GrafanaCON '26

Lessons from that security incident when everything went wrong (but ended up right)

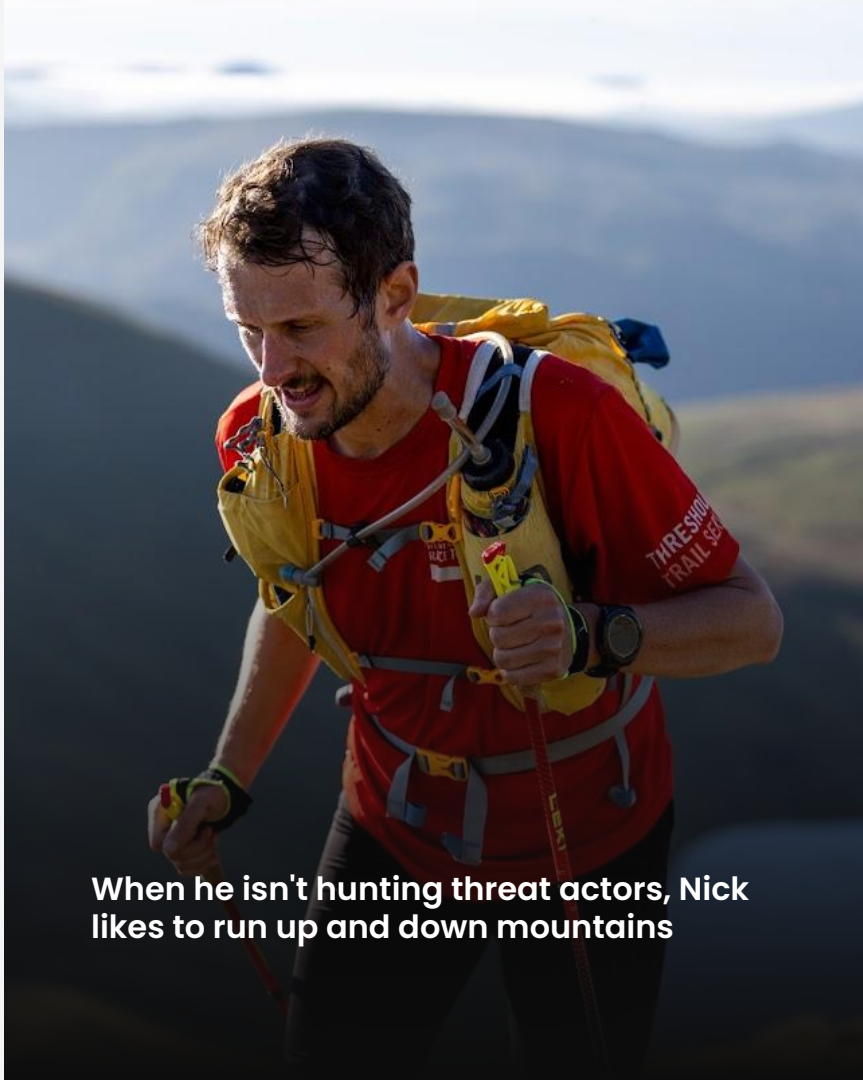
Nick Moore
Principal Security Engineer
Grafana Labs

David Andersson
Director Security Engineering
Grafana Labs

Nick Moore

Principal Security Engineer

Grafana Detection & Response Engineering (DRE)



David Andersson

Director of Security Engineering

Saturday morning

A normal weekend

A single alert fires, notifying some eagle-eyed security team members

Complete CI/CD compromise

All of Grafana's GitHub secrets had been exfiltrated

No customer impact

Conclusively showed no successful use of our secrets

What went wrong?

