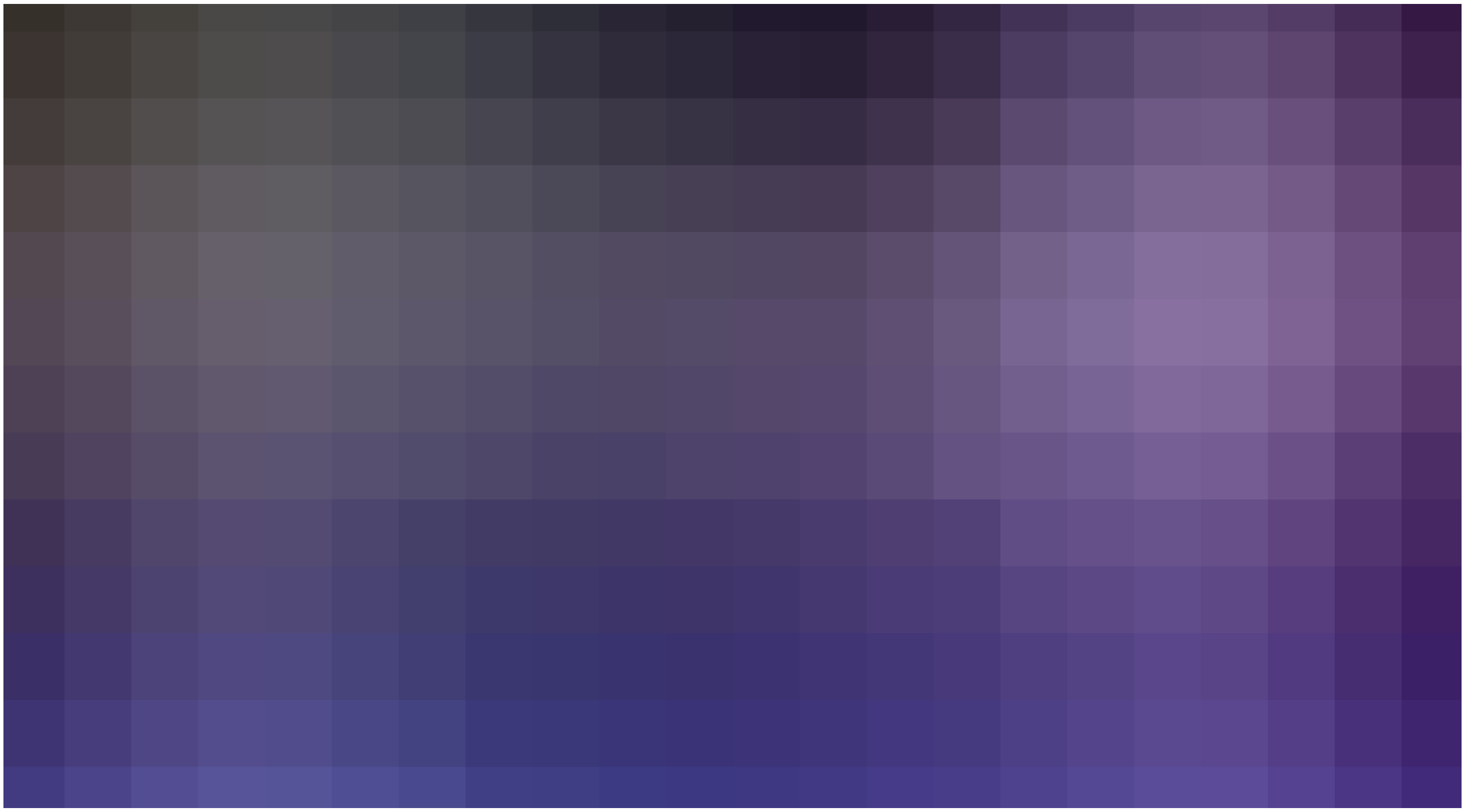
← Blog

GitHub Action tj-actions/changed-files supply chain attack: everything you need to know

A supply chain attack on popular GitHub Action tj-actions/changed-files caused many repositories to leak their secrets. Discover how it unfolded and the steps to mitigate the risk.

