15 Hours of Open-Sourced Hell: Lessons Learned from tj-actions/changed-files

March 17, 2025 | Liam Troper and Liad Cohen

Table of Contents

Written by Liad Cohen and Liam Troper

On March 14, 2025, at precisely 6:57 PM (GMT+2), the open-source ecosystem experienced a significant security breach that is still sending shockwaves through the developer community: The popular GitHub Action ***tj-actions/changed-files***—trusted and integrated by over 23,000 repositories—was compromised when attackers injected malicious code into its codebase.

For approximately 15 hours, this widely used action silently betrayed its users, exposing sensitive information, including environment variables, secrets, and more, to CI/CD logs. The breach (assigned **CVE-2025-30066**) affected all repositories referencing the action, regardless of whether they had been pinned to specific versions.

GitHub eventually shut down the compromised action, and it took another 8 hours before a safe version was restored. During this timeframe and while it lasted, all repositories referencing this GitHub action and triggering it were unknowingly also executing the malicious code.

The Compromise

The attackers managed to commit a malicious change

The attackers managed to commit a malicious change (0e58ed8671d6b60d0890c21b07f8835ace038e67). This change contained an obfuscated segment of malicious code, which targeted the `Runner.Worker` process to extract secrets (CI/CD and env. variables in the context), which were then printed in the workflow logs.

These logs are readable to anyone in public repositories – making them most severely impacted. Private repositories are still affected – but logs with sensitive information will be available to whoever has access to view workflow logs.

While no external exfiltration to attacker-controlled servers was observed, the exposure in public logs poses significant risks. The malicious commit was injected into **all versions** of tj-actions/changed-files, up to 45.0.7. As attackers modified existing version tags and retroactively updated them so that older versions would execute the malicious code as well. Users employing commit-SHA-pinning remained unaffected unless they updated to a compromised hash during the few hours of the exposure period.

The compromised commits have been reverted, and the malicious scripts have been removed. However, secrets that were leaked in logs remain at risk, especially in public repositories.

A quick search on GitHub unveils the sheer number of exposed repositories that ran CI/CD jobs, executing malicious code and printing out secrets.

A Victim Repository Example

Here's an example of a relatively popular GitHub repository (2.3k+ stars) and its secrets being publicly exposed in workflow logs in a CI/CD job during the exposure period:

https://github.com/chromebrew/chromebrew/blob/76a796ec6b706f987968ff498cd8b455159ff55a/.github/workflows/Build.yml#L76 (https://github.com/chromebrew/chromebrew/blob/76a796ec6b706f987968ff498cd8b455159ff55a/.github/workflows/Build.yml#L76)

The action is pinned to a compromised version:

Looking at the workflow logs, on the specific vulnerable job:

We can see a dumped base64 double-encoded in the logs (line 54), which, after double-decoding, reveals some GitHub tokens:

“github_token”:{“value”:”ghs_***”,”isSecret”:true}

“system.github.token”:{“value”:”ghs_***”,”isSecret”:true}

(Masked for obvious concerns).
A **ghs_** is a GitHub temporal token, valid for a maximum of 24 hours. The above tokens are no longer active.

How to Check For Affected Public Repositories?

GitHub search is our buddy. More than 11.1K public repositories are referencing a compromised version of the action;

Search as follows:

path:/^\.github\/workflows\// AND (path:.yml OR path:.yaml) “uses: tj-actions/changed-files@v”

How to Check If Your Private Repositories Were Affected?

Similar to the public repository search, look for ***tj-actions/changed-files*** references without commit-SHA pinning, then examine CI/CD jobs executed during the exposure period. Once identified, carefully review the logs and check for base64 double-encoded strings that may indicate compromise.

How to Safely Reference External Actions: Immutable SHA Pinning

The best practice for preventing the attack vector is twofold: First, reference an internal action, which will be specific to your use case, and make sure to restrict access to modify it.

Second, when you must reference an external GitHub action – a vetting process is required as well

as pinning it to a commit SHA.

Here's an example of a **secure** reference to the GitHub action:

https://github.com/DataDog/dd-trace-py/blob/e08d99c22a4237cc3d04e0a863e5b8a889646220/.github/workflows/codeowners.yml#L19 (https://github.com/DataDog/dd-trace-py/blob/e08d99c22a4237cc3d04e0a863e5b8a889646220/.github/workflows/codeowners.yml#L19)

The above action is pinned to a specific commit SHA – not to a release-version or branch name, thus **immutable.** Any new (possibly malicious) commits introduced, to any branch or version, will **not** affect the action behavior or execution.