The subtle trap: `pull_request` vs `pull_request_target`

pull_request

pull_request_target

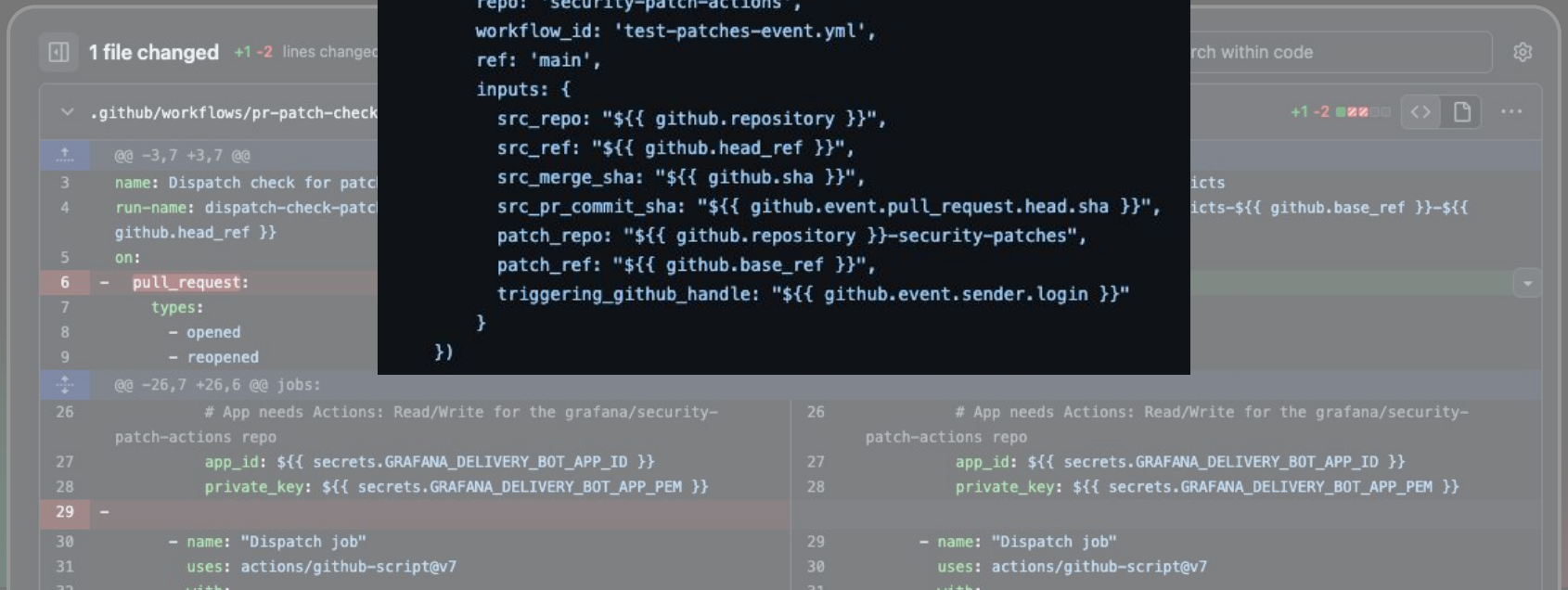
How?

