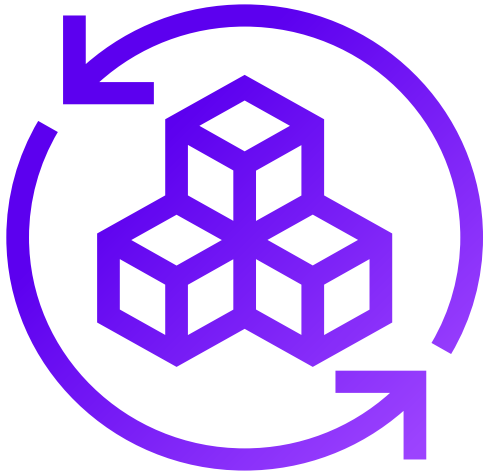
AWS + Datadog

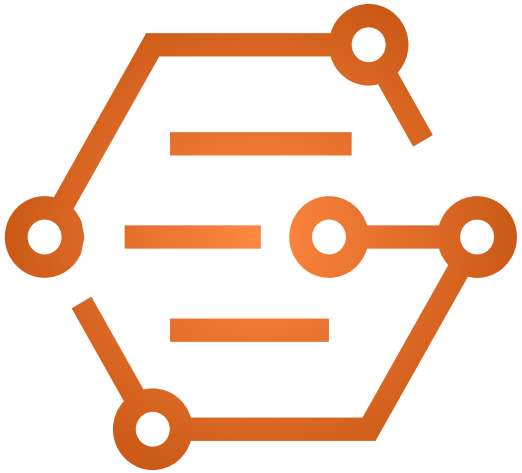
Securing the blind spot in your pipeline

Daniel « *phrawzty* » Maher
Senior Security Advocate & Researcher



Do you really know what's in your stack?





What the data tell us

	npm	PyPI	RubyGems	crates.io	Hex.pm	Packagist	*Max*
Auth (Human)	3	3	0	1	0	0	*4*
Auth (Machine)	6	4	4	4	2	2	*7*
Access Control	6	4	4	4	5	4	*7*
Build & Provenance	6	4	3	2	2	2	*8*
Integrity & Signing	5	5	6	2	3	2	*6*
Vulnerability Mgmt	7	3	4	3	4	4	*7*
Abuse Prevention	6	5	5	5	5	2	*6*
Transparency	5	4	5	3	3	2	*7*

The gaps that matter

- **Authentication**
  - optional controls are not actually controls
- **Provenance**
  - no guarantee that the package actually comes from the source
- **Build**
  - none of the registries enforce isolated build environments

The gaps that matter

- **Authentication**
  - optional controls are not actually controls
- **Provenance**
  - no guarantee that the package actually comes from the source
- **Build**
  - none of the registries enforce isolated build environments

Registries are not your security team. They are infrastructure.

AI amplifies the problem

- AI Assistants add dependencies automatically
- The last point of human friction has now disappeared
- What was already largely invisible is now *even faster*

