Google

The Consequence of Success: OSS is Critical Infrastructure

Eric Brewer
Google

June 21, 2022
Open Source Summit North America

Open Source is Hugely Successful

The benefits of modern open source sharing are enormous:

- High velocity of changes, thus higher innovation
- Extensive software ***reuse*** (the shoulders of giants)

But now we depend on it world wide… it has become critical to everyday life

Congratulations!

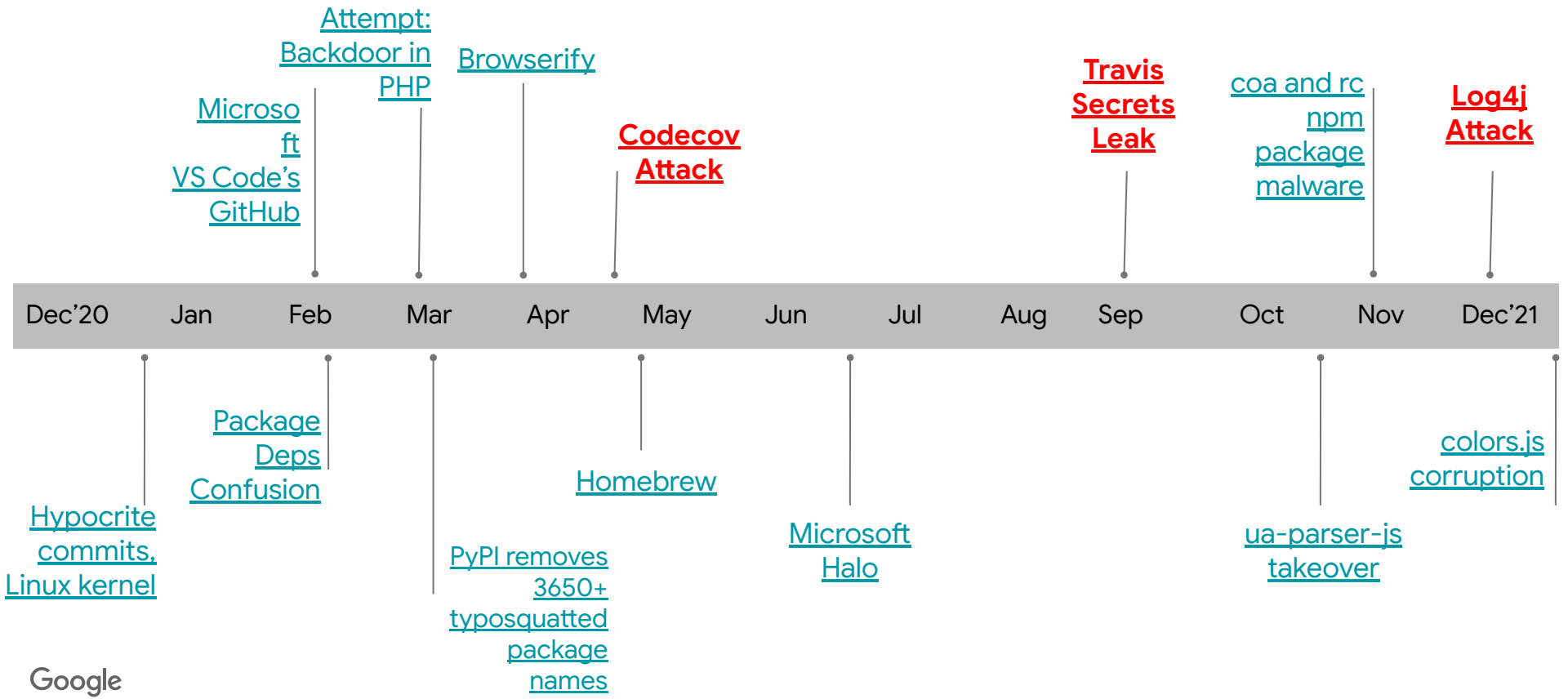
We need to be worthy of that trust

Electrical grids, water supplies, public transit, oil pipelines, connected devices…

The world ***needs*** to solve this problem

- … in a way that fits with modern tooling and development practices
- … in a way that works for maintainers

Open Source incidents are real and costly



Natural outcome: top down pressure to “fix it”

