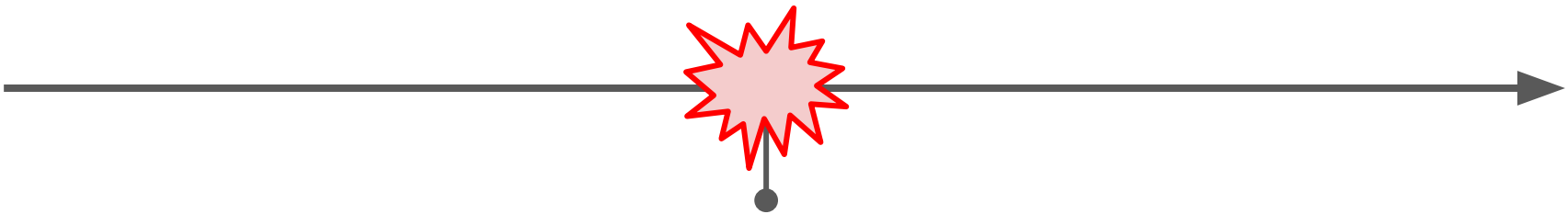


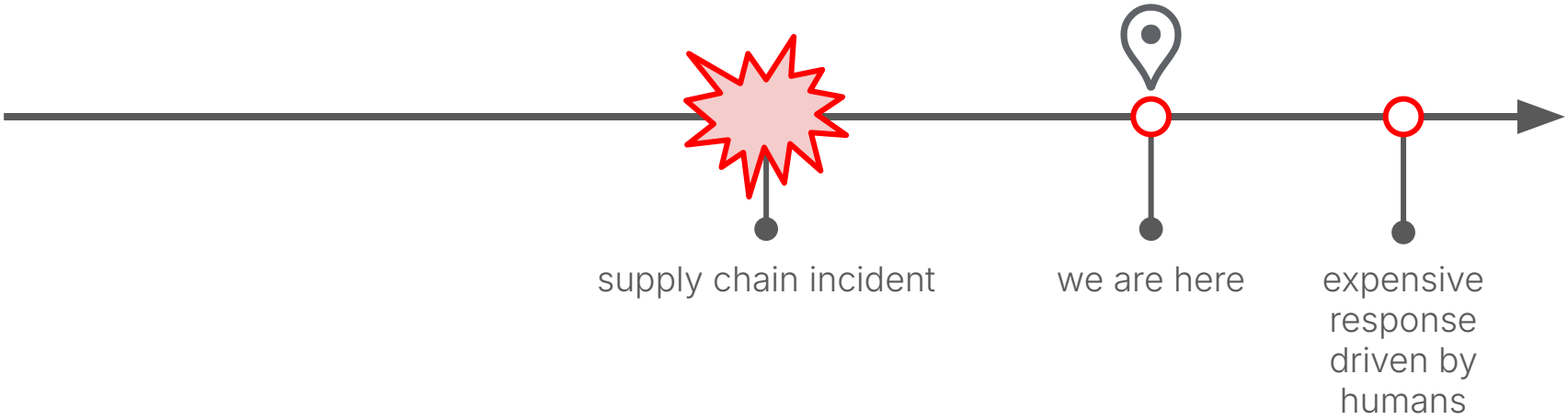
Building Proactive and Reactive Defenses for Google's OSS Supply Chain

