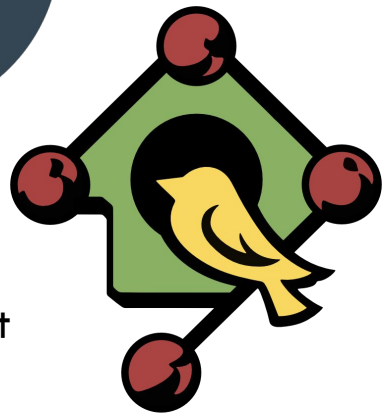


Improving the Trustworthiness of Javascript on the Web

Ezzudin Alkotob* Giulio Berra‡ Benjamin Beurdouche†

Dennis Jackson† Cory Myers‡ Daniel Huigens§

Richard Hansen* Michael Rosenberg¶

*Meta †Mozilla ‡Freedom of the Press Foundation §Proton AG ¶Cloudflare

Motivating Problem

Browser E2EE Messaging is Currently Impossible

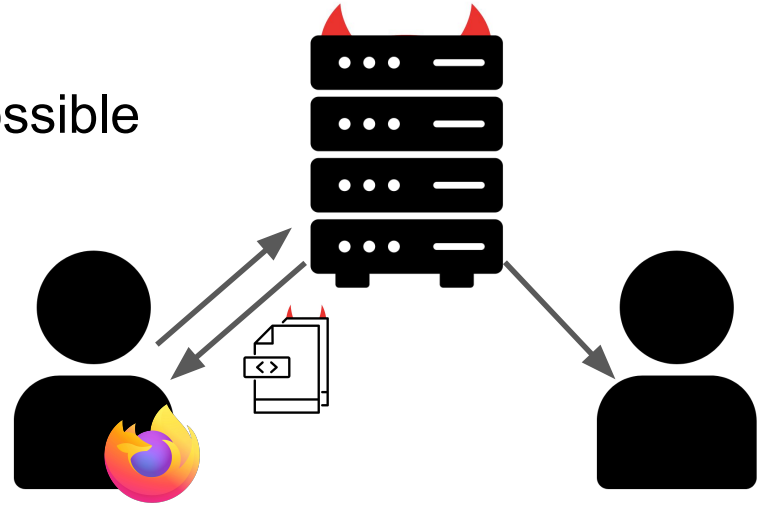
Must be secure even if server is malicious

The code is malicious!

Prevention ❌ **Detection later on ❌**

This is a general problem:

- E2EE Messaging
- Confidential compute
- Password managers
- Online voting



nccgroup

Javascript Cryptography Considered Harmful

If you don't trust the network to deliver a password, or, worse, don't trust the server not to keep user secrets, you can't trust them to deliver security code.

Thomas Ptacek, 2011

Signal for iOS is secure. Why?

App stores are a third party providing:

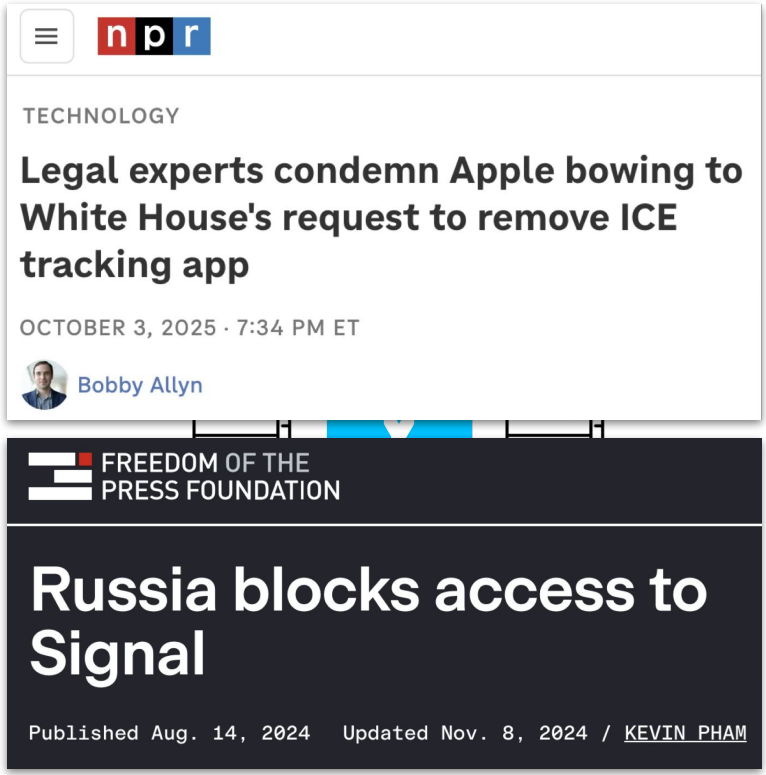
Integrity Apps aren't modified without authorization