One small change

A small change for Grafana,

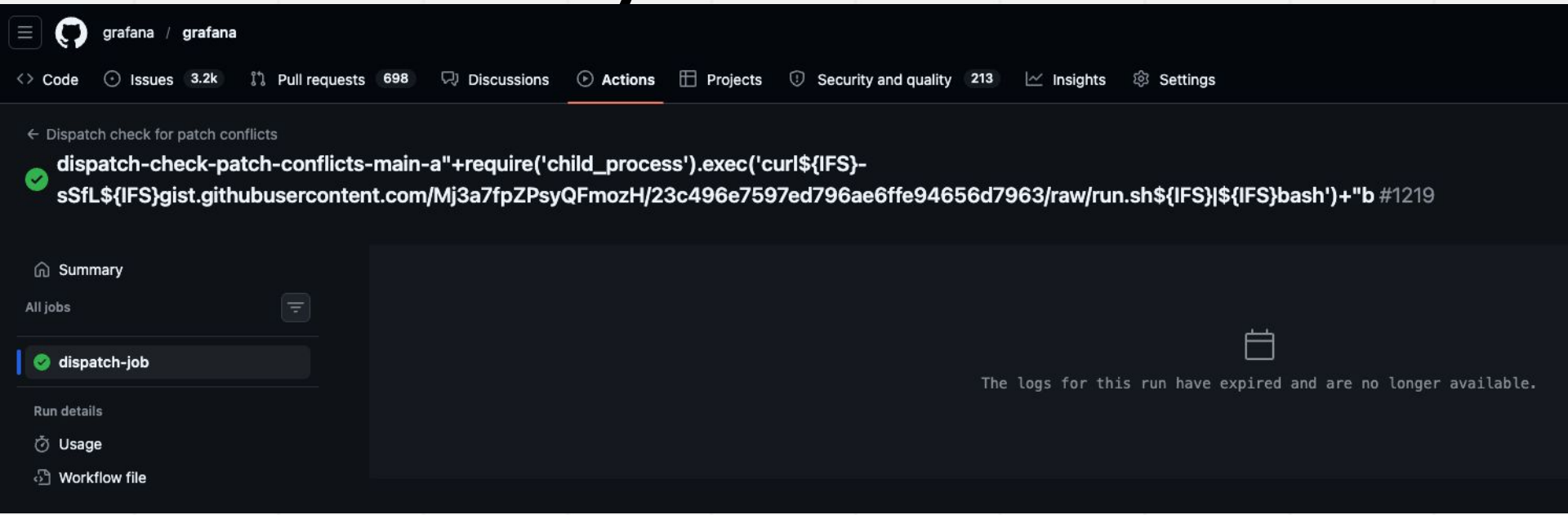
A giant opportunity for hackers!

```
script: |
  await github.rest.actions.createWorkflowDispatch({
      owner: 'grafana',
      repo: 'security-patch-actions',
      workflow_id: 'test-patches-event.yml',
      ref: 'main',
      inputs: {
        src_repo: "${{ github.repository }}",
        src_ref: "${{ github.head_ref }}",
        src_merge_sha: "${{ github.sha }}",
        src_pr_commit_sha: "${{ github.event.pull_request.head.sha }}",
        patch_repo: "${{ github.repository }}-security-patches",
        patch_ref: "${{ github.base_ref }}",
        triggering_github_handle: "${{ github.event.sender.login }}"
      }
  })
```



```
script: |
  await github.rest.actions.createWorkflowDispatch({
      owner: 'grafana',
      repo: 'security-patch-actions',
      workflow_id: 'test-patches-event.yml',
      ref: 'main',
      inputs: {
        src_repo: "${{ github.repository }}",
        src_ref: "${{ github.head_ref }}",
        src_merge_sha: "${{ github.sha }}",
        src_pr_commit_sha: "${{ github.event.pull_request.head.sha }}",
        patch_repo: "${{ github.repository }}-security-patches",
        patch_ref: "${{ github.base_ref }}",
        triggering_github_handle: "${{ github.event.sender.login }}"
      }
  })
```

Key to the door




ANY SCRIPT IN YOUR WORKFLOW
WHEN IT SEES $GITHUB_TOKEN AND ENV VARS
It's Free


VAULT
AFTER THE ATTACKER REALIZES THEY NEED A VALID
OIDC TOKEN AND THE CORRECT APPROLE TO SEE A SINGLE STRING

The Attack Timeline

2025-04-25
16:40 UTC

2025-04-25
17:52 UTC
(Zero Hour)

2025-04-26
01:05 UTC
(~+7h 15m)

2025-04-26
04:30 UTC
(~+10h 40m)

2025-04-26
06:15 UTC
(~+12h 25m)


AdnaneKhan's
GATO-X

Grafana IRM

Incident response tooling, built directly into Grafana Cloud

Received initial alert and alerted responders

Provided a centralized location for incident discussion, analysis, and announcements

Grafana Loki

Ingest huge volumes of logs, great for detailed sources like GitHub Actions

Supports powerful, complex queries, allowing us to precisely identify relevant log entries

Combined with Grafana Alerting, gives us fast response to future events of concern

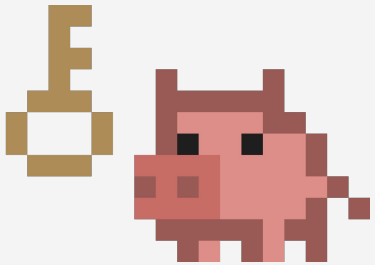
Zizmor

Static analysis for GitHub Action workflows

Identifies relevant risks, prioritized by impact and likelihood of exploitability

Now includes automated resolution suggestions thanks to contributions from our team!