Bob Callaway, PhD
bcallaway@google.com

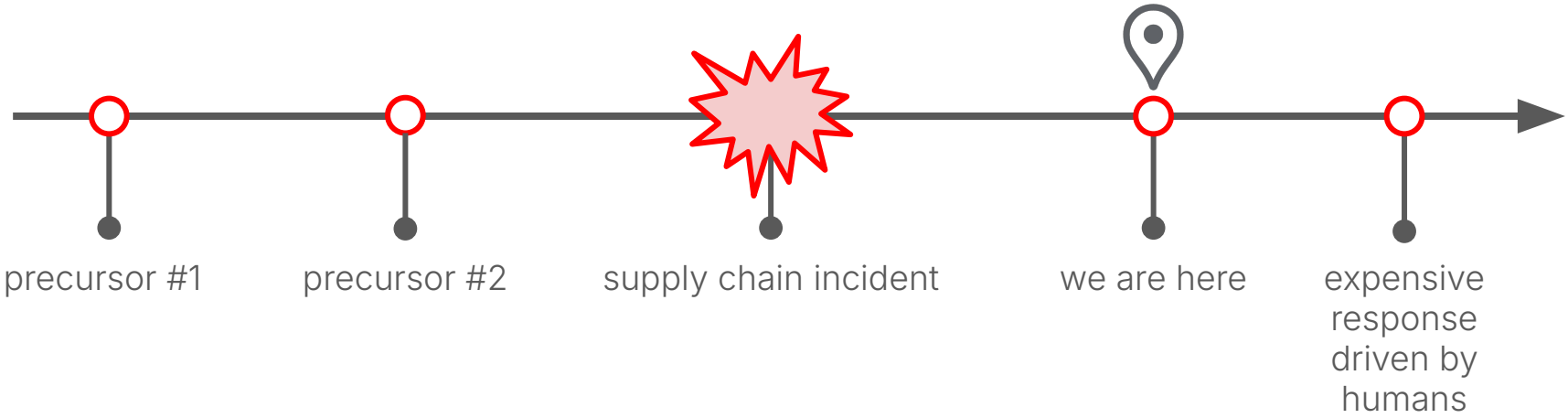
Supply Chain Incident Timeline



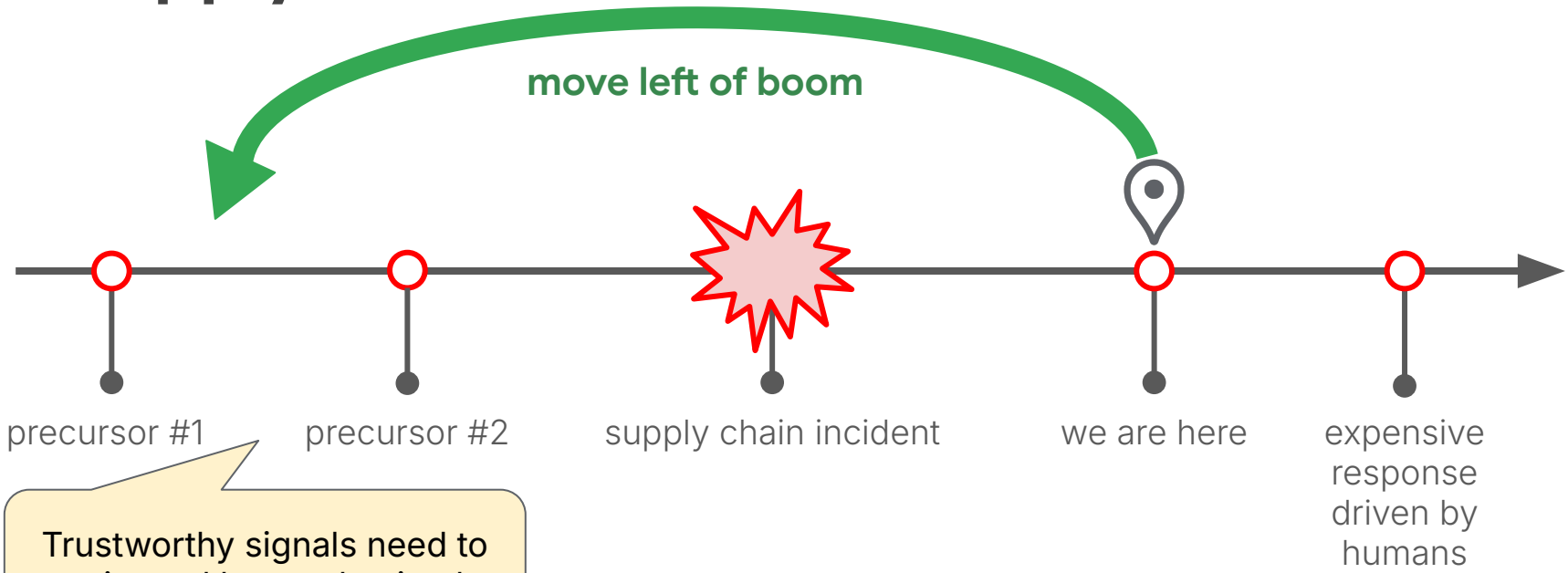
Supply Chain Incident Timeline



Supply Chain Incident Timeline



Supply Chain Incident Timeline



Moving left of boom (ie: think proactively)

An incident is the materialization of a supply chain risk.

Precursors indicate posture, and allow you to measure **potential** materialization of risk in the future.

- Precursors do not guarantee materialization risk.

Automating actions based on precursors enables substantial risk reduction, with a very low friction cost for false positives.

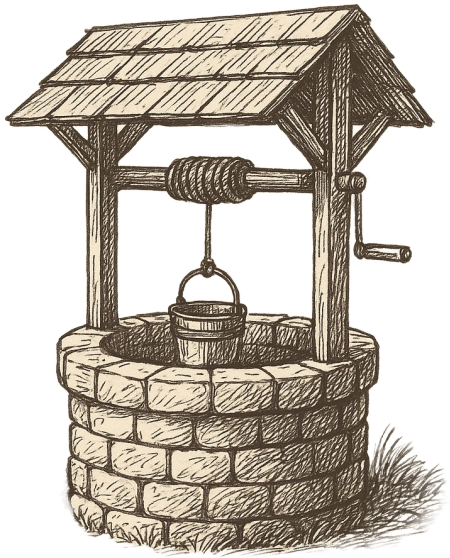
Opportunity and tragedy of the open source commons