Consistency Everyone gets the same app

Transparency* App versions are logged

App stores are too centralizing

Let's build these features for the **open web**



Overview

Bringing Integrity, Consistency, and Transparency to the Web

Code Verify (Meta; 2022)

Code transparency for WhatsApp, IG, Messenger

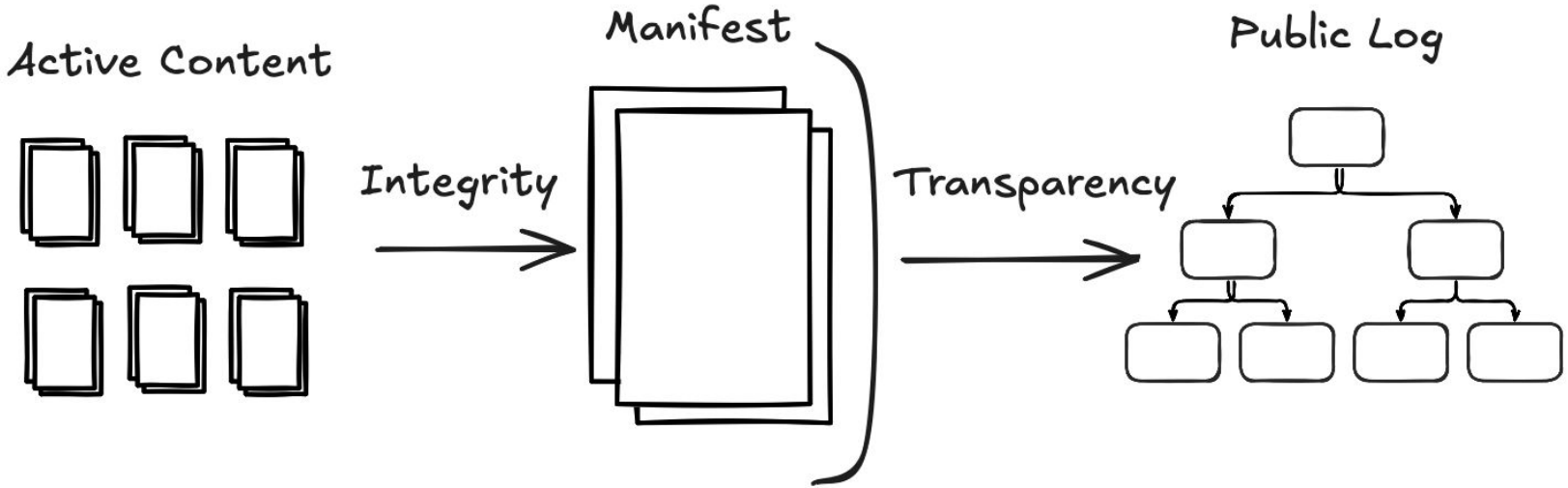
WEBCAT (FPF; 2025)

Code signing and transparency for any site

WAICT (in progress)

Standardized code transparency for any site

Preliminaries



Code Verify

Firefox Browser ADD-ONS

chrome web store

How Code Verify Works

Three-Party Trust Architecture

1. **Meta** publishes web app code + JSON manifest of SHA-256 hashes for every JS/CSS file