Merav Bar, Shay Berkovich, Gal Nagli
March 15, 2025

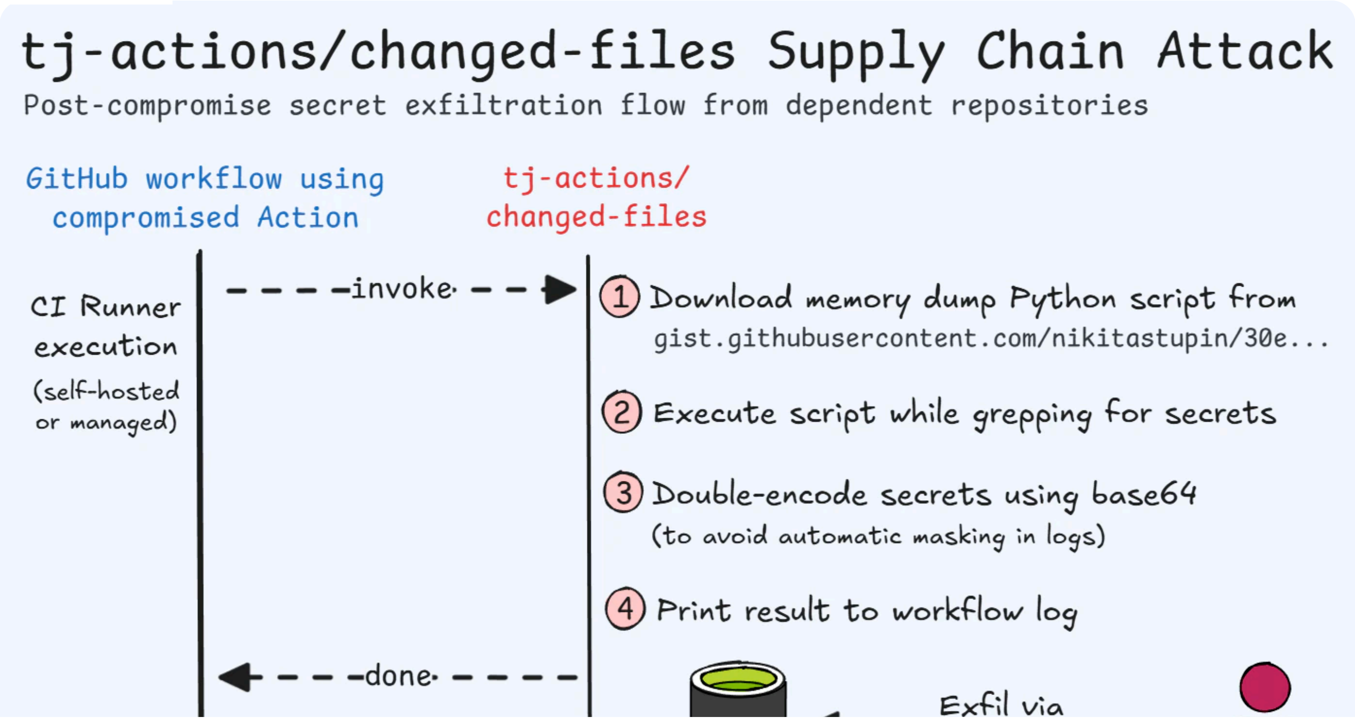
6 minute read



March 17, 2025 update: Wiz Threat Research has **identified** another compromised GitHub Action called `reviewdog/action-setup`, that may have contributed to the compromise of `tj-actions/changed-files`.

TL;DR

- As first reported by **Step Security**, the widely used GitHub Action `tj-actions/changed-files` was compromised sometime before March 14, 2025 with a malicious payload that caused affected public repositories to leak their secrets in logs. The compromise is also being tracked as a vulnerability, and has been assigned CVE-2025-30066.
- We don't yet know how exactly this happened, who is behind this, or what their specific goals were (edit: the tj-actions maintainers have since **stated** that the attacker somehow compromised a GitHub personal access token (PAT) used by a bot with access to the repo).
- The malicious commits to the `tj-actions/changed-files` repository have since been reverted, and the GitHub gist which stored a 3rd party script executed by the compromised action has been deleted. This means that future exploitation has been mitigated (with the possible exception of cached actions), and the risk of most concern is currently the continued exposure of secrets in the logs of affected repositories.
- Immediate response therefore remains necessary to mitigate the risk of credential theft and CI pipeline compromise. However, the primary risk is to **public repositories**, where secrets dumped into workflow logs are public as well; conversely, the risk to private repos is limited.
- Wiz Threat Research has observed first-hand the deployment of the script designed to dump secrets as part of the malicious payload's execution (as reported elsewhere as well). We've also identified dozens of impacted public repositories with exposed sensitive secrets, and have been reaching out to the affected parties.


tj-actions/changed-files Supply Chain Attack
Post-compromise secret exfiltration flow from dependent repositories
GitHub workflow using compromised Action
tj-actions/ changed-files
CI Runner execution (self-hosted or managed)
invoke
1 Download memory dump Python script from gist.githubusercontent.com/nikitastupin/30e...
2 Execute script while grepping for secrets
3 Double-encode secrets using base64 (to avoid automatic masking in logs)
4 Print result to workflow log
done
Exfil via

Overview of the compromised GitHub Action's malicious functionality

What is the impact on affected organizations?