Google is deeply reliant on open source software (OSS)

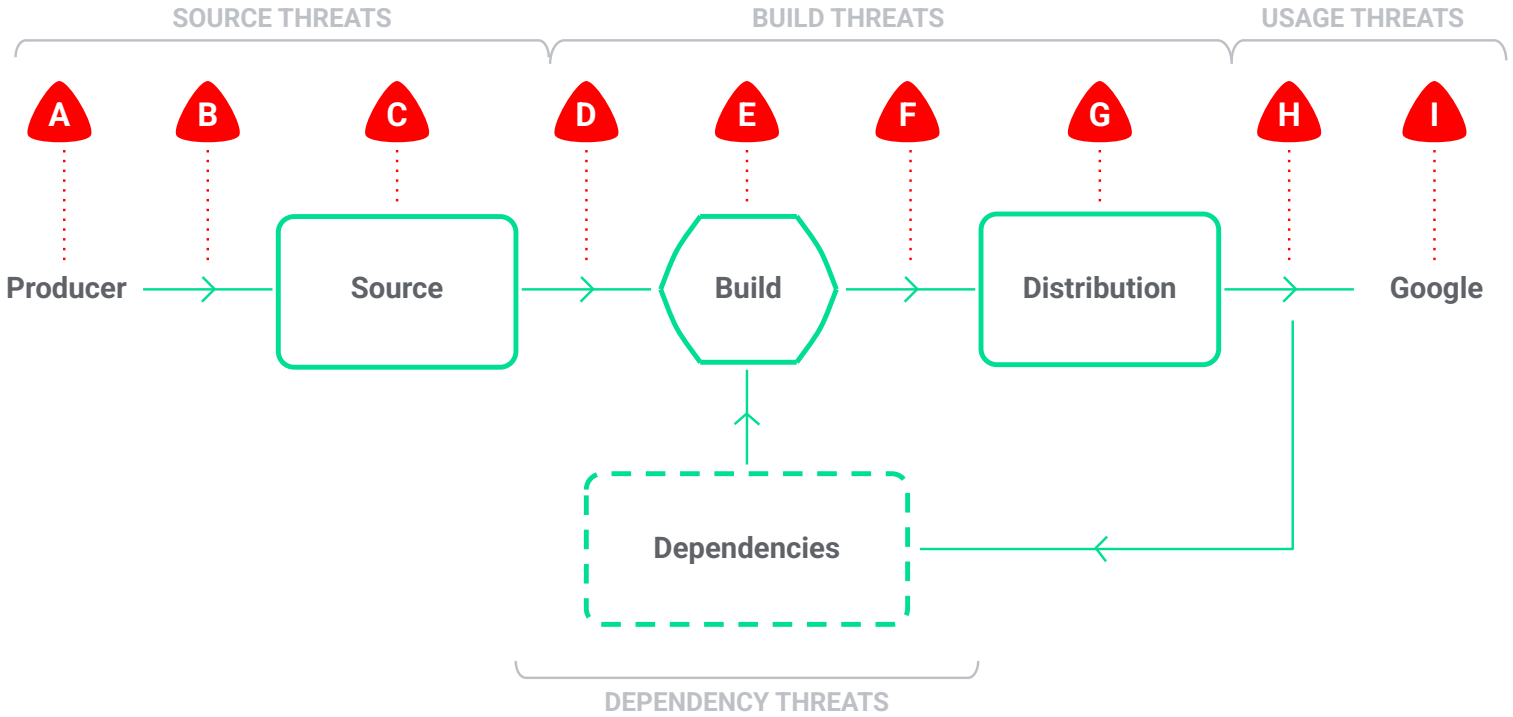
- Innovation catalyst, essential infrastructure, and standards engine
- However, open source comes with significant latent security risks

It's infeasible for us to secure just what *we care about today*

- It's an O(N) problem (or worse!)
- We need sustainable, proactive, and secure-by-default approaches for our most critical ecosystems
- We also need timely and thorough reactive remediation capabilities



Goal: Identify, build and embed `O(1)` or `O(log N)` solutions into open source ecosystems and platforms to **durably and scalably improve security outcomes**



A Producer / **Community**
B Authoring & reviewing
C Source code management

D External build parameters
E Build process
F Artifact production

G Distribution channel
H Package selection
I Usage

The Ecosystem Security Ratchet

The “ratchet” is an intentional sequence of step improvements to workflows, package managers, and package repositories — progressively improving security in a sustainable way.

In the next three years we expect the final stages in the ratchet to be widely adopted across languages.

The primary trade off is that much of this can only be done via influence.