Governments are the most obvious source of pressure

- Biden Executive Order on Cybersecurity
- Regulations like FedRAMP (e.g. no vulnerabilities in VMs or containers)
- But EU, UK, also and eventually most nations...
- As a citizen, you should *want* them to care

Brands have to care too: Equifax, Sony, many others

Log4j Legal Request

Daniel 🇸🇪 Stenberg ✓
@bagder

If you are a multi billion dollar company and are concerned about log4j, why not just email OSS authors you never paid anything and demand a response for free within 24 hours with lots of info? (company name redacted for *my* peace of mind)

Dear Haxx Team Partner,

You are receiving this message because [redacted] uses a product you developed. We request you review and respond within 24 hours of receiving this email. If you are not the right person, please forward this message to the appropriate contact.

As you may already be aware, a newly discovered zero-day vulnerability is currently impacting Java logging library Apache Log4j globally, potentially allowing attackers to gain full control of affected servers.

The security and protection of our customers' confidential information is our top priority. As a key partner in serving our customers, we need to understand your risk and mitigation plans for this vulnerability.

Please respond to the following questions using the template provided below.

1. If you utilize a Java logging library for any of your applications, what Log4j versions are running?
2. Have there been any confirmed security incidents to your company?
3. If yes, what applications, products, services, and associated versions are impacted?
4. Were any [redacted] product and services impacted?
5. Has [redacted] non-public or personal information been affected?
6. If yes, please provide details of affected [redacted] information immediately.
7. What is the timeline (MM/DD/YY) for completing remediation? List the steps, including dates for each.
8. What action is required from [redacted] to complete this remediation?

In an effort to maintain the integrity of this inquiry, we request that you do not share information relating to [redacted] outside of your company and to keep this request to pertinent personnel only.

Thank you in advance for your prompt attention to this inquiry and your partnership!

Sincerely,

[redacted] Information Security

Log4j Legal Request (thanks to Daniel Stenberg, @bagder)

In the letter:

- We use your thing that uses log4j, so we need a response in 24 hours
  - Have you had any incidents, what was affected?
- **What is your remediation plan? ... with steps and dates**

This company never paid the maintainers, before or during.

This is a misunderstanding of how OSS works...

Open Source is “free” but “as is”

Is **free, easy to use,** BUT, **it says “as is” all over the place!**

- Developed by **resource-constrained developers**
- Limited investment for **ongoing maintenance**
  - Security bug fixing, dependency updates, incident response, etc
- High variation for testing, secure processes

Can be secure... Many eyes make all bugs shallow**, but only if they are looking**

- 30% of packages have 1 maintainer, in practice many have zero

Two roles: Distribution vs Accountability

Package Manages deliver on *distribution*:

- Great for maximizing the number of packages available
- But “as is” with no promises

