Securing Open Source

Lessons from the GitHub Security Lab

Open Source Summit NA · 2026

Bas Alberts
@anticomputer
GitHub Security Lab

Open source runs the world.

And we all share the bill when it breaks.

The scale we're talking about

Hundreds of thousands of maintainers

Millions of packages

100M+ developers

Billions of devices

Sources: GitHub Octoverse 2024 · npmjs.com

The challenges

Asymmetry between security experts and developers

Although the commoditization of security skills through LLMs is closing that gap rapidly — when leveraged with care.

Adversarial innovation

Offense often moves faster than defense, and is quicker to adopt new tech — e.g. the axios npm social-engineering compromise, March 2026.

Tragedy of the commons

Everyone uses it. No one pays for the upkeep.

The GitHub Security Lab

Helping secure open source software

What we do

Open Source Audits

Find vulnerabilities in widely-used OSS and work with maintainers on fixes.

Education

Free CodeQL training, the Secure Code Game, and security training for Secure Open Source Fund participants.

Advisory DB Curation

Curate the GitHub Advisory Database so alerts come with useful context.

By the numbers

1,213

vulnerabilities found by Security Lab researchers

906

CVEs credited

30,000+

advisories curated in the GitHub Advisory Database

14,000+

CVEs assigned for OS maintainers

Source: securitylab.github.com (verified 2026-05-18)

Three principles

#1 · Make it easy

Security as the default path. Tooling, automation, secure defaults.

#2 · Make it delicious

Developer experience matters. If it isn't pleasant, it won't get used.

#3 · Make it social

Open source is a team sport. Invest resources in your dependencies.

#1 · Make it easy

CodeQL · a short timeline

2019	CodeQL launches as part of GitHub Advanced Security
2023	One-click default setup for code scanning
2024	Buildless mode for Java and C# · Copilot Autofix GA
2025	Buildless mode for C/C++ · Autofix coverage expanded
2026	PR insights extended to all protected branches · ongoing language coverage

Sources: codeql.github.com · github.blog changelog (2024-03, 2024-06, 2024-08, 2025-02, 2025-06, 2026-03)

Copilot Autofix

3×

Faster median time to commit a fix vs. manual remediation.

Up to 7× faster for XSS · 12× faster for SQL injection.

Source: github.blog/news-insights/product-news/secure-code-more-than-three-times-faster-with-copilot-autofix/ (2024)

Finding a bug is half the work.

What about disclosure?

How we disclose

Research

Find vulnerabilities — CodeQL variant analysis, targeted audits, and LLM-aided discovery with careful validation.

Coordinate

Private report; flexible 90-day window, with a maintainer-first posture from our team.

Publish

CVE assigned, advisory added to the GitHub Advisory Database, fix released.

Sources: securitylab.github.com · docs.github.com (coordinated disclosure)

Case study · GitHub Actions workflows

75+

workflows secured

90+

vulnerabilities disclosed

Now shipped as **built-in CodeQL queries** — taint tracking, Bash support, workflows as a first-class language. Free for every public GitHub repository.

Source: github.blog/security/application-security/how-to-secure-your-github-actions-workflows-with-codeql/ (Jan 2025, updated Feb 2026)

The pattern

01

Find

Discover a vulnerability through research, audit, or AI-assisted exploration.

02

Extract

Identify the core pattern — the sources, sinks, taint flow, and control surfaces.

03

Codify

Capture it as a reproducible CodeQL query — deterministic, auditable, version-controlled.

04

Scale

Run across thousands of repos via variant analysis; ship to every default-setup repo.

Sources: codeql.github.com · github.blog (CodeQL Actions support, MRVA)

AI fits the same pattern

LLMs are good at finding

- Model APIs that humans miss
- Spot variants in unfamiliar code
- Triage alerts that resist heuristics

But their findings are **variable** — sometimes brilliant, sometimes hallucinated.

CodeQL makes them durable

- Encode the best LLM-found pattern as a deterministic query
- Run it across thousands of repos via MRVA
- Ship it as a default code-scanning query for every public repo

The same pattern, with **AI in the loop**: find faster, codify the keepers, scale.

Sources: github.blog (CodeQL+AI 2023 · Taskflow Agent Jan 2026 · MRVA)

In maintainers' hands

800+

candidates in our active triage pipeline (open-source taskflows)

20+

already disclosed (and counting)

Same Taskflow framework. **Open source.** You can go run this on your own project right now — read the blog.

Source: github.blog/security/how-to-scan-for-vulnerabilities-with-github-security-labs-open-source-ai-powered-framework/ (March 2026)

Try it yourself

1 · Open

Go to seclab-taskflows and launch a Codespace.

2 · Run

```
./scripts/audit/run_audit.sh myorg/myrepo
```

3 · Review

Results land in SQLite. Filter for `has_vulnerability`. Run multiple times — LLMs are non-deterministic.