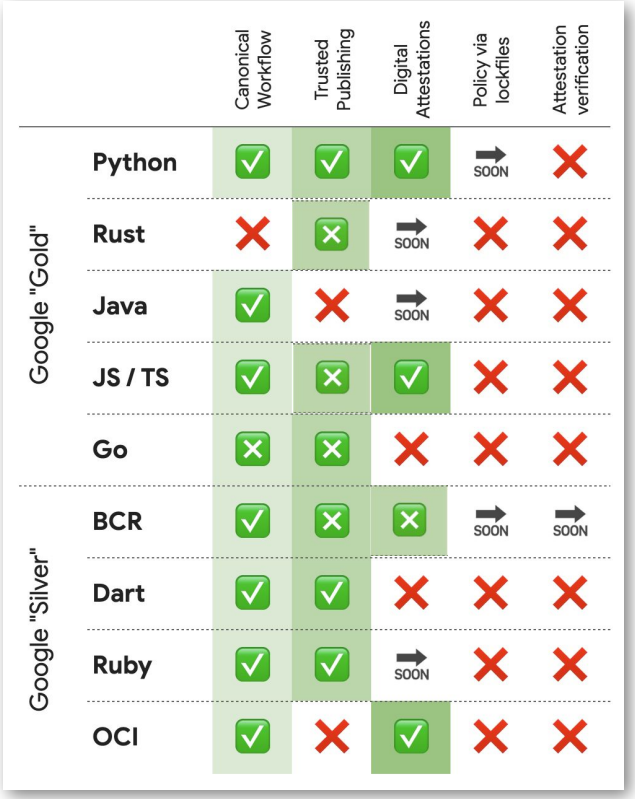
The Ecosystem Security Ratchet: Adoption Status - Q4 2025

GOSST has been directly involved in the adoption of Ecosystem Security Ratchet elements in all key open source ecosystems.

As a result, Google is now able to consume and verify attestations when ingesting third-party open source for many ecosystems.

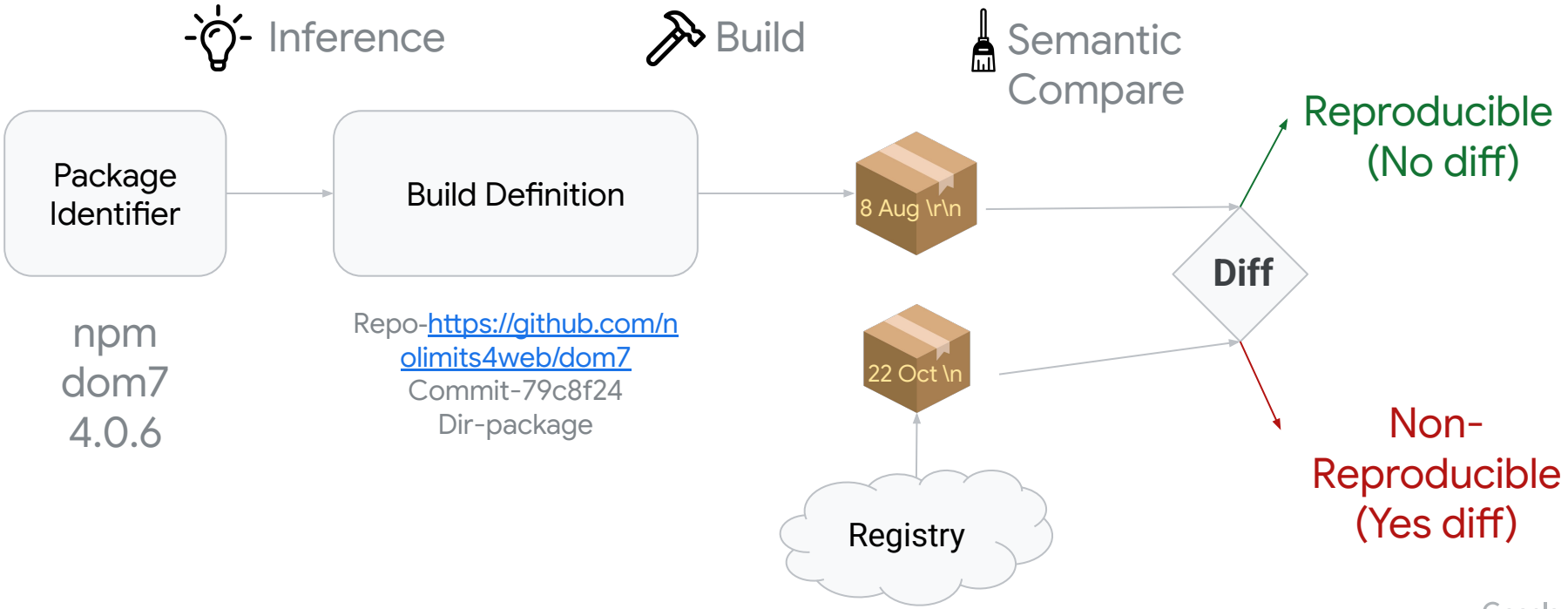
LATEST

Sept 16th: nuget announced upcoming support for Trusted Publishing



OSS Rebuild: https://oss-rebuild.dev

Rebuilder infrastructure for applying reproducible builds at scale AND MORE!