2 in the morning…

a maintainer is compromised

---

a malicious package is published

---

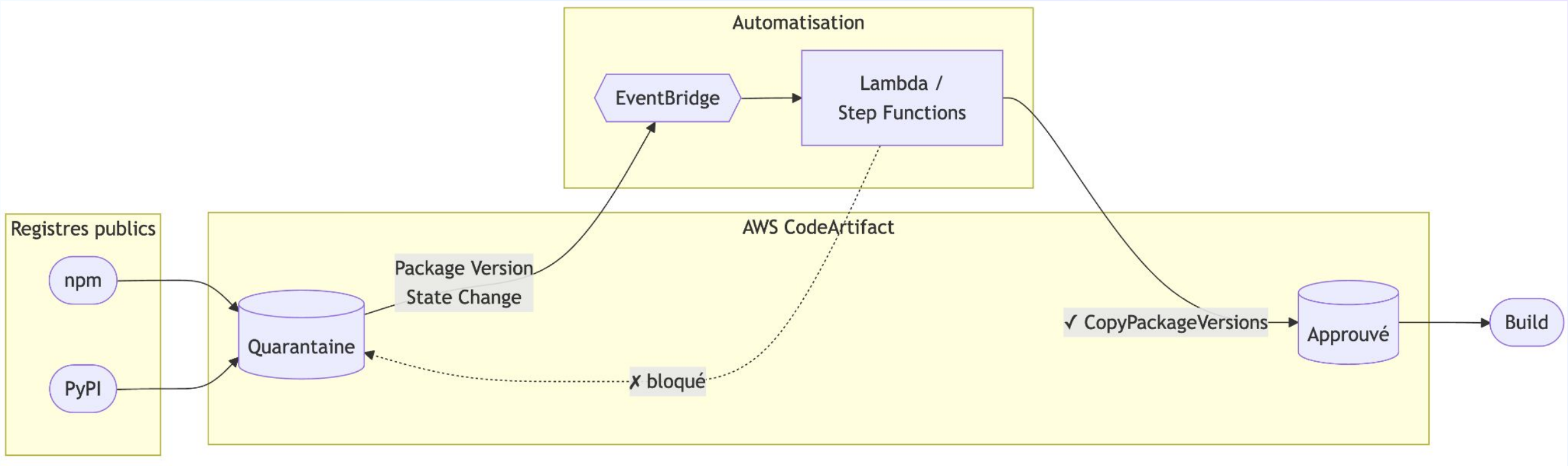
an automated « `latest` » build runs



AWS CodeArtifact: the preventive control

- Managed artefact repository service
- Compatible with npm, pip, Maven, NuGet, etc…
- Your tools, your workflow
- **You decide what gets in**

Control architecture





Three barriers

- **Architectural isolation**
  - your builds never talk to public registries
- **Automated validation**
  - every new version is scanned before a decision is made
- **Explicit promotion**
  - by default, nothing gets through (spoiler : *version pinning*)



Three barriers

- **Architectural isolation**
  - your builds never talk to public registries
- **Automated validation**
  - every new version is scanned before a decision is made
- **Explicit promotion**
  - by default, nothing gets through (spoiler : *version pinning*)

Datadog Code Security

SCA

Software Composition Analysis

SAST

Static Code Analysis

Secret Scanning

(just that)

IAST

Runtime Code Analysis

Datadog Code Security

SCA

Software Composition Analysis

SAST

Static Code Analysis

Secret Scanning

(que ça)

IAST

Runtime Code Analysis

6 months later...

malicious package published (oops)

---

vulnerability in prod (OOPS)

---

time to panic ?!

Six months later

- **SCA « active »** ***(Software Composition Analysis)***
  - the vulnerable package is detected (service, version, CVE, etc.)

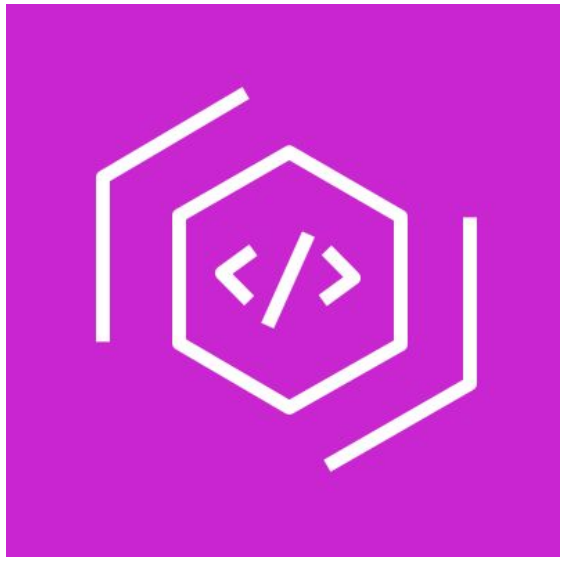
- **IAST** ***(Interactive Application Security Testing)***
  - the exec path is detected and the impact is confirmed

Preventive vs. Detective

- **Preventive** : block *before* it gets in
  - CodeArtifact, SAST, SCA (static mode)

- **Detective** : know what's already there — *and act*
  - SCA (active mode), IAST

AWS et Datadog together = *defence in depth*





Where to start?

- **Map** your dependency graph — know what you're building on
- **Identify** your source registries
- **Establish** a preventive control layer
- **Ensure** full production visibility

Where to start?

- **Map** your dependency graph — know what you're building on
- **Identify** your source registries
- **Establish** a preventive control layer
- **Ensure** full production visibility

These are not four projects over six months. These are four steps you can start this very week!

Merci !

AWS CodeArtifact

`https://aws.amazon.com/codeartifact`

Datadog Code Security

`https://docs.datadoghq.com/security/code_security`

Our super interesting blog 😁

`https://securitylabs.datadoghq.com`