Recommended Actions for Affected Users

Follow these steps and best practices to ensure your secrets stay concealed:

1. **Identify Usage**: Search your codebase for references of ***tj-actions/changed-files*** to determine if your workflows are affected.
2. **Cease Usage**: Immediately discontinue using the compromised action. Consider alternatives or implement in-house solutions for file-change detection.
3. **Rotate Secrets**: Assume that secrets used in workflows involving the compromised action have been exposed. Rotate these secrets promptly.
4. **Audit Logs**: Review your GitHub Actions logs for any unauthorized access or anomalies during the period of compromise.

A prevention security best practice:

Pin your GitHub Actions to specific commit SHAs to avoid such supply chain attacks slipping through your SDLC. This guarantees that your workflows execute a fixed, immutable version of the action's code, safeguarding against third-party repository risks such as unauthorized modifications, malicious code merged, a maintainer going rogue, a maintainer account takeover, or a repository takeover.

You should have a vetting process in place when referencing or fetching external content, e.g., an external GitHub action. In these cases, governance is key—have policies in place when a developer wants to introduce new references to external content, such as a GitHub action, and have it pass an approval process. You should always track what your developers are doing to enforce the policy.

How can OX Security help?

1. OX has a specific policy in place for this attack vector. The OX Security research team introduced this policy over a year ago, as we anticipated such attacks would happen. **OX alerts on any external GitHub action referenced without a commit-SHA pinning.**
2. OX provided its customers with the means to **prevent** the attack for those who have used the policy in their pipeline workflows in block mode.

3. You can set a no-code workflow automation in OX to take specific actions once a detection is found. This includes **blocking the pipeline**, sending alerts and messages, opening Jira tickets, sharing information through webhooks, and more. Developers will automatically receive all the context they need to pinpoint and mitigate the problem with as little friction as possible.

Want to learn more about how GitHub will be used to introduce backdoors across your Software Development Lifecycle without you noticing?

Join us on-stage at RSA Conference, on Monday, 28 April (2:20 PM – 3:10 PM PDT) at San Francisco (Moscone Center) for an in-depth session:
“In GitHub We Trust: 10 Ways You Could Get Pwned” by Eyal Paz & Liad Cohen, with eye-opening demos and mitigation strategies.

Learn more here (https://path.rsaconference.com/flow/rsac/us25/FullAgenda/page/catalog/session/1727605446614001V2zs).

Related Content

(https://www.ox.security/secrets-detection-why-it-matters-in-appsec/)

May 8, 2025

Secrets Detection: Why It Matters in AppSec (https://www.ox.security/secrets-detection-why-it-matters-in-appsec/)

By Katie Tatuch-Shtauro

(https://www.ox.security/shift-left-budget-right-getting-the-most-from-your-appsec-budget/)

April 30, 2025

Shift Left, Budget Right: Getting the Most From Your AppSec... (https://www.ox.security/shift-left-budget-right-getting-the-most-from-your-appsec-budget/)

By Ox Security

(https://www.ox.security/cve-2025-29927-is-your-middleware-really-protecting-you/)

April 29, 2025

CVE-2025-29927: Is Your Middleware Really Protecting You? (https://www.ox.security/cve-2025-29927-is-your-middleware-really-protecting-you/)

By Shir Rovnan

Take the OX challenge

Shrink security debt by 95% in less than 90 minutes

Book a Demo (/book-a-demo/)

(https://www.g2.com/products/ox-security/reviews?source=search)

Product

Platform Overview (https://www.ox.security/the-ox-active-aspm-platform/)

Software Supply Chain Security (https://www.ox.security/ox-for-software-supply-chain-security/)

ASPM (https://www.ox.security/ox-for-application-security-posture-management-aspm/)

Compliance (https://www.ox.security/assess-appsec-processes-ensure-compliance/)

API Exposure Management (https://www.ox.security/ox-for-api-exposure-management/)

Dev Empowerment (https://www.ox.security/eliminate-the-need-to-revisit-old-code-and-workflows-use-case/)

Resources

Blog (https://www.ox.security/blog/)

Resource Library (https://www.ox.security/resources/)

Podcasts (https://www.ox.security/podcasts/)

Events (https://www.ox.security/events/)

Documentation (https://docs.ox.security/)

Company

About Us (https://www.ox.security/about-ox/)

Newsroom (https://www.ox.security/newsroom/)

Partners (https://www.ox.security/partners/)

Careers (https://www.ox.security/careers/)

Contact (https://www.ox.security/contact/)

Stay Updated

Sign-up to receive the latest news, exclusive updates, and product insights

Business email* Subscribe

(https://www.linkedin.com/company/ox-security) (https://x.com/OX__Security) (https://www.youtube.com/@OXSecurity) (https://www.instagram.com/lifeatox/)