Upstream Data

Package metadata
Timestamps, versions...

Associated repository

Popularity:
Forks, Stars, Downloads...

Package criticality
Importance to ecosystem

Domain tracking

Internal Threat Intel
Google policy: banned...

Public security findings

Build anomalies

SAST/DAST reports

Synthesized Findings

Actuation Points

Artifact registries (internal)

CI checks

Code Review Agents

Runtime scanners

SCM systems

SCM automation

CD systems

Proactive Protection from Malicious Upstream

Findings are designed to automatically reduce risk.

Signals and actuation points actively promote & enforce security best practice without pushing additional work or security decisions onto individual engineers.

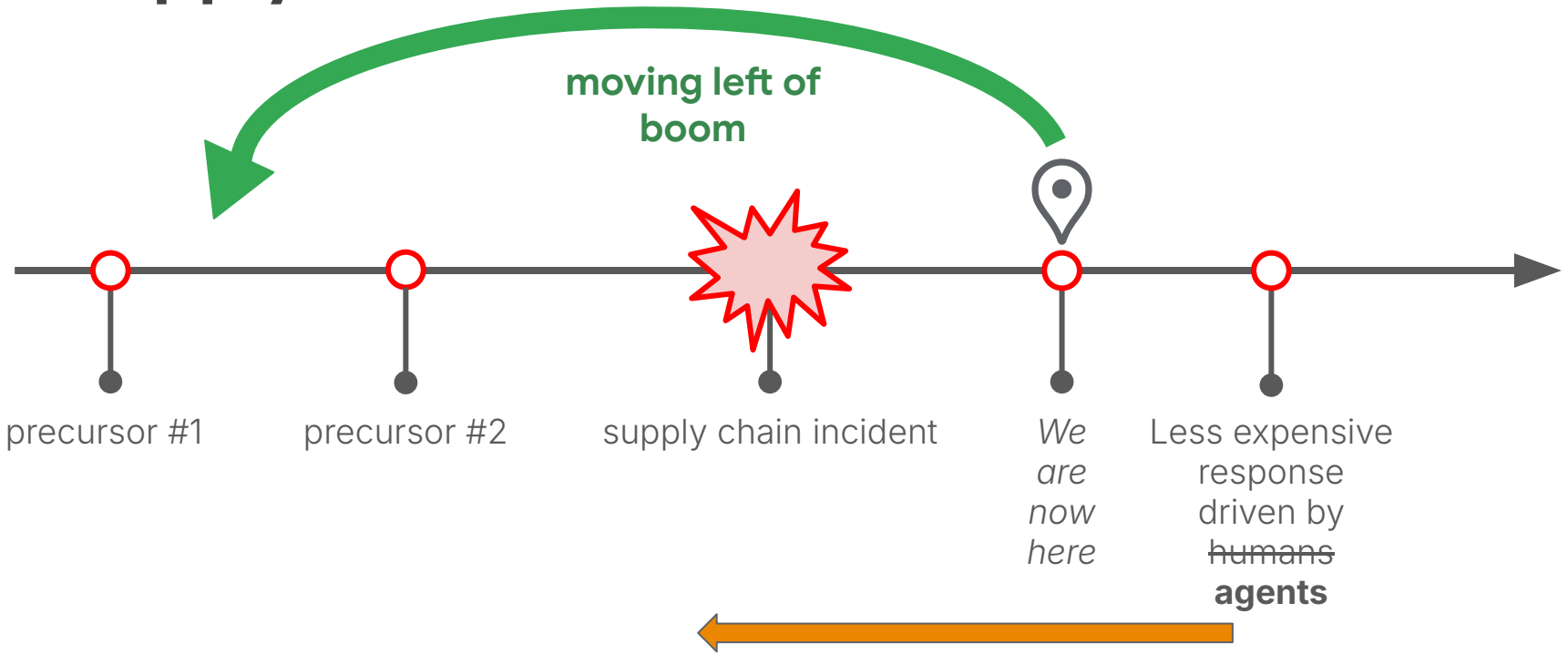
Supply chain risks are taken off the table *before* they become threats

Teams using these signals through Airlock were **automatically protected from recent attacks**, including the npm Shai-Hulud worm

The Junk Package detection identified over 100,000 malicious and low quality npm packages that were part of a coordinated tea[.]xyz attack, meaning signal **consumers are protected even before security researchers identify threats**.