2. **Cloudflare** (independent auditor) receives manifest root hash from Meta

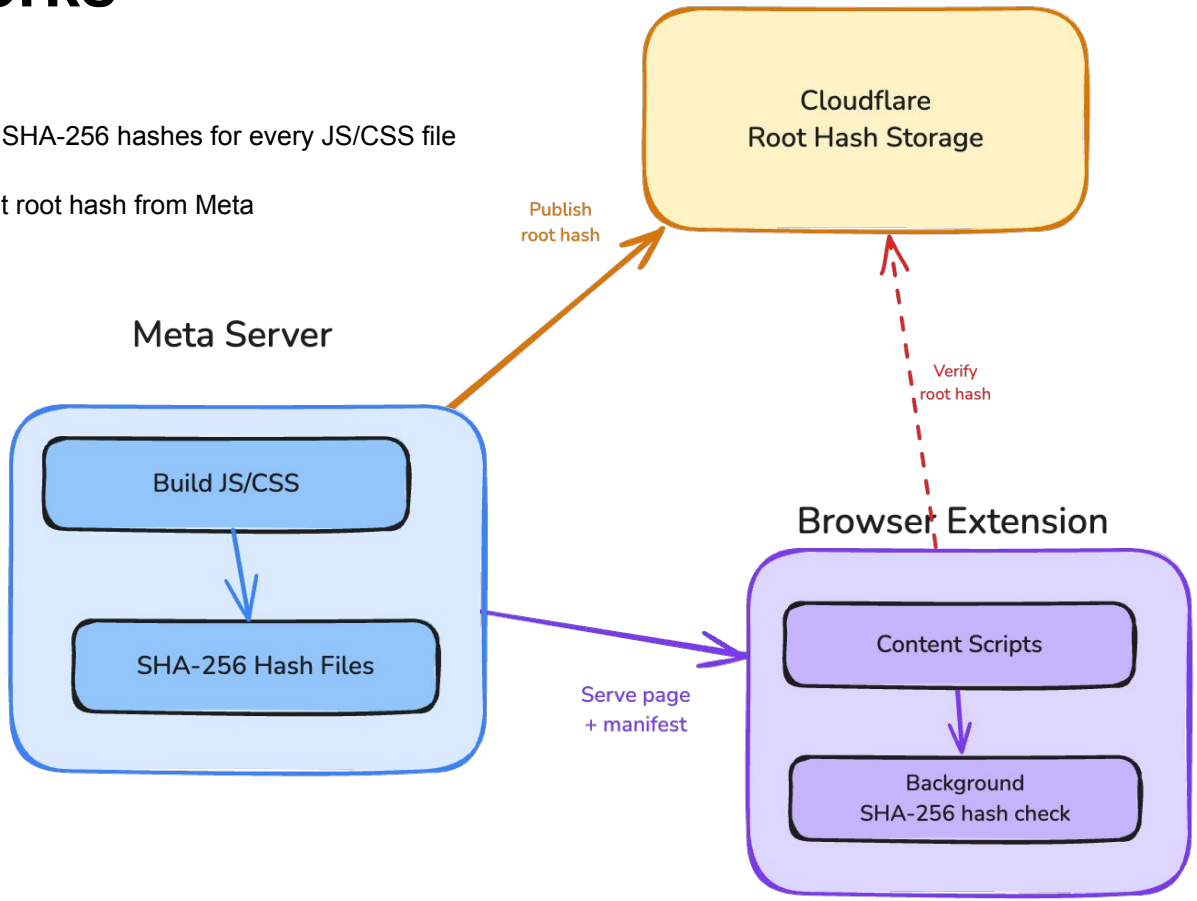
3. **Code Verify** extension independently:

   • Discovers manifest embedded in page HTML

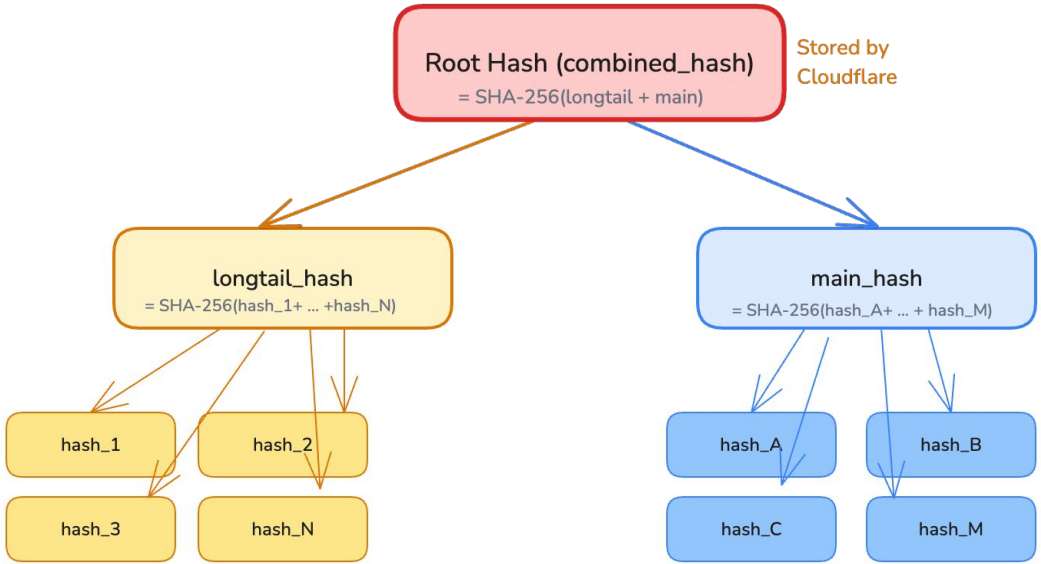
   • Fetches & hashes every script/stylesheet