Heads up: needs a Copilot license (free for verified OSS maintainers) · uses premium model requests · multi-run with different models (e.g., GPT 5.2 + Claude Opus 4.6) is recommended.

Source: github.blog/security/how-to-scan-for-vulnerabilities-with-github-security-labs-open-source-ai-powered-framework/ (Mar 2026)

Disclosure at ecosystem scale

4,101

Reviewed advisories published in 2025.

+19% more *new* advisories year-over-year.

Source: github.blog/security/supply-chain-security/a-year-of-open-source-vulnerability-trends-cves-advisories-and-malware/ (2026)

Disclosure in the AI era

Discovery is commoditizing

For us. For researchers. For bad-faith reports too.

By mid-2025, curl reported **~20% of bug submissions were AI slop** — only ~5% legitimate. The 90-day model strains under volume, overlap, and slop indistinguishable from real reports.

Expanding Private Vulnerability Reporting

Native platform features for the AI era.

PVR has been GA since 2023; at launch, 30k+ orgs and 180k+ repos had adopted it. We're investing in features to scale triage and surface signal over noise.

Sources: github.blog PVR GA (Apr 2023) · daniel.haxx.se "Death by a thousand slops" (Jul 2025) · forward-looking direction from GitHub

But disclosure is reactive.

What about preventing the next one?

A decade of supply-chain wake-up calls

2014	Heartbleed — OpenSSL memory disclosure
2017	Equifax — Apache Struts 2 RCE
2021	Log4Shell — log4j RCE, "the day the internet broke"
2024	XZ backdoor — social engineering of a single maintainer
2025	Shai-Hulud — self-replicating npm worm

Shai-Hulud · September 2025

500+

compromised packages removed from npm

Sept 14

first reported

A self-replicating worm. Phishing → stolen tokens → malicious post-install scripts → more stolen tokens. Spread across thousands of repositories before it was cut off.

Source: github.blog/security/supply-chain-security/our-plan-for-a-more-secure-npm-supply-chain/ (2025-09-22)

And it's ongoing.

Shai-Hulud 2.0 (Nov 2025): 700+ packages · 25K+ repos

Pre-install hooks, lateral movement to GitHub Actions runners; Microsoft published dedicated detection guidance in Dec 2025.

"Mini" Shai-Hulud (May 11, 2026)

170+ npm packages and — for the first time — PyPI in a single coordinated campaign. TanStack et al. Persistence now plants in `.claude/` *hooks so future AI-tool sessions re-execute the payload.*

New malicious versions appear weekly

The npm ecosystem is under sustained, automated attack.

Sources: Microsoft Security Blog (Dec 9, 2025) · Wiz · JFrog · The Hacker News (May 2026)

A more secure npm

After Shai-Hulud, GitHub committed to a tighter npm publishing model.

Mandatory 2FA

Phishing-resistant FIDO-based 2FA for publishing. TOTP being retired.

Granular tokens

Short-lived (7 days) and scoped. All classic tokens revoked December 2025.

Trusted publishing

OIDC for CI/CD — no long-lived secrets in workflows. OpenSSF best practice.

Sources: github.blog/security/supply-chain-security/our-plan-for-a-more-secure-npm-supply-chain/ (2025-09) · github.blog changelog 2025-09 / 2025-11 / 2025-12

Staged publishing: instant → reviewed

> *"Staged publishing: A new publication model that gives maintainers a review period before packages go live, with MFA-verified approval from package owners. This empowers teams to catch unintended changes before they reach downstream users — a capability the community has been requesting for years."*
>
> *— GitHub Blog, Strengthening supply chain security (Dec 23, 2025)*

Automated publishing got us speed. It also got us Shai-Hulud.

A review window — even a short one — is a structural answer to "how did a malicious version reach a million installs in ten minutes?"

Shipping soon.

Source: github.blog/security/supply-chain-security/strengthening-supply-chain-security-preparing-for-the-next-malware-campaign/ (Dec 23, 2025)

Actions: 2026 security roadmap

CI/CD is critical infrastructure. Three layers, all in public preview within 3–6 months.

Ecosystem

Workflow-level dependency locking (SHA-pinned, transitive). Immutable releases. Reviewable diffs for every Action update.

Attack surface

Policy-driven workflow execution (actor + event rules). Scoped secrets bound to specific workflows. Evaluate mode before enforce.

Infrastructure

Actions Data Stream for real-time CI/CD telemetry. Native egress firewall on hosted runners — monitor first, then enforce.

Source: github.blog/news-insights/product-news/whats-coming-to-our-github-actions-2026-security-roadmap/ (March 2026)

The cycle, in real time · May 19, 2026

> *"To prevent supply chain attacks following the pattern of Mini Shai Hulud, we rotated npm granular access tokens with write access that bypass 2FA.*
>
> *Update the stored token and rerun the workflow for your automations. Use npm Trusted Publishing to reduce reliance on such tokens."*
>
> *— @npmjs, May 19, 2026*

If you maintain a package and your automation just broke, that's why.