Trufflehog



Open Source

Verified / Unverified

Tries authentication using the secret

Follows the normal flow

Gato-X

Open source suite of GHA attack tools

The solution used by our attacker!

Want to make sure we can't be attacked with it again? Use it ourselves to be sure!

How?

Canary Tokens

Permission-less tokens for empty AWS projects, alerting the security team when they are used

Makes hackers think twice about any credential they come across - the hunter becomes the hunted!

Trufflehog validation post-exfiltration triggered our response*

The Response Timeline

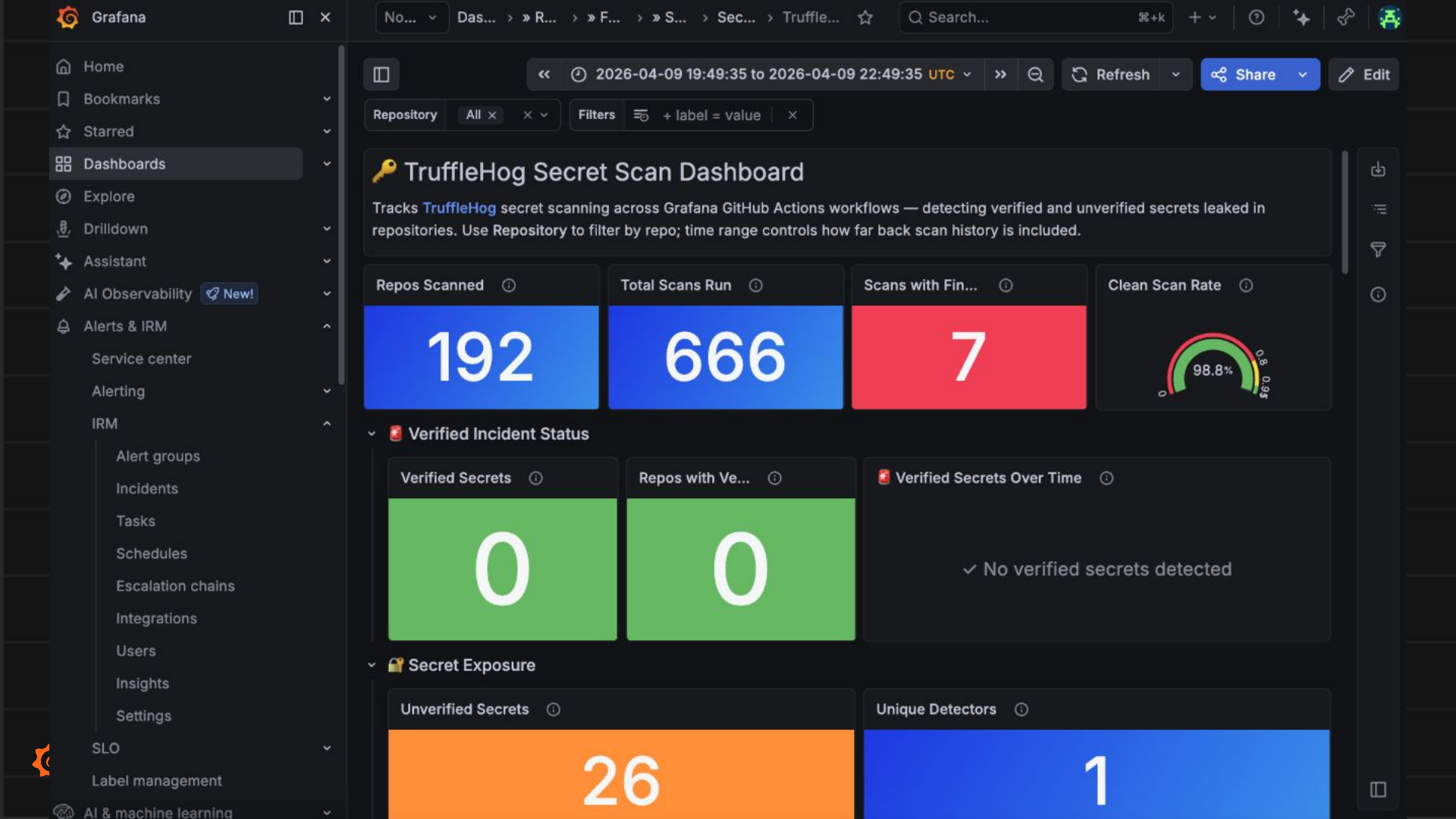
2025-04-26
04:30
Attack event

2025-04-26
06:15
Canary token fires

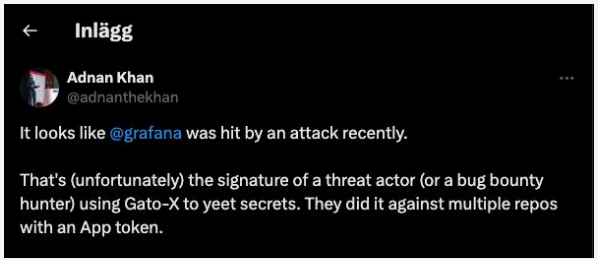
2025-04-26
08:00
Incident declared

2025-04-26
11:00
Logs validation

2025-04-26
11:00-23:30
Token rotation

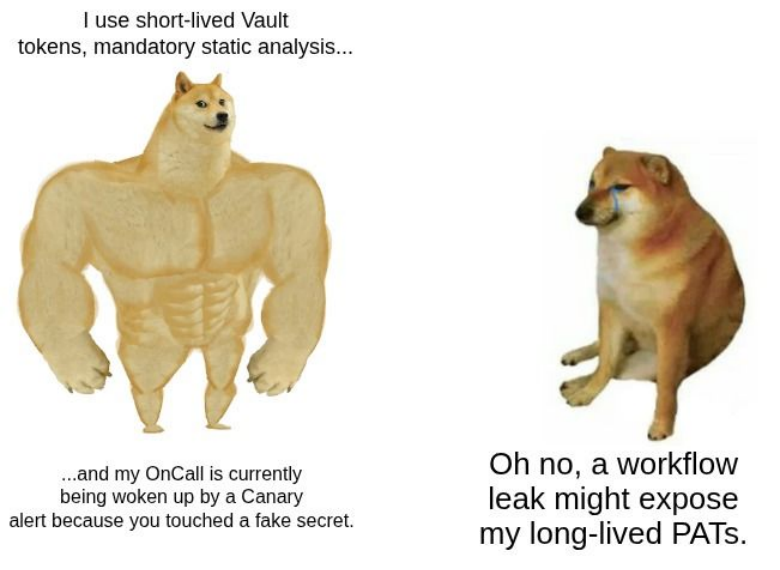

Grafana
Home
Bookmarks
Starred
Dashboards
Explore
Drilldown
Assistant
AI Observability
New!
Alerts & IRM
Service center
Alerting
IRM
Alert groups
Incidents
Tasks
Schedules
Escalation chains
Integrations
Users
Insights
Settings
SLO
Label management
AI & machine learning
Search...
2026-04-09 19:49:35 to 2026-04-09 22:49:35 UTC
Refresh
Share
Edit
Repository
All
Filters
+ label = value
TruffleHog Secret Scan Dashboard
Tracks TruffleHog secret scanning across Grafana GitHub Actions workflows — detecting verified and unverified secrets leaked in repositories. Use Repository to filter by repo; time range controls how far back scan history is included.
Repos Scanned
192
Total Scans Run
666
Scans with Fin...
7
Clean Scan Rate
98.8%
Verified Incident Status
Verified Secrets
0
Repos with Ve...
0
Verified Secrets Over Time
No verified secrets detected
Secret Exposure
Unverified Secrets
26
Unique Detectors
1

What went right?



- Preparation beats reaction: canaries, static analysis, secret hygiene
- Observability isn't just for production — your CI/CD pipeline needs it too
- Open source tools and open collaboration are a security advantage, not just an engineering one
- You get to write "no customer impact" when you've done the work beforehand

Lessons learned



Moved **all** our CI/CD secrets from GitHub secrets to Vault

Implemented mandatory Zizmor and Trufflehog scans

Broadened canary token coverage in GitHub and ensured alerts will wake our OnCall 😴🥱

Reduced the access scope of widely used GitHub Apps, eliminating catch-all accesses

User education around GitHub workflow vulnerabilities

Oops, we did it again...

Thank you!