THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA

Embedded Linux Conference

The Architecture of Accountability: Transparency in Software

Hayden Blauzvern
Google Open Source Security Team

Who am I?

- Technical Lead on Google's Open Source Security Team
- Core maintainer on Sigstore, open source signing tooling
- Interested in software supply chain security, code signing

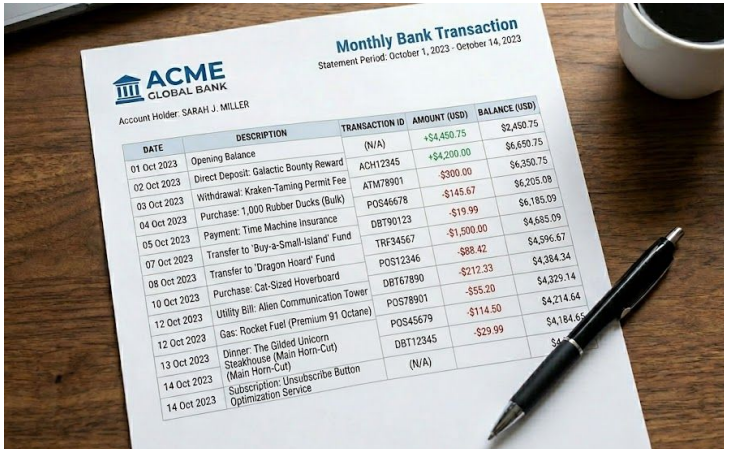
Motivating "Transparency"

Making a "claim"

1. Make a statement
2. Record that statement
3. Be held accountable for it
   a. Can't take it back
   b. Someone will verify it
4. Another party will rely on that claim

Bank Account Transactions

- How do I audit all deposits and withdrawals from my bank account?
- Can someone steal money without me noticing?
- Can my bank take money out without me noticing?



Releasing a Binary

- Record every digest of binaries for applications I manage
- Can I tamper with the binaries after I released them?
- Can I inject malware into them?

What I Ate for Lunch

- Recording my meals for my nutritionist
- Can I lie to them?
- Does anyone else care about my eating habits?

Defining "Transparency"

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA
Embedded Linux Conference

Discoverability & Auditability

Claimant Model

"There is a **Claimant** that makes a **Claim** that is relied upon by a **Believer** as a precondition to take an action they would not have taken if the claim was false. Claims are represented by signed **Statements**. The veracity of a Claim can be verified by a **Claim Verifier**, who will notify a **Claim Arbiter** of any false Claims." - Claimant Model

Claimant Model - Bank Transactions

Claimant	The bank
Claim	There exists a list of all deposits and withdrawals for a given bank account
Believer	Owner of bank account
Statement	A record of a transaction on a bank account
Verifier	Owner of bank account
Arbiter	Bank owner, the government

Claimant Model - Binaries

Claimant	Binary owner
Claim	There exists a list of unique binaries that the owner published
Believer	Anyone using that binary
Statement	Binary digest, name & version
Verifier	Anyone who can see the list, verifying that it doesn't change over time and there are not multiple digests for a given binary name/version
Arbiter	Package registry or similar

Claimant Model - Meals

Claimant	Me, who is hungry
Claim	I make available a list of all meals I've eaten
Believer	My nutritionist
Statement	Food and time of meal
Verifier	???
Arbiter	???

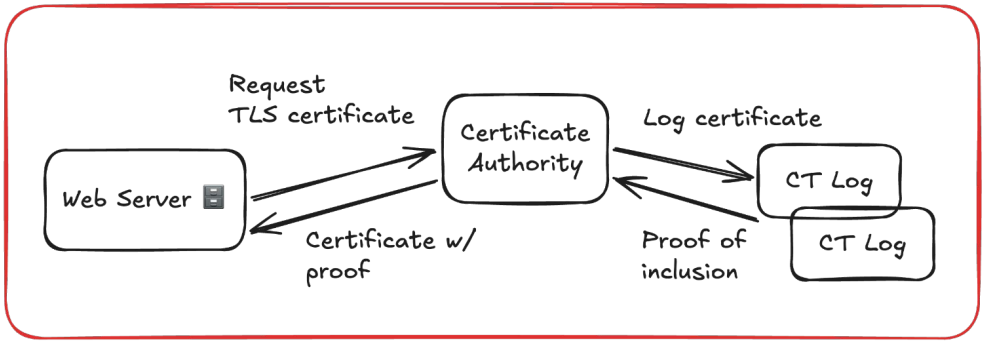
Building Blocks for a Transparent System