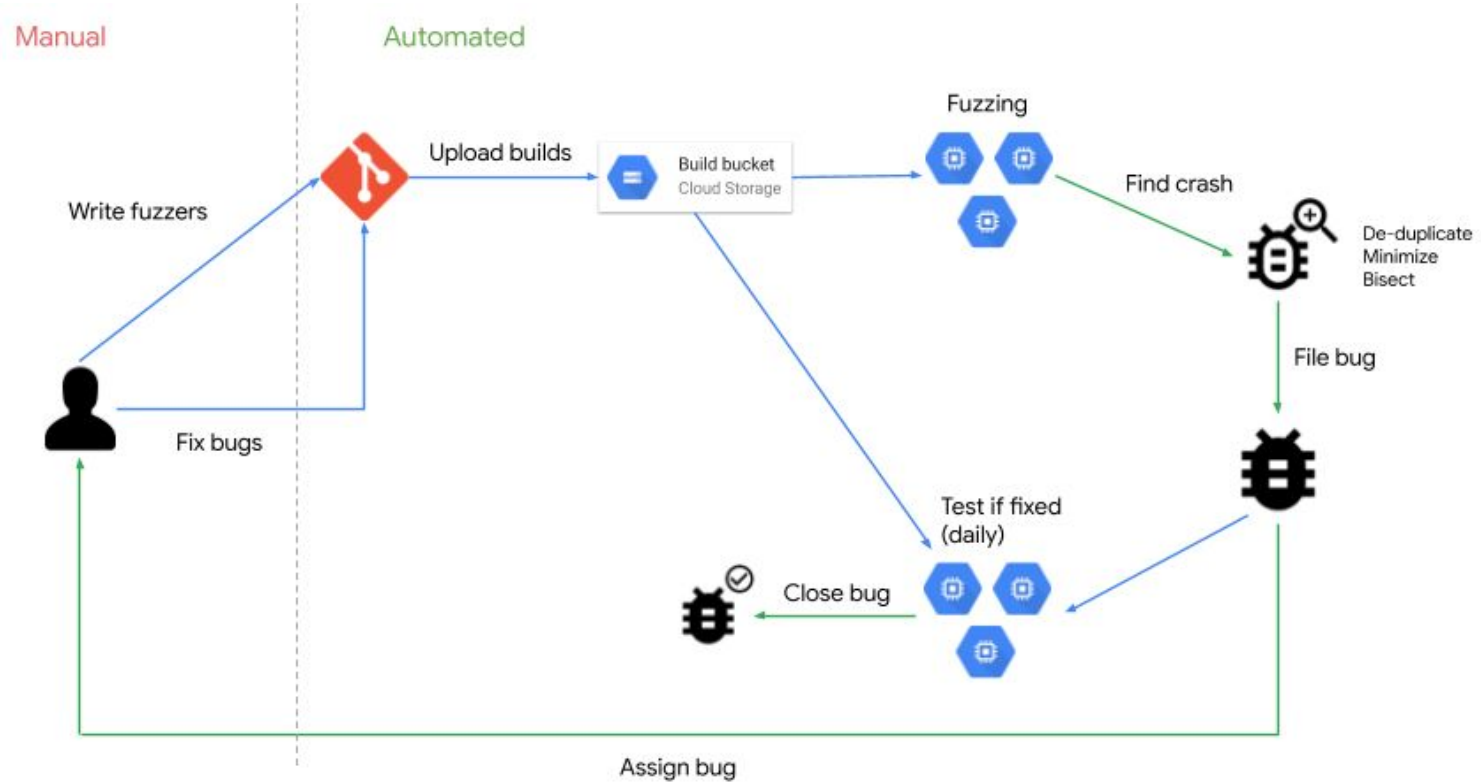
Soak Time is industry best practice to delay new, and therefore risky, packages and versions. But delaying security vulnerability fixes causes Vuln SLO issues. Dynamic Soak Time enables us to **expedite critical security fixes while still providing risk reduction**.

