DEVOPSDAYS CAIRO 2026

Your supply chain has an identity problem

Dan Maher — Senior Technical Advocate & Researcher



DATADOG

Why this matters

- A supply-chain incident is not (just) a tooling problem
- The risk surface is not new; the speed *is*
- Agents act at machine speed
- Sustainable DevOps means *safer by default*

whoami



Dan

Senior Technical Advocate & Researcher

Security research

Global core organiser, DevOpsDays

DATADOG

AI-powered observability and security

Ask me about the Tetris easter egg!

Many systems turn a trusted identity into authority too easily, too broadly, and for too long.

Execution context

- People, CI jobs and agents hold identities
- Dependencies run inside those contexts
- Malicious or merely flawed, the potential for harm is real

SHIFT ONE

Identity and access management

Identity and authority

01

The maintainer's account

Upstream, on a registry you do not control

02

The CI job

Publishing on behalf of the project

03

The laptop in front of you

Right now, in this room

OWASP top 10

#1

Broken Access Control

#3

Software Supply Chain Failures

INCIDENT 01 — SEPTEMBER 2025

The npm maintainer compromise

debug · chalk · and more

Adversary-in-the-middle phishing

- A reverse proxy on **`npmjs.help`**
- A plausible two-factor-reset email
- The maintainer signed in for real
- The attacker got the live session

Version management did the distribution

- 18+ packages, published from a real account
- Downstream, it looked like maintenance
- No suspicious link; no PR to approve

50%

of organisations pull a third-party library within a day of its release

*Datadog, 2026 State of DevSecOps

AiTM-resistant authentication

- FIDO2 keys and passkeys bind to a domain
- A passkey for **`npmjs.com`** cannot sign in to a proxy
- Defeats this vector of attack

Automated publishing

- A long-lived **`NPM_TOKEN`** in CI secrets
- The job can read it; so can an attacker
- Spoiler alert: there's no reason for a long-lived token here

Trusted Publishing

- From the PyPI community, 2023
- CI presents an OIDC identity token
- The registry mints a token for that release
- Nothing standing left to steal

INCIDENT 02 – SEPTEMBER 2025

Shai-Hulud

a self-propagating npm worm

Shai-Hulud was fundamentally lazy

- Found publish tokens on infected machines
- Backdoored every accessible package
- A later variant took cloud credentials too

SURPRISE!

The developer laptop is now part of production.

What the laptop is holding

- Registry tokens, cloud credentials, SSH keys
- Broader access than most production servers
- It can trigger a production change

SCOPE AND TIME

Decide which actors may act, and make their authority temporary and specific.

SHIFT TWO

Verify continuously

A checksum proves integrity, not legitimacy

- It proves the download matches the registry
- A poisoned release matches just as well
- Nobody vouched for the contents

Quarantine a new release, then check it against known-bad

- Pinning works until you want to upgrade
- Quarantine new versions, check **`osv.dev`**
- No mirror? Use **`scfw`** at the package manager
- Catches what the community already found

Pin your GitHub Actions to a commit hash, not a version tag

4%

of organisations pin every public GitHub Action

*Datadog, 2026 State of DevSecOps

- A tag can start pointing at new code
- A commit hash cannot move
- You're probably vulnerable right now

Provenance: where did this actually come from?

SLSA

Supply-chain Levels for Software Artifacts

Checkable metadata showing which source repository and build pipeline produced an artefact.

`git blame` , but for the finished thing you installed.

Sigstore

Signing, without a key to guard forever

Signs the artefact with a short-lived certificate tied to the CI identity that built it.

Not a private key someone has to protect indefinitely.

A malicious release can look legitimate

- Checksum matches
- Not in a threat database (yet)
- Built via a legitimate pipeline
- You probably pulled it already

INCIDENT 03

SANDWORM_MODE

a new target: the AI coding environment

The attack was a tool description

- Local MCP server, wired into your editor
- Harmless names like **`lint_check`**, **`scan_dependencies`**
- Prompt injection in the description
- Looks legit—the linter did lint!

Persistence and the limits of static matching

- Malicious Git hooks in local repositories
- And in Git's global **`init.templateDir`**
- A dormant code-rewrite component, disabled
- Changing code defeats fingerprints

Three controls, three different questions

Provenance

Where did this come from?

Threat database

Has someone already flagged it?

Runtime

Is it doing the job we expected — *and nothing else?*

A tool called `lint_check` should not read your SSH keys.

INCIDENT 04 — MARCH 2026

LiteLLM

two malicious releases, quarantined in ~40 minutes

The pipeline handed the attacker the key

- The attacker compromised Trivy first
- LiteLLM ran Trivy in CI
- The action took the PyPI credential
- No social engineering. No guessed password
- 2,500+ organisations potentially exposed

Same scenario, different machine

- Trusted authority, outside its intended job
- An immutable pin closes that entry path
- Runtime gives you a second chance
- A tool call deserves the same scrutiny

REPEATED FOREVER

Validate *what* enters the system, *where* it came from, and whether it behaves *within* the limits you set.

SHIFT THREE

Guardrails that scale

Guardrails enforce the safety decisions that humans have already made.

INCIDENT 05 — MARCH 2026

Axios

70 million weekly downloads · live for ~3 hours

The simplest defence: turn off post-install hooks

- A script npm runs automatically on install
- Axios used it. Shai-Hulud used it first
- No need to know the attacker or the payload
- One less thing for people to worry about

Give a dependency update an observation window

8/10

high-profile supply-chain attacks had exploitation windows shorter than a week

*2025 analysis of ten high-profile attacks

- You do not ship everywhere at once
- Make the window a week by default
- Time for a bad release to be pulled

You have a coding agent? Put it in a sandbox

- Nono: a kernel-enforced micro-sandbox
- **gh** gets a phantom credential, not your token
- SSH keys are off the filesystem entirely
- The agent cannot talk past the kernel

Automate the routine

- Block the build, open the fix, quarantine
- Route real judgment to a person
- People set the boundaries and the exceptions
- The system enforces them every time

Humans set the boundaries. Systems enforce them.

Incident round-up

`npm maintainer`	Lookalike login page (AiTM)
`Shai-Hulud`	Long-lived standing credentials
`SANDWORM_MODE`	Poisoned tool descriptions
`LiteLLM`	Compromised security scanner
`Axios`	Maintainer's laptop as launchpad

An actor operating with trusted authority, crossing a boundary it should not have been able to.

That is not five problems. It is one problem, five times.

Three questions to take home

01

Authority

Phishing-resistant authentication, and temporary, specific credentials.

02

Verification

Provenance, known-bad checks, and runtime policy.

03

Guardrails

Block unnecessary mechanisms, create observation windows, and sandbox agent access.

We are who society has implicitly agreed to trust with this

We need to do a better job of earning it

Thank you

Dan Maher — Senior Technical Advocate & Researcher



DATADOG