- Append-only list of statements, i.e. "log"
- Prove inclusion in the list to a Believer
  - Inclusion means that a Verifier can actually verify the claim
  - No inclusion means the statement shouldn't be relied upon
- Auditability for the Verifier
  - Efficiently verify that all relevant claims are correct

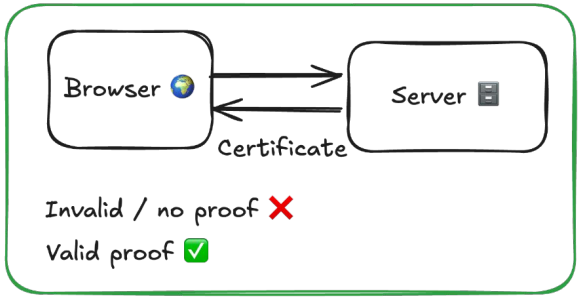
Real-World Examples

Certificate Transparency

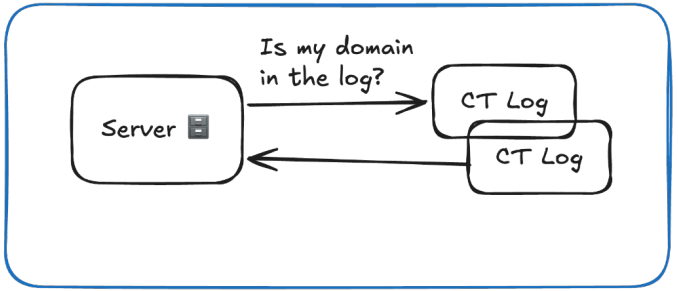
Claimant



Believer



Verifier

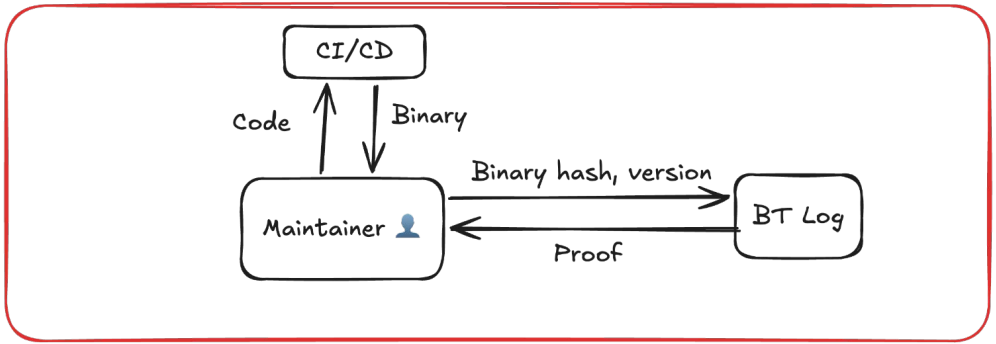


Certificate Transparency

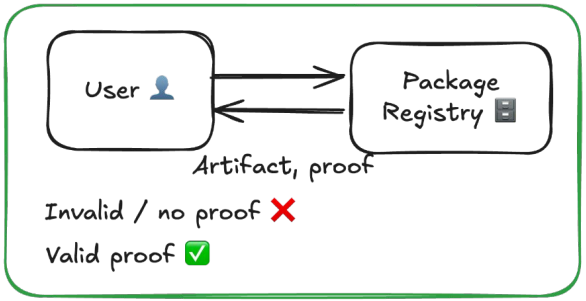
- In response to a major CA compromise (DigiNotar)
- 2013: First public CT log, standardized in RFC 6962
- 2015: Chrome mandated CT log inclusion
- Now, all major browsers mandate log inclusion, with a network of log operators