(some) “distros” do limited distribution, but with some accountability

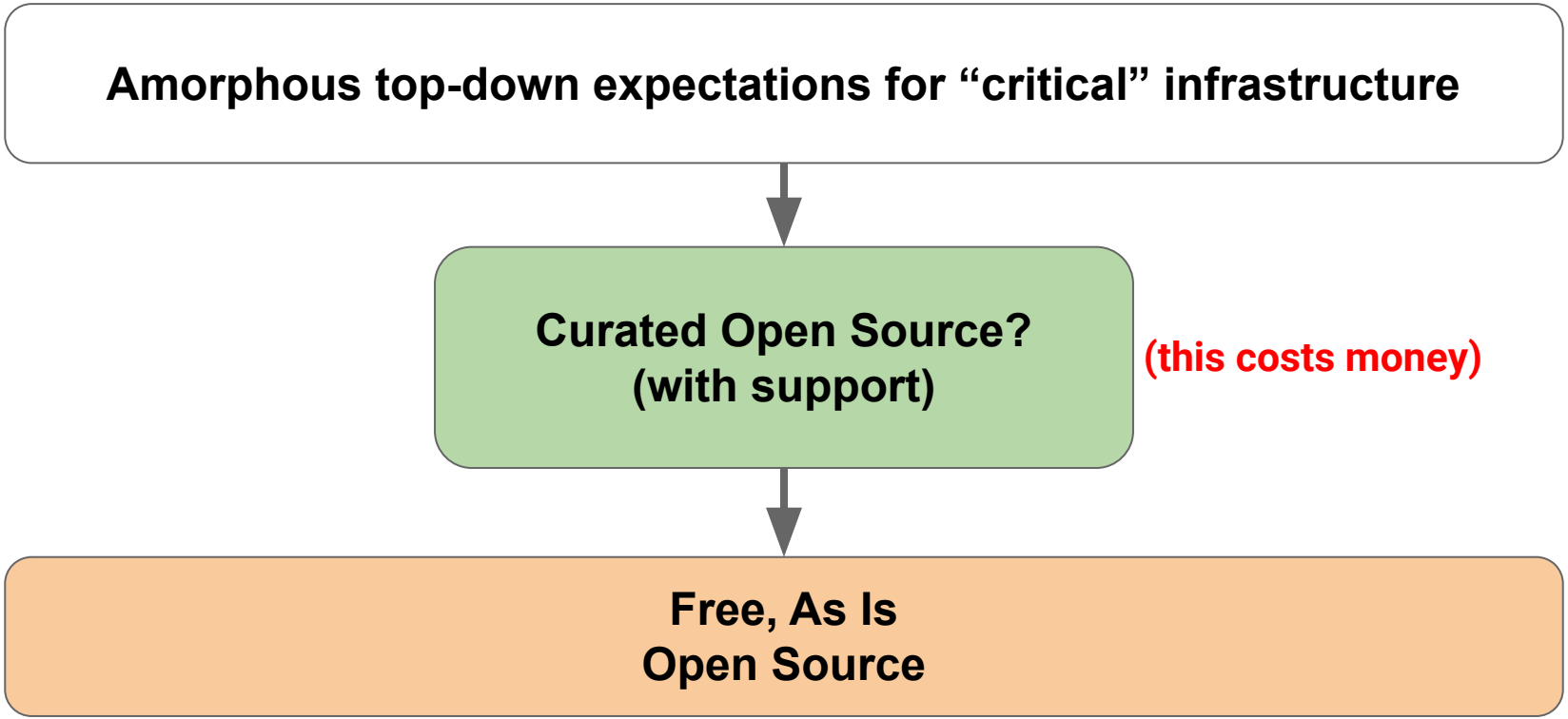
- Debian: packages work together with some validation
- Red Hat: packages with long-term support and security fixes

There are two kinds of roles:

- **Distribution**: maximize availability of AS IS software
- **Accountability**: supported, more secure versions, perhaps for a fee

(A project or even a package manager could choose to do both roles)

Curation: Adding a Layer of Accountability



Curation: Open Source with “Support”

Focus on **fixing vulnerabilities**, not just finding them[1]