The compromised action injected malicious code into any CI workflows using it, dumping the CI runner memory containing the workflow secrets. On public repositories, the secrets would then be visible to everyone as part of the workflow logs, though obfuscated as a double-encoded base64 payload. As of now, no external exfiltration of secrets to an attacker-controlled server were observed; secrets were only observable within the affected repositories themselves.

The GitHub gist hosting the malicious script was taken down at some point on March 15, 2025, and the compromised repository was also temporarily taken down around 10:30AM UTC on the same day, and later restored without the offending commits, meaning that future exploitation has been mitigated. However, there is still a risk of actions being cached and secrets that have already been leaked as a result of this compromise, so customer action is still required.

Which products are affected?

As of March 15, 2025, all versions of `tj-actions/changed-files` were found to be affected, as the attacker managed to modify existing version tags to make them all point to their malicious code. Customers who were using a hash-pinned version of `tj-actions/changed-files` would not be impacted, unless they had updated to an impacted hash during the exploitation timeframe.

How did this happen?

The investigation is still ongoing – at the moment, we can only theorize that the attacker gained sufficient access to update tags to the malicious code they had placed on a fork of the repository (edit: the `tj-actions` maintainers have since **stated** that the attacker somehow compromised a GitHub personal access token (PAT) used by `@tj-actions-bot`, a bot with privileged access to the compromised repository). While the attacker impersonated the Renovate bot user, the lack of GitHub verification on the commit shows that user was not compromised. We'll update this blogpost as new information comes to light.

What's the risk to cloud environments?

In workflows that perform production deployments, leaked secrets might include keys allowing access to cloud production environments as well as internal source code repositories.

What sort of exploitation has been identified in the wild?

While conducting threat hunting related to this malicious activity, in several instances Wiz Threat Research has observed the deployment of a script designed to dump secrets as part of the malicious payload's execution. Additionally, Wiz Threat Research has so far identified dozens of repositories affected by the malicious GitHub action, including repos operated by large enterprise organizations. In these repositories, the malicious payload successfully executed and caused secrets to leak in workflow logs. Some of the leaked secrets we've identified so far include valid AWS access keys, GitHub Personal Access Tokens (PATs), npm pokens, private RSA Keys and more.

Which actions should security teams take?

1. Use **this Github query** to find references to the affected GitHub action in your organization's repositories (replace `your-org` with the name of your organization).
2. If any affected repositories are identified, navigate to the "Actions" tab or click "View Runs". Check for any GitHub Actions that include the affected compromised component and ran during the relevant timeframe.

3. Search for "Get changed files" within the Action run, expand the changed-files line and check whether there is a double-encoded base64 string within. If one exists, then that means that the malicious payload was successfully executed, and the impact depends on whether the repo is public or private: If the repo is public, then it means that the secrets in the string were publicly leaked in workflow logs, and we highly recommend rotating them as soon as possible. Otherwise, if the repo is private, then that means the secrets were not leaked publicly, but you should still consider rotating them.

Note that if your repository was impacted but the only secrets exposed were GitHub tokens beginning with the prefix `ghs_`, these are short-lived tokens that automatically expire within 24 hours or once the workflow job has completed. Consequently, unless a job was interrupted and failed to finish, there is limited long-term risk from the exposure of a `ghs_` token alone.

Additionally, if your workflow does not explicitly reference custom secrets (for example, `token: ${{ secrets.MYSECRET }}`), it is unlikely that sensitive secrets were compromised as a result of this incident. The primary risk lies in the potential exposure of explicitly referenced custom secrets within affected workflows.

What should I do if my organization is affected?

1. Stop using `tj-actions/changed-files` immediately and replace it with a safer alternative if possible.
2. Remove all references to the action across all branches of your repositories, not just the main branch, to prevent potential execution in other branches.

3. Rotate any leaked secrets as soon as possible. Deleting the relevant workflow will also remove all the logs, which can prevent further exposure of the secrets. However, it is also recommended to download workflow logs from the exposure window before deleting anything.