Binary Transparency

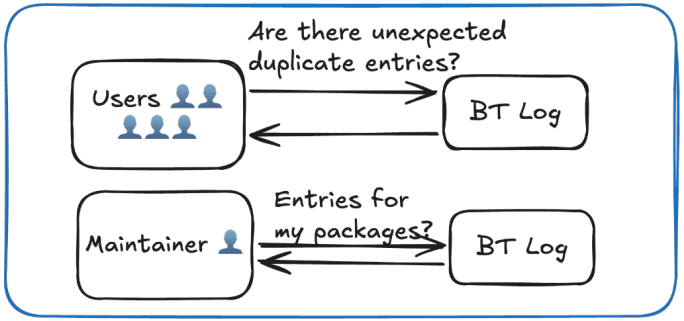
Claimant



Believer



Verifier

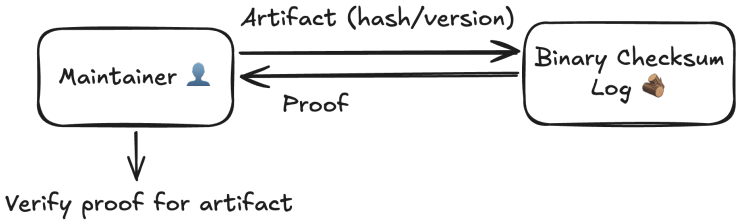


Binary Transparency

- Sigstore for key/identity usage transparency
- Go checksum database
- Android Pixel firmware transparency
- F-Secure USB Armory Drive

Binary Transparency

- Future work
  - More package registries: Registry & developer can guarantee that the same package that was uploaded is also distributed globally to all users

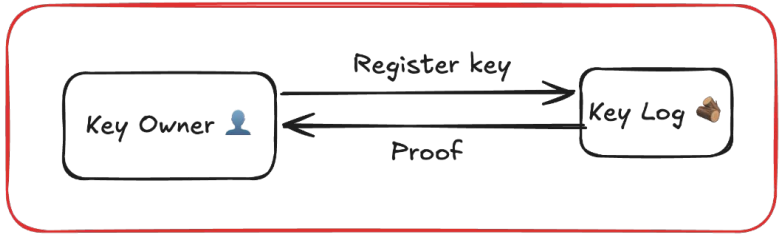


Binary Transparency

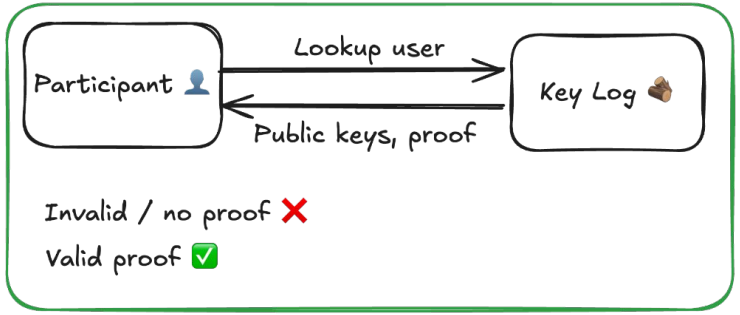
- Future work
  - More package registries: Registry & developer can guarantee that the same package that was uploaded is also distributed globally to all users
  - Web resource transparency
    - Web-based Code Assurance and Transparency (Freedom of the Press Foundation)
    - Web Application Integrity, Consistency, and Transparency (Cloudflare/Mozilla)