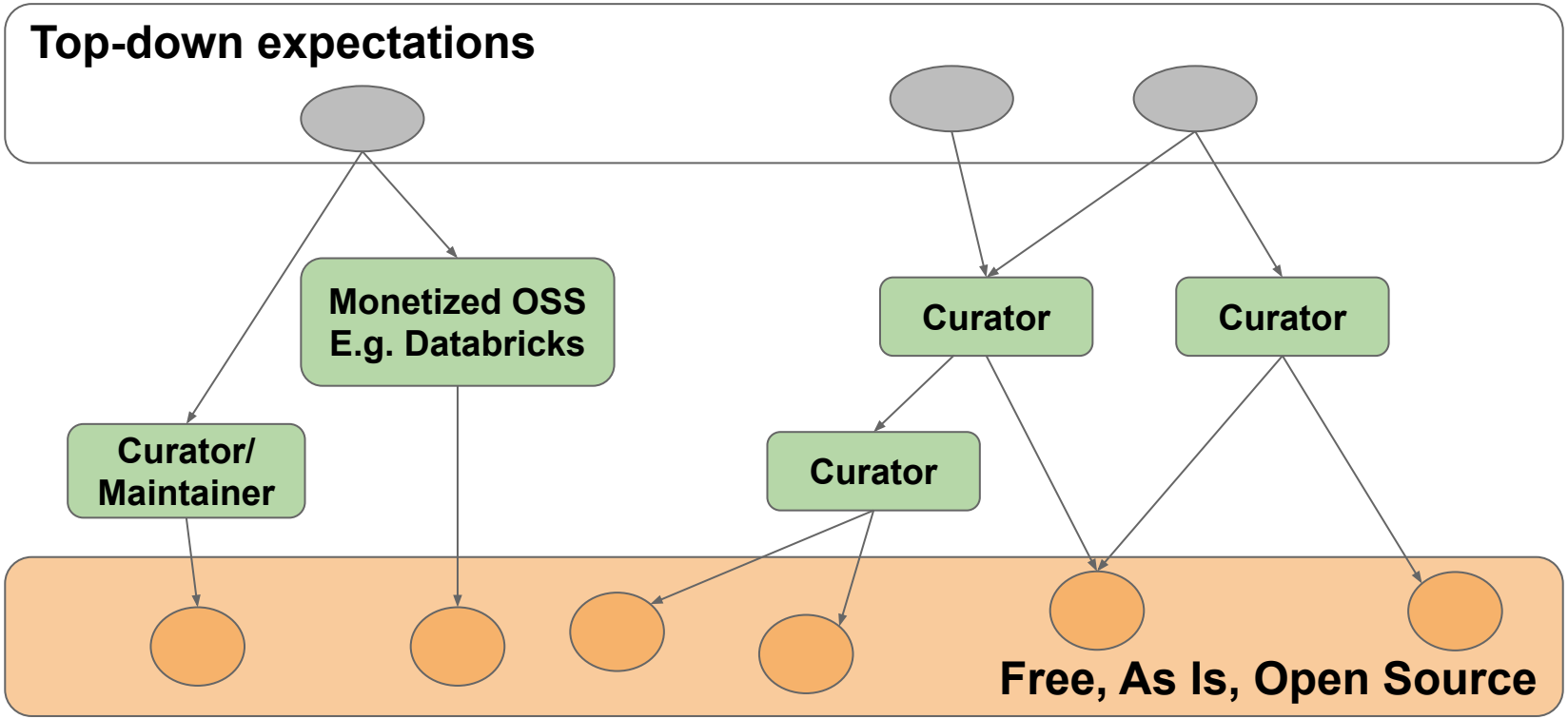
- Updating **old, vulnerable dependencies** (esp in widely used software)
- A**utomation** for testing
- Tracking dependencies

Service level agreement for responsiveness

Provide some liability protection? Reduced cyber-insurance rates?

1. https://github.com/google/oss-fuzz#trophies

Curation: Adding a Layer of Accountability



Google Cloud *Assured OSS* (one example of curation)

Key idea: make available (for fee) packages that we are already curating.

Supported versions of packages that Google already uses:

- are regularly scanned, analyzed, and fuzz-tested for vulnerabilities
- corresponding enriched metadata
- are built with Cloud Build including evidence of verifiable SLSA level
- are verifiably signed by Google
- are distributed from an Artifact Registry secured and protected by Google

Avoid forks: upstream the fixes

What should *maintainers* want out of Curators?

- Make contributions, not requests (or demands)
- Submit “good” pull requests (PRs):
  - Actually work and pass tests
  - Meet various guidelines
  - Easy to review
- Add test cases (helps everyone)
- Share the wealth?
  - At least opex if not funding for people
- Help in a crisis
  - Fix dependencies, help get consumers to upgrade

Some tooling would help Curation

- Automated builds and test
  - Let’s make it easy for anyone to build and test a package
  - We also need this for automated fuzzing (see OSS-Fuzz)
  - SLSA provenance
- Proof of “Green” — include a proof with a PR that it passes all tests
- Use OSV — precise vulnerability data enables automation
- Clear rules for versioning and resolution
  - Versions: clarity on backwards compatibility
  - Signatures: Curators need to sign “fixed” variants
  - Resolution: figure out which versions will be installed, but without having do installation

What You Can Do

Governments/Enterprises

- Decide “as is” or curated
  - Pay for curation
- Support the things you use

75% of GitHub repos to which Alphabet contributes… are not Alphabet projects[1]

Maintainers

- Basics: Scorecards, 2FA
- Do you want to do curation?
- Work with curators
- Curators for your dependencies?
- Enable build and test by others (at their expense)
- SLSA provenance

1) https://opensource.googleblog.com/2021/08/metrics-spikes-and-uncertainty-open-source-contribution-during-a-global-pandemic.html

