tj-actions/changed-files

In March 2025, a stolen bot token let attackers repoint many tj-actions/changed-files version tags to one commit that printed CI secrets into public build logs.

Reflects the csoh.org kill chain as of 2026-11-22

Cloud Security Office Hours, csoh.org

Slides licensed CC BY-NC 4.0

How to use this deck

Read it on your own or present it. Every fact about the incident comes from the csoh.org kill chain and the sources it cites.

Slides marked Background explain a general idea and name their own source. The speaker notes are a presenter's script.

Full kill chain: csoh.org/breaches/tj-actions-changed-files.html. These slides reflect it as of 2026-11-22.

At a glance

23,000

or more repositories referenced it

Most callers pinned a mutable tag, not a commit SHA

Dozens

of cases of confirmed live secret leakage

Per Wiz Threat Research (disclosure: the site's author works at Wiz)

218

public repositories leaked secrets, per Endor Labs

Of 614 that ran it in the window

CVE-2025-30066. Root cause: a compromised bot personal access token (PAT), per the maintainers. The page traces it to reviewdog.

What happened, and when

Earlier

reviewdog/action-setup@v1, poisoned to dump secrets, exposes the tj-actions-bot PAT

March 14 to 15, 2025

The days NVD gives for the affected tags; secrets readable in public build logs

Around March 15

The malicious gist and commit are taken down

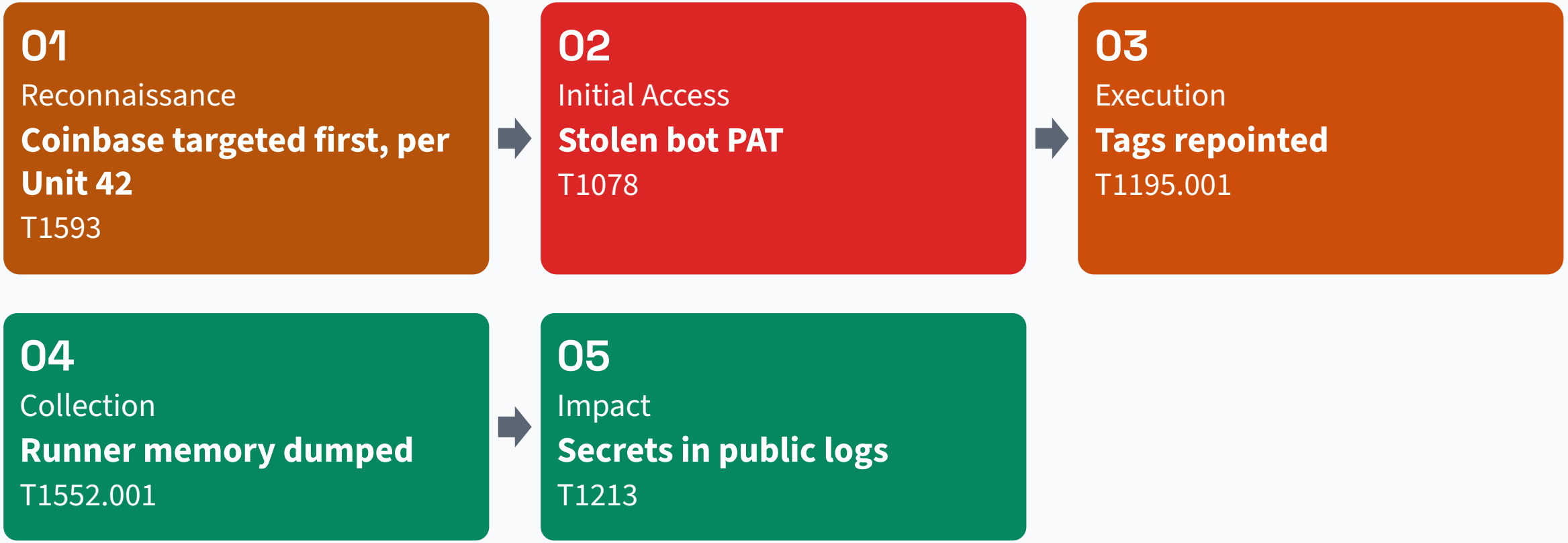
Endor Labs measured the window as 15:00 to 15:00 UTC. CISA asks for an audit from March 12 to 15. The page gives no date for the clean release, v46.0.1. Spacing is not to scale.

How many leaked: two sources, two counts

Wiz Threat Research	Endor Labs
• Confirmed live secret leakage in only dozens of cases • Gives no breakdown of the rest • SHA-pinned workflows unaffected unless updated to an impacted hash during the exploitation timeframe • Notes private repositories did not leak secrets publicly	• 5,416 public repositories referenced the action • Only 614 of them ran it during the window • 218 of those leaked secrets

The page reports both counts side by side and does not reconcile them.

The kill chain on one slide



Arrows follow the page's stage order.

