules (for example, required reviewers) pass" [4]. Moving production credentials from repository- and organization-level secrets into environments that have required reviewers and deployment branch restrictions directly blunts Gheist's environment-matrix variant, which succeeds where environments lack protection [1][4]. Repository-level secrets that remain should be limited to low-impact, easily rotated values.

Token permissions offer a second, inexpensive lever. Because planting a workflow file requires the `workflow` scope on classic tokens or write access to workflows on fine-grained tokens and GitHub Apps, organizations should audit personal access tokens and OAuth and GitHub App grants for that scope or permission and remove it wherever it is not required.

Cloud trust relationships should be tightened in the same pass. OIDC trust policies should accept tokens only from protected branches or specific environments rather than any ref in a repository, so that a workflow running on an attacker-created branch cannot assume a privileged role [1][6]. Organizations should also stream GitHub audit logs to a SIEM with retention independent of GitHub, because the attacker's run deletion otherwise removes the most direct evidence [1]. Detections for rapid create-then-delete run sequences and for new branches whose only change is a workflow file are likely to outlast the campaign's current indicators.

Where the organization's GitHub plan supports them, repository rulesets that restrict branch creation patterns or changes to `.github/workflows/` paths can add friction to this technique. Teams should confirm that such rules are enforced for automation accounts as well as human users, since Wiz observed automation-account tokens being used in the campaign [1].

Strategic Considerations

The underlying problem is that long-lived, broadly scoped personal access tokens function as standing keys: a token with workflow write access can expose every unprotected secret its owner's repositories can reach. Wiz recommends GitHub Apps or short-lived fine-grained tokens in place of personal access tokens [1]. Organizations should treat this as a token governance program covering both human-owned tokens and non-human identities, rather than as a one-time cleanup. Such a program would inventory tokens and automation accounts, enforce expiry and least-privilege scopes, and monitor for anomalous cross-repository use, such as a single token touching dozens of repositories within minutes. CSA's analysis of the finalized NIST and CISA guidance on identity token theft [14] offers further direction for that work.

Self-hosted runners warrant particular attention because they convert a repository-level compromise into a host- and cloud-level one. Wiz recommends ephemeral runner hosts without Docker socket access or unrestricted cloud metadata access [1]. Runner instance roles should carry only the permissions the build requires, and runners serving sensitive repositories should not be shared with less-trusted ones.

Finally, organizations should press their platform providers on control coverage. GitHub's approval hold stopped every public-repository Gheist run Wiz examined [1]. Extending equivalent protection to private repositories and GitHub Enterprise Server would remove this campaign's main advantage, though a detection-based hold is only as durable as GitHub's ability to flag adapted payloads. Until that happens, enterprises should assume that private repositories have no automatic platform-level backstop against malicious workflows pushed with valid credentials.

CSA Resource Alignment

CSA has tracked the wave of 2026 GitHub Actions secret-theft campaigns of which Gheist is a part. CSA's research note [Megalodon: Mass CI/CD Pipeline Poisoning via GitHub Actions](#) [10] analyzed the May 2026 campaign that shares 57 victim accounts with Gheist [1]. That note's central finding, that missing protection on workflow paths and over-privileged runner environments enabled poisoning at scale, applies directly here. Gheist shows the same weakness exploited with valid user credentials instead of throwaway accounts, which makes identity-side controls more important than contributor-permission controls alone. CSA's note [TeamPCP: CI/CD Security Tool Supply Chain Compromise](#) [13] covers the Trivy compromise that Wiz identifies as one source of Gheist's tokens.

The companion note [GhostAction Returns: Maintainer Takeover Steals Workflow Secrets](#) [9] covers the October 2026 campaign that Wiz distinguishes from Gheist [1]. Read together, the two notes show two concurrent approaches to the same weakness: GhostAction exfiltrates to an attacker-controlled server, while Gheist exfiltrates through the platform itself. Gheist's approach reduces the value of egress monitoring and increases the value of audit-log and cloud-trust telemetry.

Gheist is, in large part, a failure of token and secrets governance, spanning both human-owned tokens and non-human identities. CSA's [The State of Non-Human Identity Security](#) [12] survey report examines the gaps in how organizations inventory, scope, and rotate tokens, service accounts, and API keys. Those are the same artifacts Gheist both consumes as its entry point and harvests as its payload. The AI Controls Matrix (AICM) v1.1 [11] provides the control baseline for operationalizing these recommendations. Its Identity and Access Management domain covers least privilege and credential lifecycle for tokens and automation accounts. Its Cryptography, Encryption and Key Management domain covers secret rotation. Its Application and Interface Security and Threat and Vulnerability Management domains cover pipeline hardening and detection. Because AICM extends these controls to AI-specific assets, it also frames model provider API keys as credentials subject to the same governance as cloud keys.

References

- [1] McCarthy, Rami and Hila Ramati (Wiz Research). "[Gheist: Attempted Scaled Abuse of Stolen GitHub Tokens Stumbles](#)." Wiz Blog, October 25, 2026.
- [2] Khandelwal, Swati. "[Gheist Uses Stolen GitHub Tokens to Steal Private Repository Secrets via Actions Logs](#)." The Hacker News, October 26, 2026.
- [3] GitHub. "[GitHub Actions holds potentially malicious workflows for approval](#)." GitHub Changelog, July 28, 2026.
- [4] GitHub. "[Managing environments for deployment](#)." GitHub Docs, accessed October 29, 2026.
- [5] GitHub. "[Secrets](#)." GitHub Docs, accessed October 29, 2026.
- [6] GitHub. "[OpenID Connect](#)." GitHub Docs, accessed October 29, 2026.
- [7] Wiz Research. "[Trivy Compromised: Everything You Need to Know about the Latest Supply Chain Attack](#)." Wiz Blog, March 20, 2026.
- [8] Prabhu, Rohan (StepSecurity). "[Megalodon: Mass GitHub Actions Secret Exfiltration Across 5,500+ Public Repositories](#)." StepSecurity Blog, May 22, 2026.
- [9] Cloud Security Alliance. "[GhostAction Returns: Maintainer Takeover Steals Workflow Secrets](#)." CSA Lab Space, October 11, 2026.
- [10] Cloud Security Alliance. "[Megalodon: Mass CI/CD Pipeline Poisoning via GitHub Actions](#)." CSA Lab Space, May 24, 2026.
- [11] Cloud Security Alliance. "[AI Controls Matrix \(AICM\) v1.1](#)." Cloud Security Alliance, 2025.
- [12] Cloud Security Alliance. "[The State of Non-Human Identity Security](#)." Cloud Security Alliance, September 11, 2024.
- [13] Cloud Security Alliance. "[TeamPCP: CI/CD Security Tool Supply Chain Compromise](#)." CSA Lab Space, March 25, 2026.
- [14] Cloud Security Alliance. "[NIST and CISA Finalize Guidance on Identity Token Theft](#)." CSA Lab Space, September 23, 2026.