Key Transparency

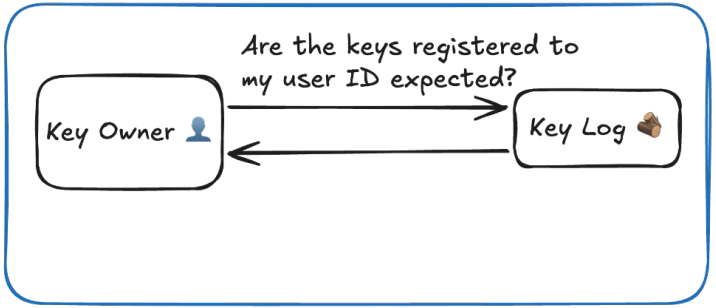
Claimant



Believer



Verifier



Key Transparency

- Messaging applications for end-to-end encryption
  - Apple Messages for iMessage Contact Key Verification
  - WhatsApp
  - Signal
  - Proton Mail
- age public key server
  - https://keyserver.geomys.org/

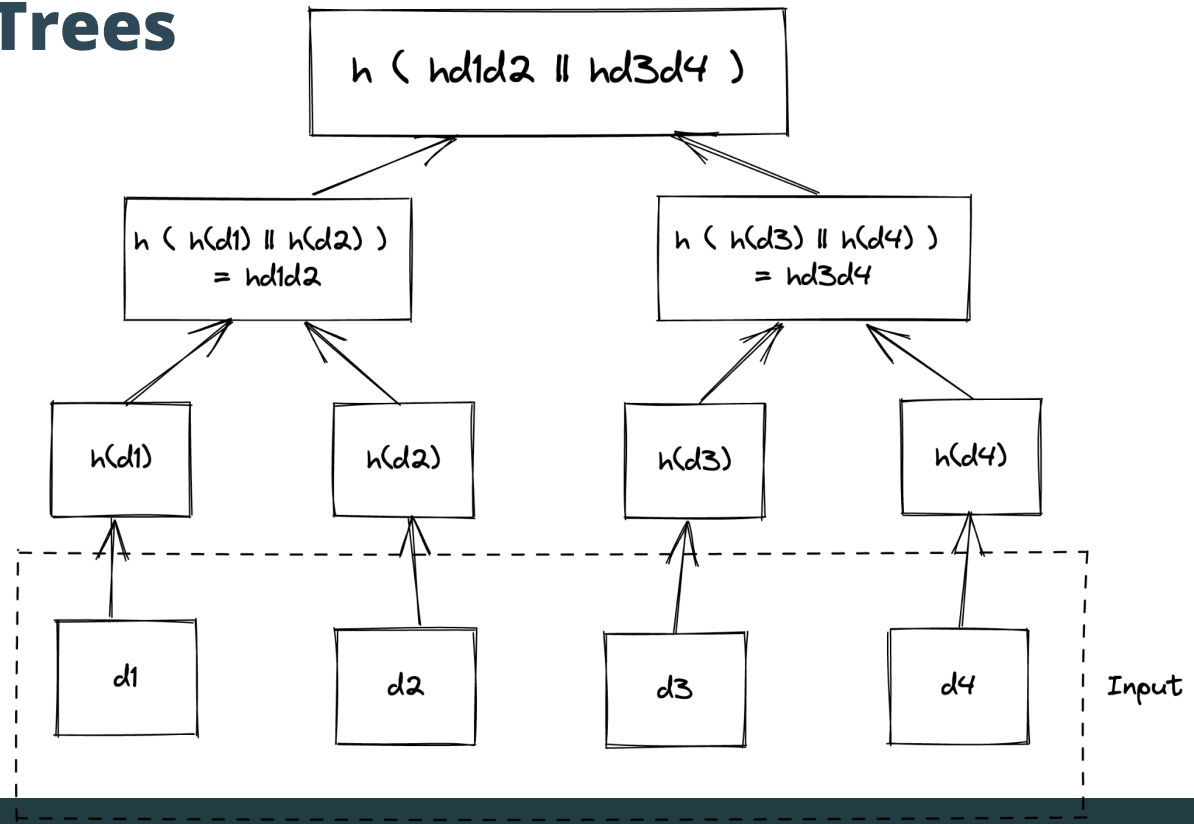
Implementation & Standards

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA
Embedded Linux Conference

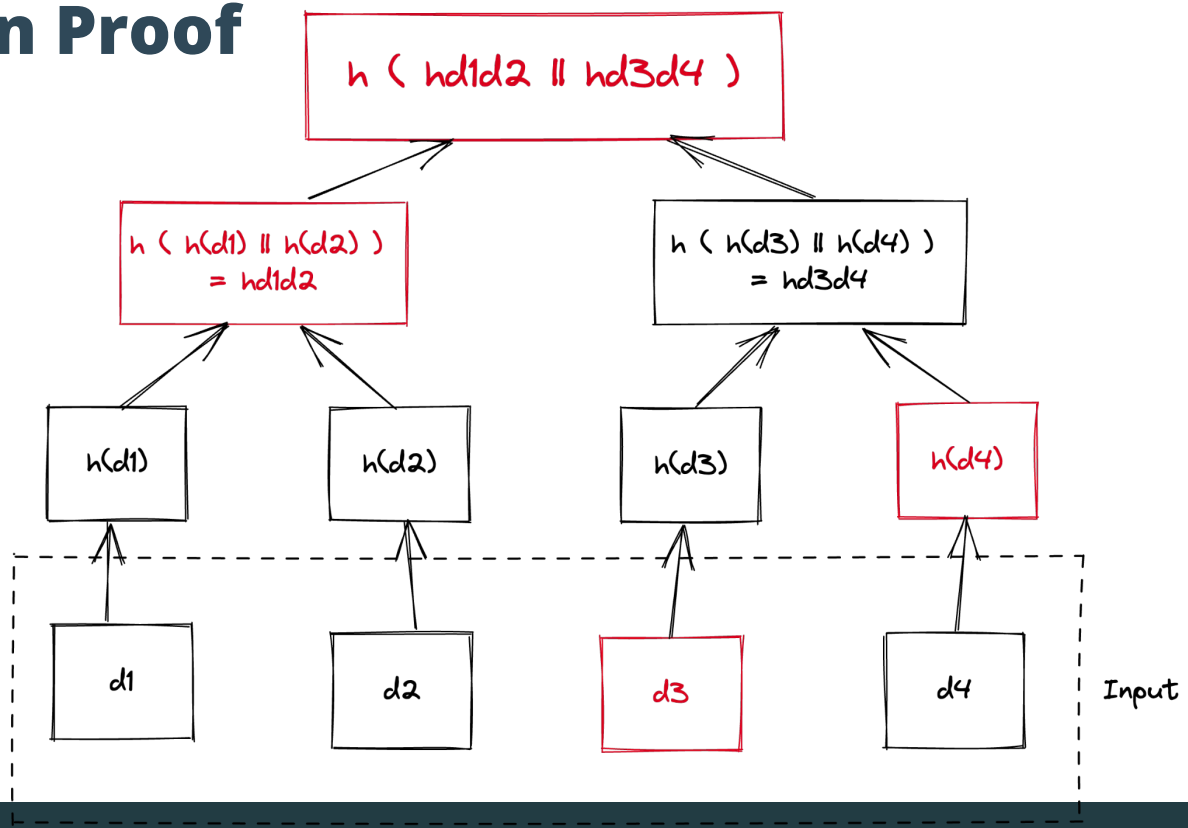
Implementing Transparency

- Append-only data structure - Merkle trees, authenticated dictionaries
- Inclusion proofs - Cryptographic commitment for an included claim, verified with checkpoint and statement
- Checkpoint - Current state of log, signed by log
- Consistency proofs - Cryptographic proof log remains append-only
- Witnesses - Independently verify consistency, prevent split-views