Community discussion + support: `github.com/orgs/community/discussions/196340`

Trusted Publishing docs: `docs.npmjs.com/trusted-publishers`

Sources: @npmjs on X (2026-05-19) · github.com/orgs/community/discussions/196340 — both verified 2026-05-20

And nobody is immune.

> *"Yesterday we detected and contained a compromise of an employee device involving a poisoned VS Code extension."*
>
> *"Our current assessment is that the activity involved exfiltration of GitHub-internal repositories only."*
>
> *— @github, May 20, 2026*

> *Critical secrets were rotated overnight. Log analysis and follow-on monitoring are ongoing.*
>
> *"We will publish a fuller report once the investigation is complete."*
>
> *Official updates:* `x.com/github`

Source: @github thread on X (May 20, 2026) — quoted directly. Refer to @github for the canonical, current statement.

No one is securing OSS alone.

Security has to fit the work — not fight it.

Where to start

Your CI/CD is privileged service code. Audit, review, and maintain it accordingly.

Secrets

- Publish from CI with trusted publishing (OIDC) — no long-lived token to leak
- If not: minimally-scoped tokens, shortest practical lifetime
- 2FA via passkeys / WebAuthn, not TOTP
- Set a rotation schedule for any secret that doesn't manage its own lifetime

CI/CD

- Pin CI/CD dependencies to immutable refs (commit SHA), not mutable tags
- Never run untrusted PR code with access to your secrets
- Run every CI job with the minimum permissions it needs
- Treat CI logs and artifacts as sensitive — they can leak secrets

References: GitHub Security Lab GitHub-Actions series (Parts 1–4) · github.blog npm supply chain plan (Sept 2025) · Microsoft Defender Shai-Hulud guidance (Dec 2025, updated May 2026)

Make it delicious.

Developer experience matters.

The Secure Code Game

10,000+

developers played across all seasons

4

seasons · Foundations → multi-stack → LLM → agentic AI

A free, open-source in-editor course. Exploit intentionally vulnerable code, then fix it. Training alone won't solve security — but if you must train, make it fun.

gh.io/scg

Source: github.blog (Secure Code Game tag; Season 4 post April 2026)

Make it social.

Open Source is a team sport.

Secure Open Source Fund

138

projects across 3 cohorts (38 countries)

219

maintainers funded and mentored

191

new CVEs issued by alumni projects

$1.38M

non-dilutive funding via GitHub Sponsors

Three weeks of mentorship and tooling. Starting with the OSS layers everyone else depends on.

Sources: github.blog/open-source/maintainers/securing-the-supply-chain-at-scale-starting-with-71-important-open-source-projects/ (Aug 2025) · github.blog/open-source/maintainers/securing-the-ai-software-supply-chain-security-results-across-67-open-source-projects/ (Feb 2026)

Pay it forward.

Help one dependency. Help the whole tree.

In summary

Make it easy

Make security the default path. Ship the tools, and use them ourselves first.

Make it delicious

Training helps. Structural change matters more.

Make it social

Invest resources in your dependencies. Supply-chain security is a team sport.

What should you do next?

For developers & maintainers

- Publish a `SECURITY.md` and enable Private Vulnerability Reporting — set how reports come in and what bar they must clear
- Enable code scanning and Dependabot
- Try the **Secure Code Game**
- Apply to the **Secure Open Source Fund**

For companies & teams

- Audit which OSS your business depends on
- Sponsor or contribute back to those projects
- Enable platform-native security features where possible (e.g. trusted publishing)
- Security is a baseline cost, not a tax

Thank you.

Questions?

References

- GitHub Security Lab
- GitHub Advisory Database
- CodeQL
- Octoverse 2024
- A year of OSS vulnerability trends (Mar 2026)
- Copilot Autofix · 3× faster
- How to scan Actions workflows with CodeQL
- CodeQL team uses AI
- Multi-Repository Variant Analysis
- Taskflow Agent · AI-supported triage (Jan 2026)
- Open-source AI framework for vuln scanning (Mar 2026)
- Our plan for a more secure npm supply chain (Sep 2025)
- SOSF · 71 projects (Aug 2025)
- SOSF · AI stack, 67 projects (Feb 2026)
- Secure Code Game · Season 4 (Apr 2026)
- seclab-taskflows · GitHubSecurityLab
- seclab-taskflow-agent · GitHubSecurityLab
- Coordinated disclosure (GitHub docs)
- Trusted publishing (OpenSSF)
- CVE-2025-24362 · CodeQL Action vuln (GHSA-vqf5-2xx6-9wfm)