01 Per Unit 42, Coinbase was the first target

T1593 Search Open Websites/Domains

Action	tj-actions/changed-files, a file-change detection action
Reach	referenced by more than 23,000 repositories
Pinning	most callers pinned to mutable tags like v45, not commit SHAs
Coinbase project	agentkit, whose workflow used v39 of the action, per Unit 42

WHY IT MATTERS

Attackers who could push to the action would inherit the trust and the secrets of every downstream workflow that ran it.

BACKGROUND

What GitHub Actions, workflows and runners are

A workflow is a YAML file under .github/workflows/. On an event it lists, GitHub spins up a fresh virtual machine, the runner, runs the steps and tears it down. A step can run a script or use a published Action.

Source: How We Use GitHub Actions (csoh.org)

02 A compromised bot token, traced to reviewdog, gave write access

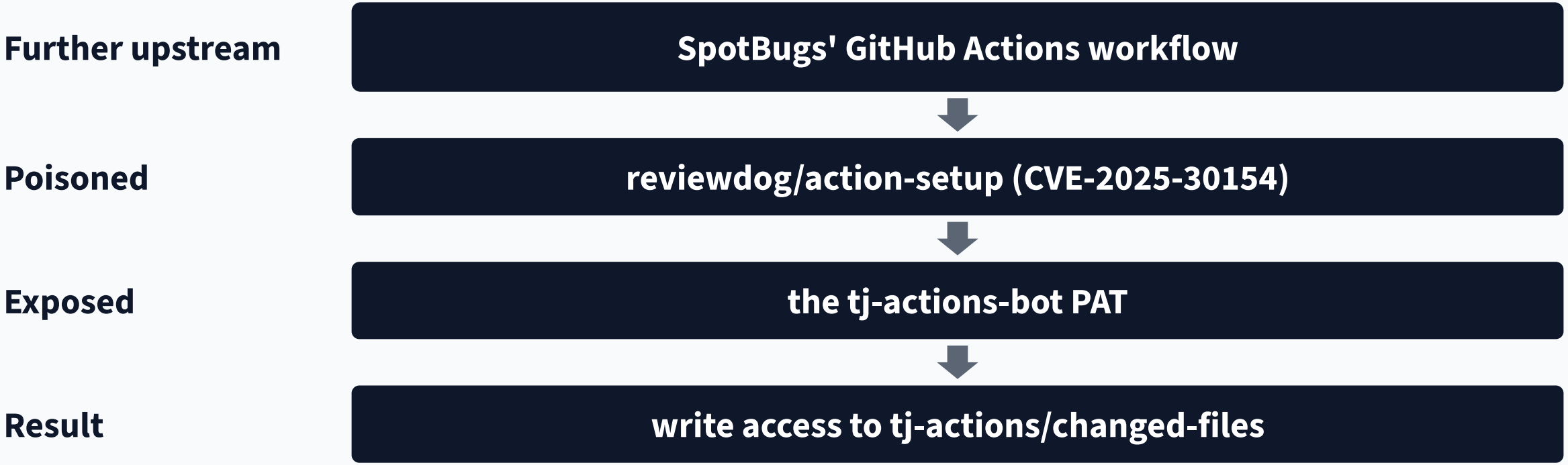
T1078 Valid Accounts

Root cause	a compromised tj-actions-bot PAT, per the maintainers
Exposed through	reviewdog/action-setup@v1, itself poisoned to dump secrets
Dependency	tj-actions/eslint-changed-files relied on reviewdog
Further upstream	SpotBugs' workflow, per analysts including Unit 42

WHY IT MATTERS

Credentials leaked in the reviewdog attack were reused to gain write access.

The upstream chain, as the page draws it



Analysts including Unit 42 traced the SpotBugs hop. tj-actions depended on reviewdog through tj-actions/eslint-changed-files. The page does not say how SpotBugs' workflow led to reviewdog.

BACKGROUND

What ATT&CK means by compromising development tools

ATT&CK describes adversaries manipulating software dependencies and development tools before a final consumer receives them. It says CI/CD components such as GitHub Actions may be targeted to reach an application's build, test and deployment cycles.

Source: MITRE ATT&CK T1195.001 (attack.mitre.org)

03 Many version tags were repointed to one malicious commit

T1195.001 Compromise Software Dependencies and Development Tools

Commit	a malicious payload, committed with the write access
Tags	many existing version tags rewritten to resolve to that commit
Dropper	Node.js function, base64-encoded instructions, Python script from a GitHub gist

WHY IT MATTERS

Consumers who trusted a tag such as v45, assuming it was an immutable, reviewed release, silently began running attacker code on their next CI run.

BACKGROUND

Version tags, commit SHAs and pinning

A reference like actions/checkout@v4 resolves to whatever the publisher has pointed that version tag at right now. Pinning names an immutable identifier instead, such as a 40-character commit SHA, which the publisher cannot quietly redefine.

Source: csoh.org glossary, Pinning (csoh.org)

04 The payload scraped runner memory and printed secrets into the log

T1552.001 Unsecured Credentials: Credentials In Files

Method	a Python script scanning runner memory, where secrets sit in cleartext during jobs
Harvested	included AWS access keys, GitHub PATs, npm tokens, private RSA keys
Output	a double base64-encoded blob in the job's standard output

WHY IT MATTERS

It needed no outbound C2 channel, and it blended into normal log volume.