Merkle Trees

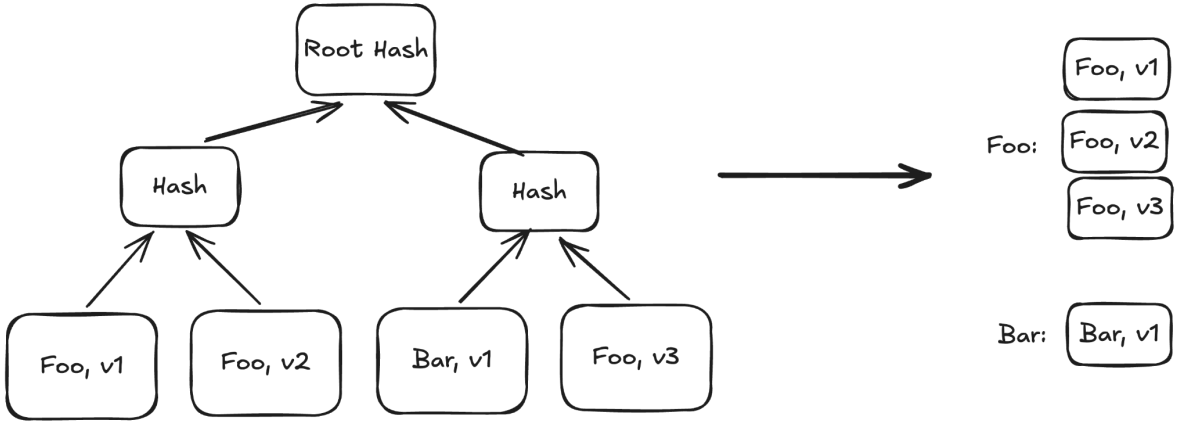


Inclusion Proof



Claim Verification

- Continuously query log
- Verifiable index



C2SP - Community Specifications

- Log API & tiled storage
- Log checkpoint
- Witnessing
- Certificate transparency

The Community Cryptography Specification Project

The Community Cryptography Specification Project (C2SP) is a project that facilitates the maintenance of cryptography specifications using software development methodologies. In other words, C2SP applies the successful processes of open source software development and maintenance to specification documents.

- C2SP decisions are **not based on consensus**. Instead, each spec is developed by its maintainers, who are responsible for reviewing and accepting changes, just like open source projects. This enables rapid, focused, and opinionated development. Since C2SP produces **specifications, not standards**, technical disagreements can be ultimately be resolved by forking.
- C2SP specs are **updateable**, and follow semantic versioning. Most specifications are expected to start at v0.x.x while in "draft" stage, then stay at v1.x.x for as long as they maintain backwards compatibility, ideally forever. Drafts are expected to bump the minor version on breaking changes.
- C2SP documents are developed as Markdown files on GitHub, and can include ancillary files such as test vectors and non-production reference implementations.