How can I prevent this sort of risk in the future?

1. **As recommended by GitHub**, pin all GitHub Actions to specific commit hashes instead of version tags to mitigate against future supply chain attacks.
2. Audit past workflow runs for suspicious activity. Check logs for unusual outbound network requests, and prioritize reviewing repositories where CI runner logs are publicly accessible, as secrets may have been exposed in logs.
3. Use GitHub's allow-listing feature to block unauthorized GitHub Actions from running and configure GitHub to allow only trusted actions.

How can Wiz help?

1. Wiz Code customers with GitHub connector can use the created controls on affected code repositories to detect the presence of compromised actions. In addition, the vulnerability finding was created to identify any instances of CVE-2025-30066 in your environment.
2. Wiz Sensor customers with Sensor deployed on CI workloads can detect malicious activity related to this attack, such as process memory dump and script deployment (see screenshot below demonstrating a detection of **James Berthoty's attack simulation**).

3. Wiz Defend customers with GitHub audit logs enabled are protected against potential secondary malicious activities facilitated by the abuse of any compromised credentials, such as workflow log deletion or merges performed by an unusual bot user.
4. Wiz customers can also refer to the Threat Intel Center advisory for this attack, which includes additional capabilities and will be updated as new ones are added.

Wiz Threat Intel Center advisory

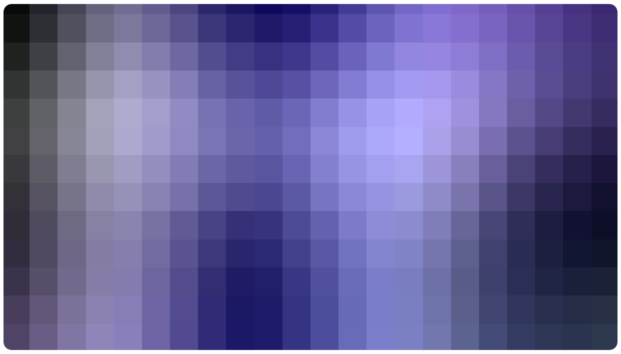
Wiz Sensor detection of memory dump execution

References

- Step Security blog
- Semgrep blog
- GitHub guidance for third party actions

Tags #Research #Product

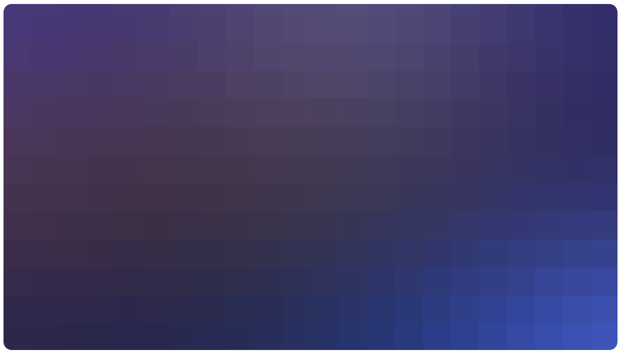
Continue reading



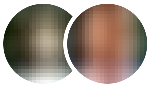
Securing the Container Frontier: Kubernetes Trends Report 2025

Shay Berkovich
January 23, 2025

From rapid-fire attack attempts to evolving defense strategies, our Kubernetes Security Report paints a...

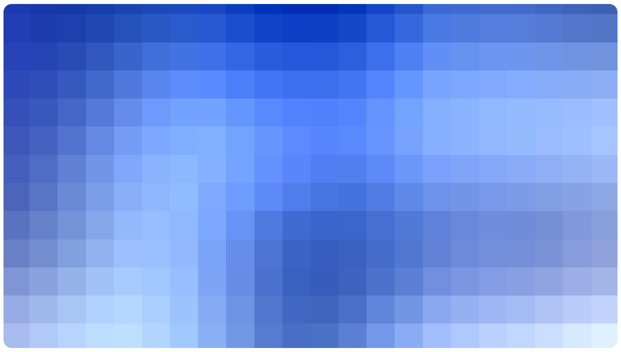


AskAI – Text to Security Graph Query

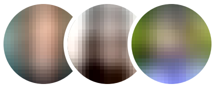


Daniel Lazarev, Erez Harush
October 23, 2024

AskAI – Text to Security Graph Query



A Cloud-First Approach to Vulnerability Remediation: A Holistic Approach



Thomas Nuth, Shaked Rotlevi, Asaf Wiener
December 10, 2024

Learn about how Wiz helps organizations operationalize vulnerability remediation with true code-to-cloud visibility