Supply Chain Incident Timeline v2

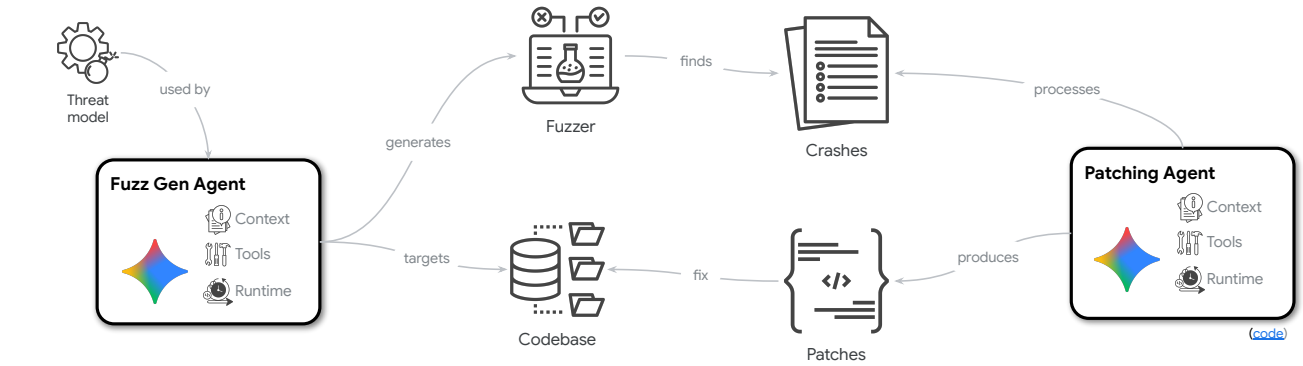


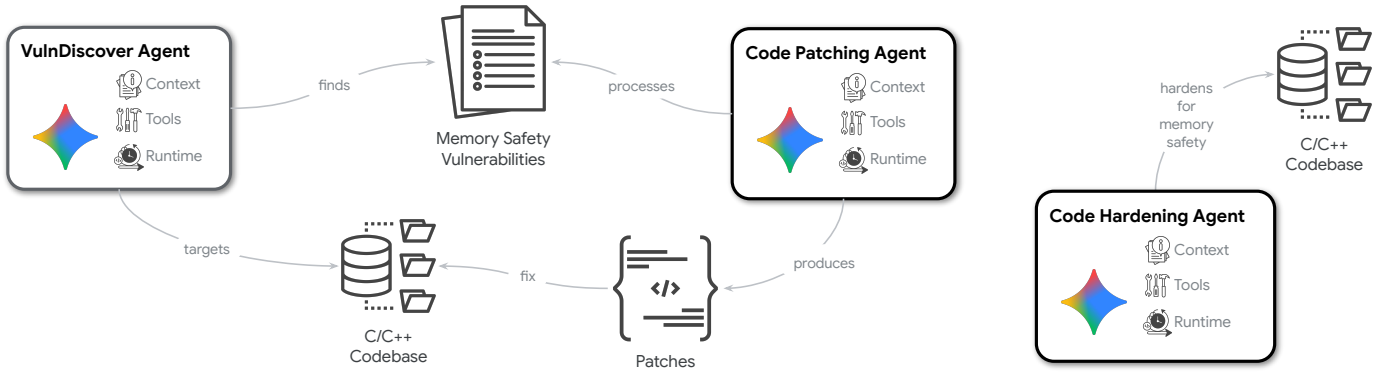

SECURE
SOFTWARE
JUST ADD
AI

OSS Fuzz: Finding vulnerabilities in OSS software

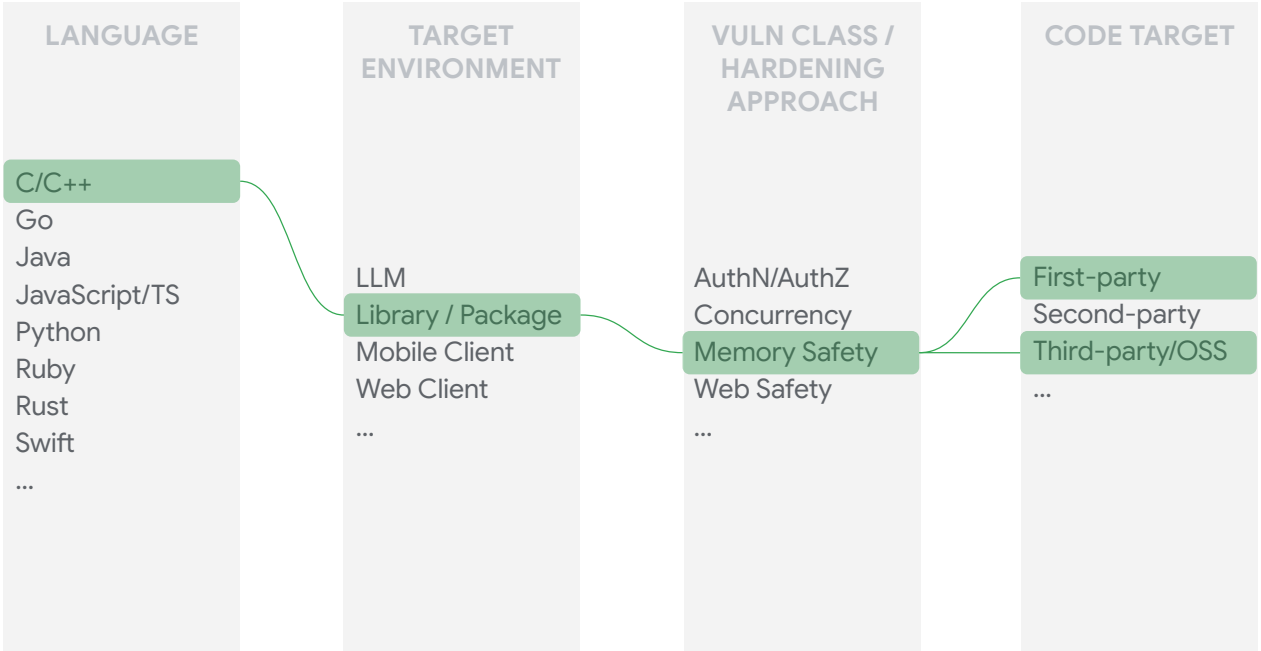


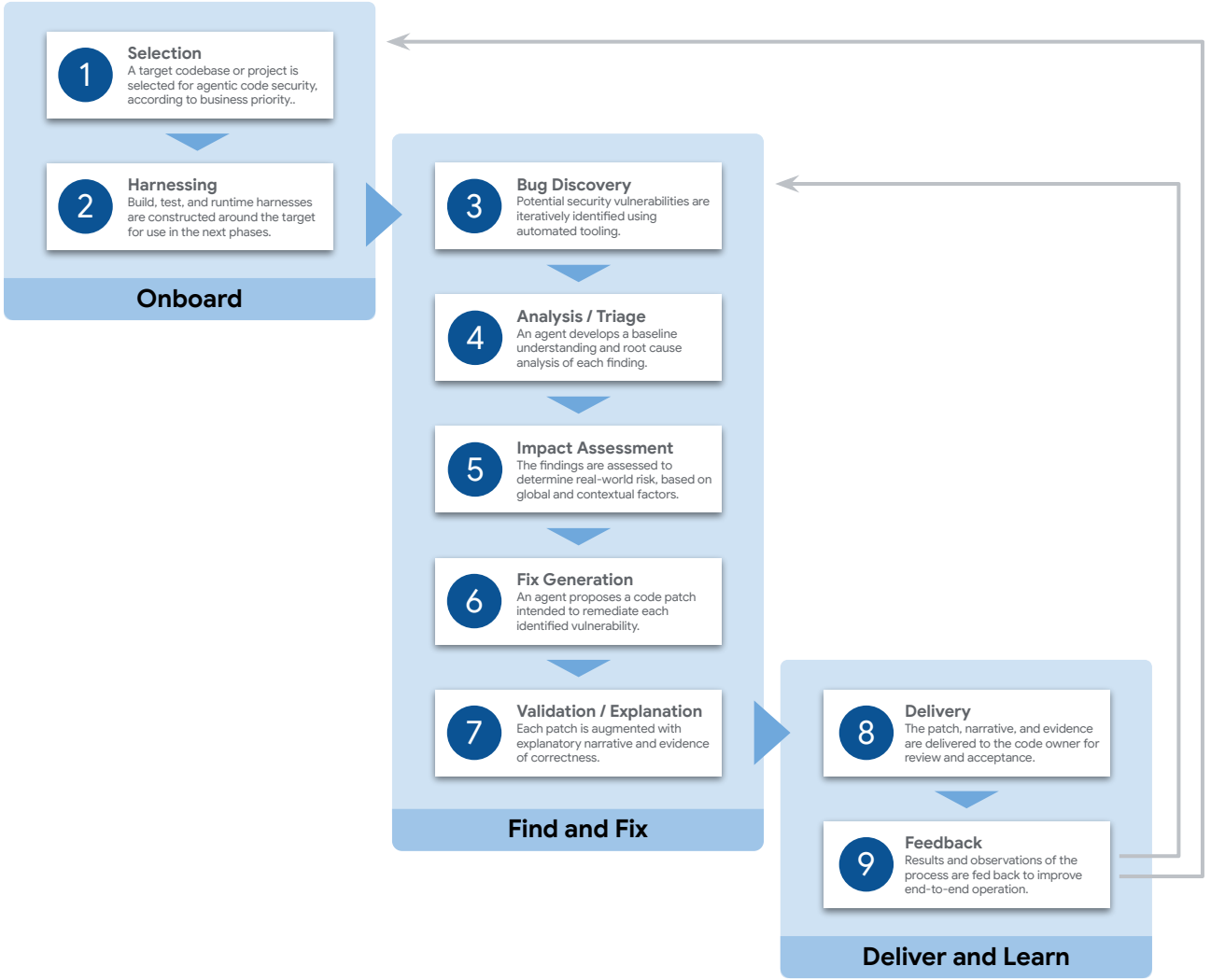
Agentic Issue Detection & Repair





Agentic Issue Detection & Repair — **Expansion Dimensions**




1
Selection
A target codebase or project is selected for agentic code security, according to business priority..
2
Harnessing
Build, test, and runtime harnesses are constructed around the target for use in the next phases.
Onboard
3
Bug Discovery
Potential security vulnerabilities are iteratively identified using automated tooling.
4
Analysis / Triage
An agent develops a baseline understanding and root cause analysis of each finding.
5
Impact Assessment
The findings are assessed to determine real-world risk, based on global and contextual factors.
6
Fix Generation
An agent proposes a code patch intended to remediate each identified vulnerability.
7
Validation / Explanation
Each patch is augmented with explanatory narrative and evidence of correctness.
Find and Fix
8
Delivery
The patch, narrative, and evidence are delivered to the code owner for review and acceptance.
9
Feedback
Results and observations of the process are fed back to improve end-to-end operation.
Deliver and Learn

Enabling safer outcomes for our users and OSS ecosystems