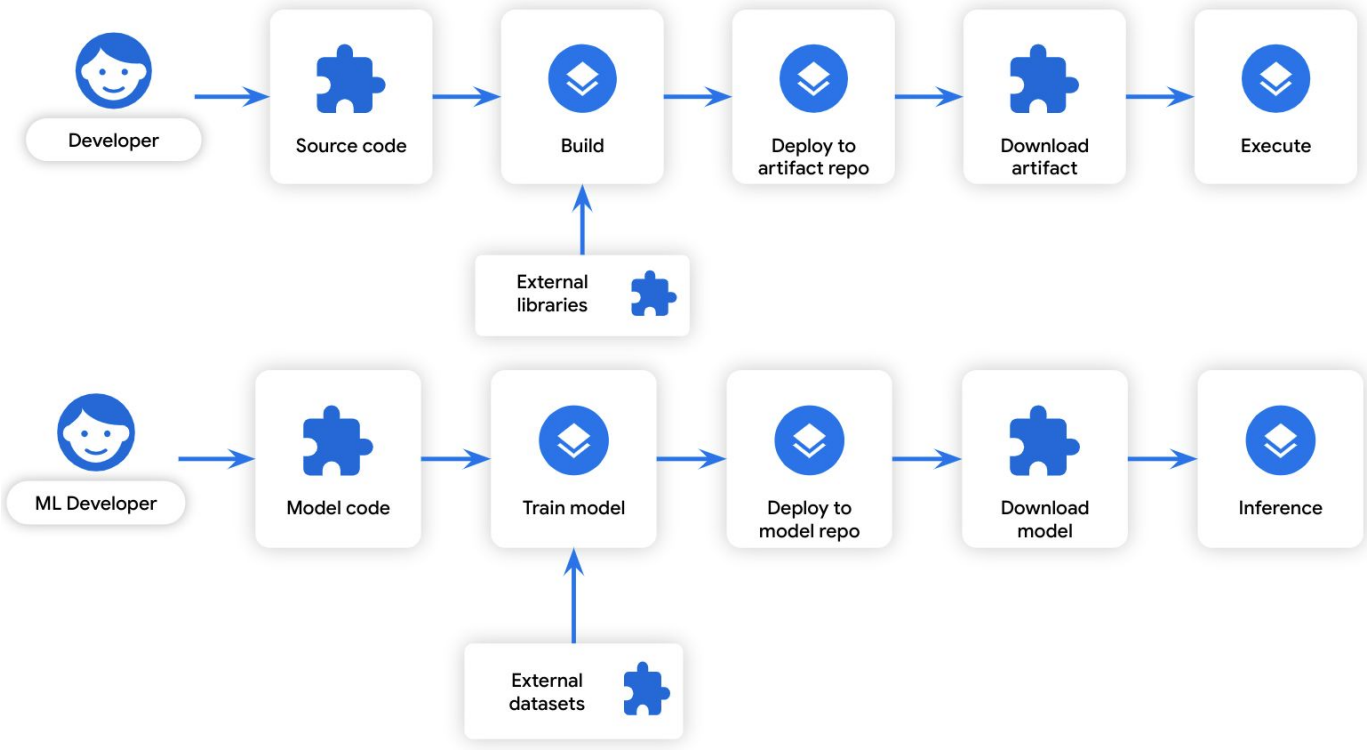
Demo

Demo Summary

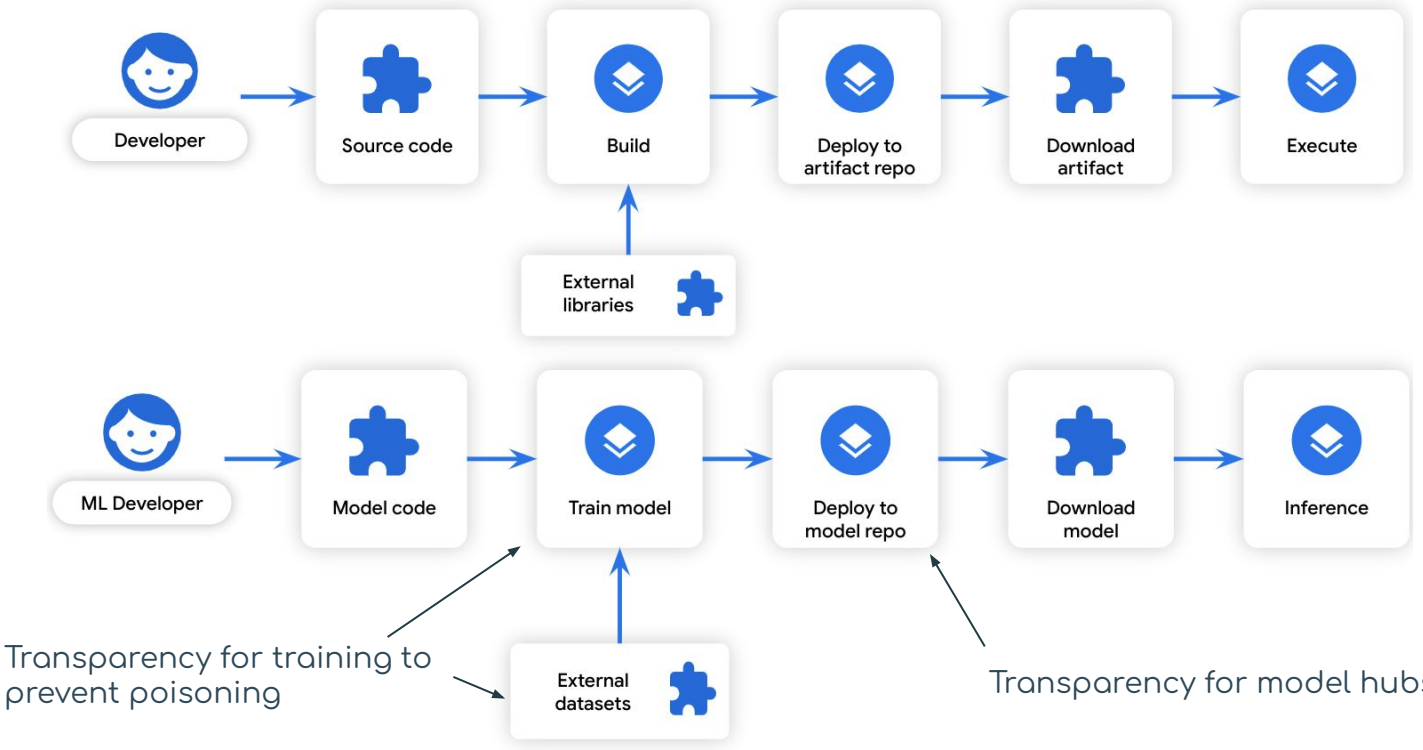
- Example binary transparency log
  - Based on Tessera
  - Records package name, version and checksum
- Serverless log
  - Integrate entries into tiles without a service frontend
  - Perfect for CI/CD