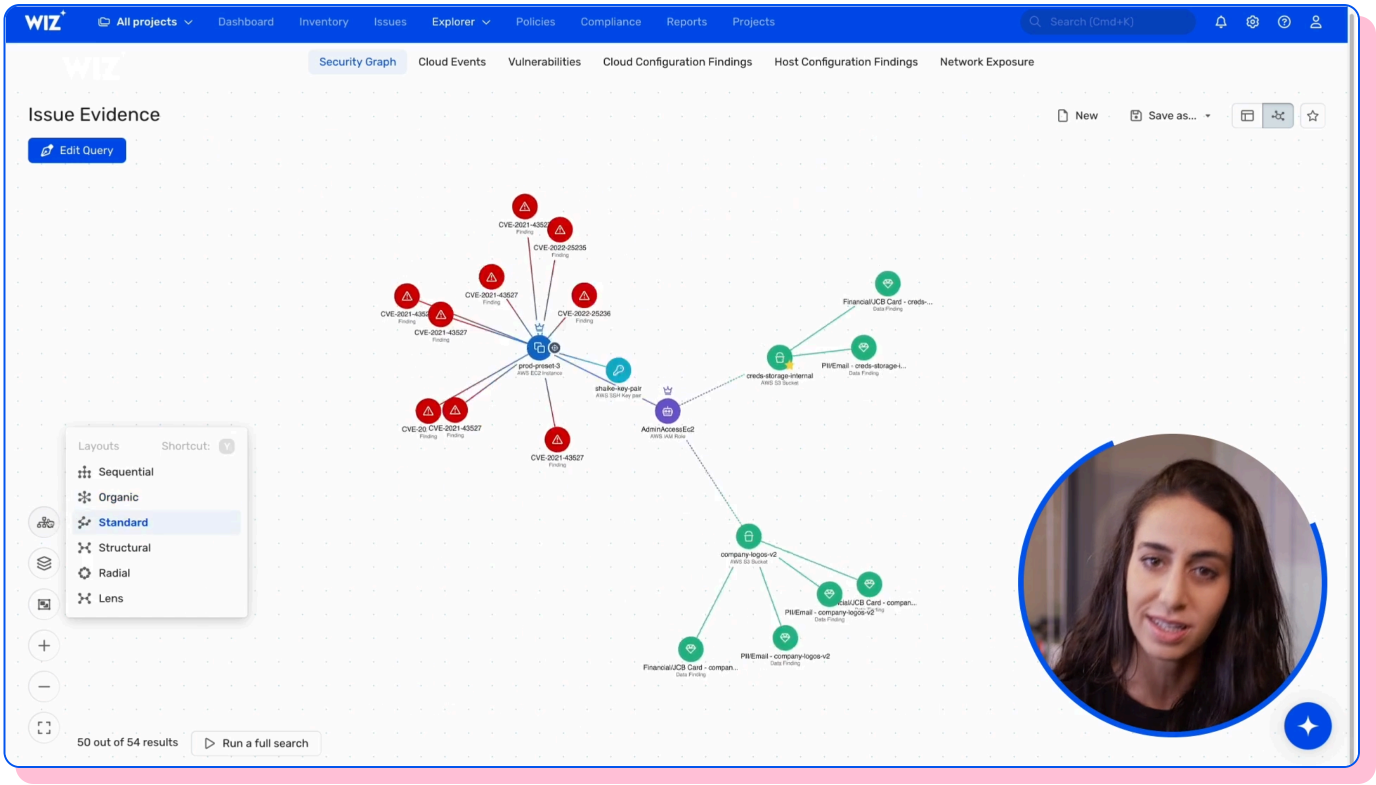
GET A PERSONALIZED DEMO

Ready to see Wiz in action?

> “Wiz provides a single pane of glass to see what is going on in our cloud environments.”

Blackstone

Adam Fletcher
Chief Security Officer



Get a demo ›

PLATFORM

Wiz CNAPP

Wiz Code

Wiz Cloud

Wiz Defend

Integrations

Environments

Documentation

LEARN

Customer stories

Train Your Team For Cloud

Blog

CloudSec Academy

Resources Center

Cloud threat landscape

Cloud Security Assessment

Vulnerability Database

COMPANY

About Wiz

Join the team

Newsroom

Events

Contact us

Trust Center

Our partners

English (US)

Status Privacy Policy Your California Privacy Rights Terms of Use Modern Slavery Statement

Do Not Sell or Share My Personal Information