05 Anyone could read the secrets in public logs

T1213 Data from Information Repositories

Window	March 14 to 15, 2025, the days NVD gives for the affected tags
Confirmed	live secret leakage in only dozens of cases, per Wiz
Endor Labs	of 614 public repos that ran it in the window, 218 leaked
Remediation	gist and commit down around March 15; clean release v46.0.1

WHY IT MATTERS

Any public workflow that ran in the window should treat all its CI secrets as compromised and rotate them.

A supply-chain attack on a supply-chain tool

Stages 02 to 04 show the summary's pattern.

- A poisoned reviewdog action exposed the PAT with write access to tj-actions.
- Mutable tags carried one malicious commit into unchanged workflows.
- Secrets went into the build log, with no outbound C2 channel.

Five defenses against five stages

	01	02	03	04	05
Pin Actions to a full commit SHA	●		●		
Fewer, scoped secrets in CI				●	
Allowlist which Actions may run (no stage named)					
Monitor build logs and egress			●	●	
Fast, rehearsed secret rotation				●	●

No defense on the page is tied to stage 02. The allowlist defense cites no stage's facts.

Pin to a commit SHA; allowlist the Actions that may run

Pin (stages 01, 03)	Allowlist (no stage cited)
• A tag like v45 can be silently repointed • A 40-character commit SHA makes a release immutable • Dependabot can propose reviewed SHA bumps	• Allow only specific verified or SHA-pinned Actions • Set it at the org or repo level; block everything else • Turns a supply-chain incident into a policy denial, not a breach

The page says a repointed tag is exactly what happened here.

Hold fewer secrets in CI; watch the logs and the egress

Fewer secrets (stage 04)	Monitor (stages 03, 04)
• A memory scraper can only steal what the runner holds • Least privilege; short-lived OIDC federation instead of long-lived AWS keys • Avoid injecting broad PATs	• Alert on unexpected outbound calls from runners • Alert on large or high-entropy base64 output in logs • Scan Actions logs with secret-scanning tooling

The payload printed base64 secret blobs and pulled code from an external gist.

Rotate within hours; treat secrets an affected runner touched as burned

When a dependency is compromised, rotation speed decides the impact.

- Inventory every credential reachable from CI.
- Know which workflows ran in the exposure window.
- Be able to revoke and reissue AWS keys, PATs, npm tokens and signing keys within hours.

Check your own environment

- [ ] Are your third-party Actions pinned to a full commit SHA?
- [ ] Does something like Dependabot propose reviewed SHA bumps?
- [ ] Does your org or repo allowlist which Actions may run?
- [ ] Do your CI jobs hold long-lived AWS keys or broad PATs?
- [ ] Would an unexpected outbound call from a runner raise an alert?
- [ ] Do you scan Actions logs for secrets and high-entropy base64?
- [ ] Could you list which workflows ran in a window, and rotate within hours?

DISCUSSION

Secrets went to the log, not a server. Would you have noticed?

The payload printed what it stole into the job's own log as a double base64-encoded blob, with no outbound C2 channel.

Which of your controls would have seen it, and where?

DISCUSSION: ANSWER

The page watches the logs as well as the network

Its monitoring defense names both: the base64 blobs in the log and the pull from a gist.

- Alert on large or high-entropy base64 output in logs.
- Scan Actions logs with secret-scanning tooling.
- Alert on unexpected outbound calls from runners.

DISCUSSION

What would you add for stage 02?

No defense on the page is tied to stage 02: the tj-actions-bot PAT, exposed through a poisoned reviewdog action.

What would you add, and which reported detail would it catch?

DISCUSSION: ANSWER

The page ties no defense to stage 02

Any control you propose is your own suggestion. Test it against these reported details.

- The tj-actions-bot PAT and its write access
- The dependency on reviewdog via tj-actions/eslint-changed-files
- The SpotBugs workflow upstream, per analysts including Unit 42

Primary sources

Disclosure: research published by Wiz is one of these sources, and the site's author works at Wiz.

Wiz Threat Research, tj-actions/changed-files attack
wiz.io

Endor Labs, blast radius of the attack
endorlabs.com

CISA advisory
cisa.gov

Unit 42, attack chain
unit42.paloaltonetworks.com

NVD, CVE-2025-30066
nvd.nist.gov/vuln/detail/cve-2025-30066

Related defense topics

The page links no other kill chain; these are its related defense topics.

GitHub Actions security
csoh.org/github-actions.html

CI/CD security
csoh.org/ci-cd.html

Git and version control
csoh.org/version-control.html

Cloud threat research
csoh.org/threat-research.html

Read the full kill chain

Every step, source and defense in detail:
https://csoh.org/breaches/tj-actions-changed-files.html

Cloud Security Office Hours, csoh.org. Slides licensed CC BY-NC 4.0 (creativecommons.org/licenses/by-nc/4.0). Third-party excerpts keep their own licenses.