Future Domains

AI Model Transparency



https://security.googleblog.com/2023/10/increasing-transparency-in-ai-security.html

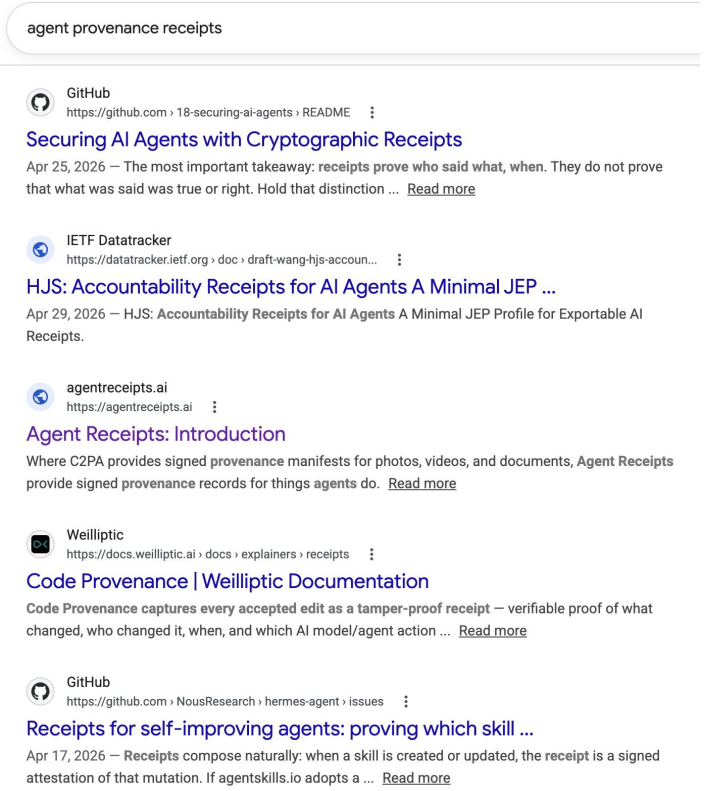


https://security.googleblog.com/2023/10/increasing-transparency-in-ai-security.html

Agent Transparency

Agent Transparency

- Provenance for agent actions, "receipts"?
- Lots of prototypes already
- Build a claimant model!



News Transparency

News Transparency

- Commit to article revisions
  - Research: https://dl.acm.org/doi/abs/10.1145/3579534
- Content authenticity
  - C2PA

Getting Involved

Getting Involved

- Learn more: https://transparency.dev/
- Slack: https://transparency.dev/slack/
- Standards: https://github.com/C2SP/C2SP

Thank you!

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA

Embedded Linux Conference