Sustainable Trustworthy Open Source

Open Source for “critical” infrastructure needs to meet higher expectations

- Needs ongoing maintenance, but paid for in a sustainable manner
- We’re trying many experiments:
  - Google: sos.dev (rewards for fixes), direct funding to foundations and maintainers
  - OpenSSF: White House summit, Alpha-Omega program

Personally working on three things:

1) **OpenSSF as a neutral place to sort out some hard problems**
2) **Open Source is public infrastructure and a public good**
   - … and deserves government funding
3) **We need (more) Curation**
   - Separate the roles of distribution from accountability
   - Provide trustworthiness for a fee to consumers that care
   - But work with maintainers in a productive, respectful way
   - Address the real costs of mundane but important work!

Backup

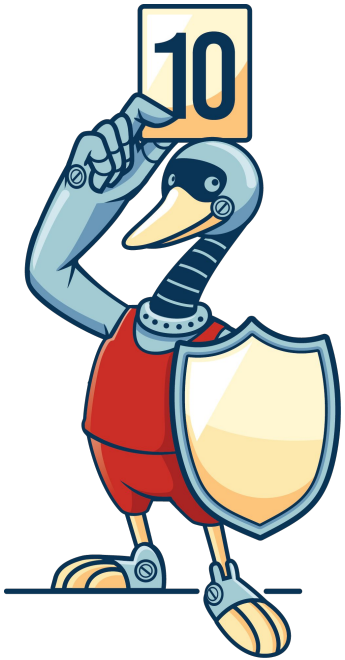
Open SSF

“The purpose of the Open Source Security Foundation (the “OpenSSF”) is to inspire and enable the community to secure the open source software we all depend on, including development, testing, fundraising, infrastructure, and to support initiatives driven by Working Groups (non-software focused) and Projects (software focused) ...”

Open Source Insights

- New, free service: https://deps.dev
- Builds accurate transitive dependency graph for open source packages in:
  - NPM, Cargo, Go, Maven, PyPi
- Connects vulnerability data to affected packages — transitively
- Web interface and BigQuery public dataset

Open Source security risk assessment



OpenSSF Security Scorecards

- Automates analysis and trust decisions on the security posture of open source projects
- Finer grain assessment
- Provide recommendations to improve the security posture of the critical projects we all depend on

Open Source Support Approach

- Maintenance is a security issue
- Google has been a significant open source contributor and will continue to be
  - 15,000+ Googlers contribute to open source
  - 2020
    - Contributed to 90,000 GitHub repositories
    - 75% of repositories with pull requests opened by Alphabet contributors were outside of Google-managed organization
- “Time, Treasure, or Talent”
  - Increasing open source security will take all three

https://opensource.googleblog.com/2020/08/open-source-by-numbers-at-google.html
https://opensource.googleblog.com/2021/08/metrics-spikes-and-uncertainty-open-source-contribution-during-a-global-pandemic.html

Supply-chain Levels for Software Artifacts (SLSA)

- Google system: Binary Authorization for Borg
  - cloud.google.com/security/binary-authorization-for-borg
- Checks in BAB → general framework → SLSA
- Complementary to SBOM
  - SBOM missing provenance
- **slsa.dev**

SLSA 4 Requires two-person review of all changes and a hermetic, reproducible build process

Requirement	SLSA 1	SLSA 2	SLSA 3	SLSA 4
Source - Version controlled		✓	✓	✓
Source - Verified history			✓	✓
Source - Retained indefinitely			18 mo.	✓
Source - Two-person reviewed				✓
Build - Scripted build	✓	✓	✓	✓
Build - Build service		✓	✓	✓
Build - Build as code			✓	✓
Build - Ephemeral environment			✓	✓
Build - Isolated			✓	✓
Build - Parameterless				✓
Build - Hermetic				✓
Build - Reproducible				○
Provenance - Available	✓	✓	✓	✓
Provenance - Authenticated		✓	✓	✓
Provenance - Service generated		✓	✓	✓
Provenance - Non-falsifiable			✓	✓
Provenance - Dependencies complete				✓
Common - Security				✓
Common - Access				✓
Common - Superusers				✓

○ = *required unless there is a justification*