   • Compares each hash against the manifest

   • Cross-checks root hash with Cloudflare



Manifest Merkle-Tree Hash Structure



Verification

Step 1: Content Script Injection

Step 2: CSP Header Validation

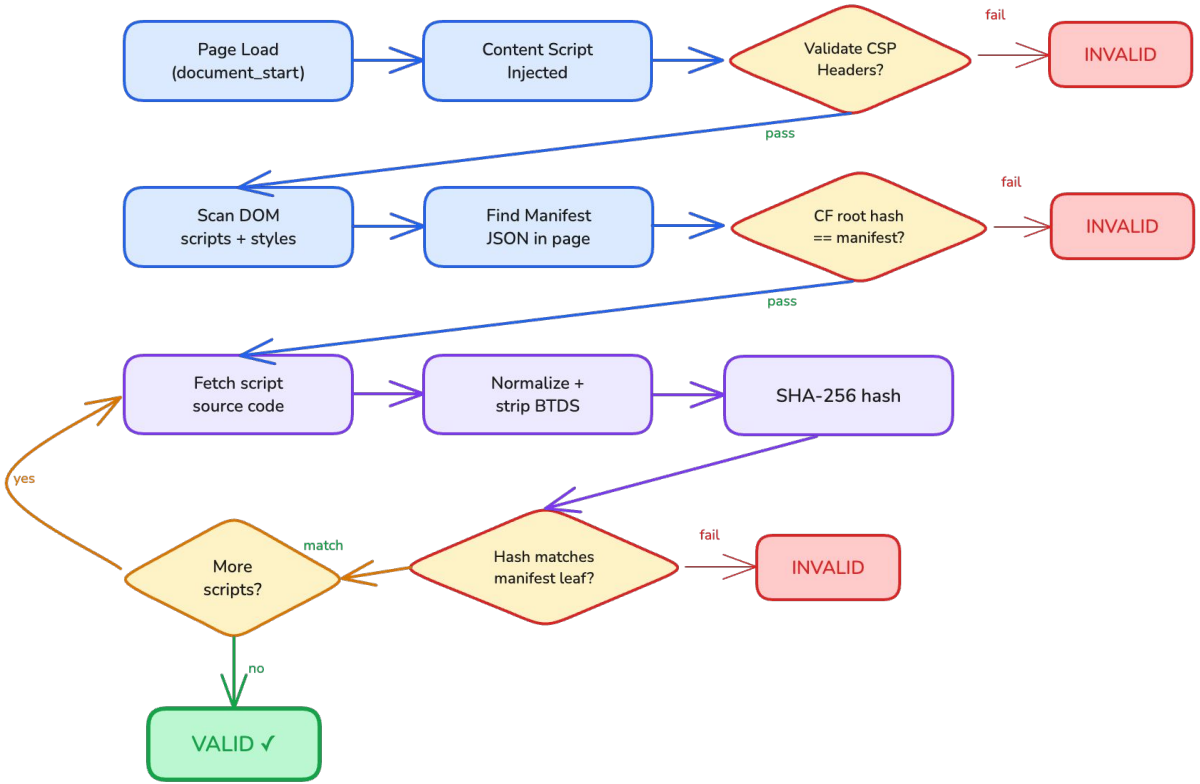
Step 3: DOM Scanning & Monitoring

Step 4: Manifest Discovery

Step 5: Cloudflare Verification

Step 6: Fetch & Normalize Scripts/Styles

Step 7: Hashing & Verifying Scripts/Styles



Additional Security Layers

CSP Header Validation

- No unsafe-inline/unsafe-eval in script-src
- Valid worker-src (no blob:, data:, wildcards)

Web Worker CSP and Monitoring

- Detects unverified scripts via importScripts()
- Worker CSP headers independently validated

Cache Integrity Monitoring

- Monitors webRequest responses
- Re-fetch not from cache → tab invalidated
- Prevents TOCTOU/MITM serving different content

WEBCAT

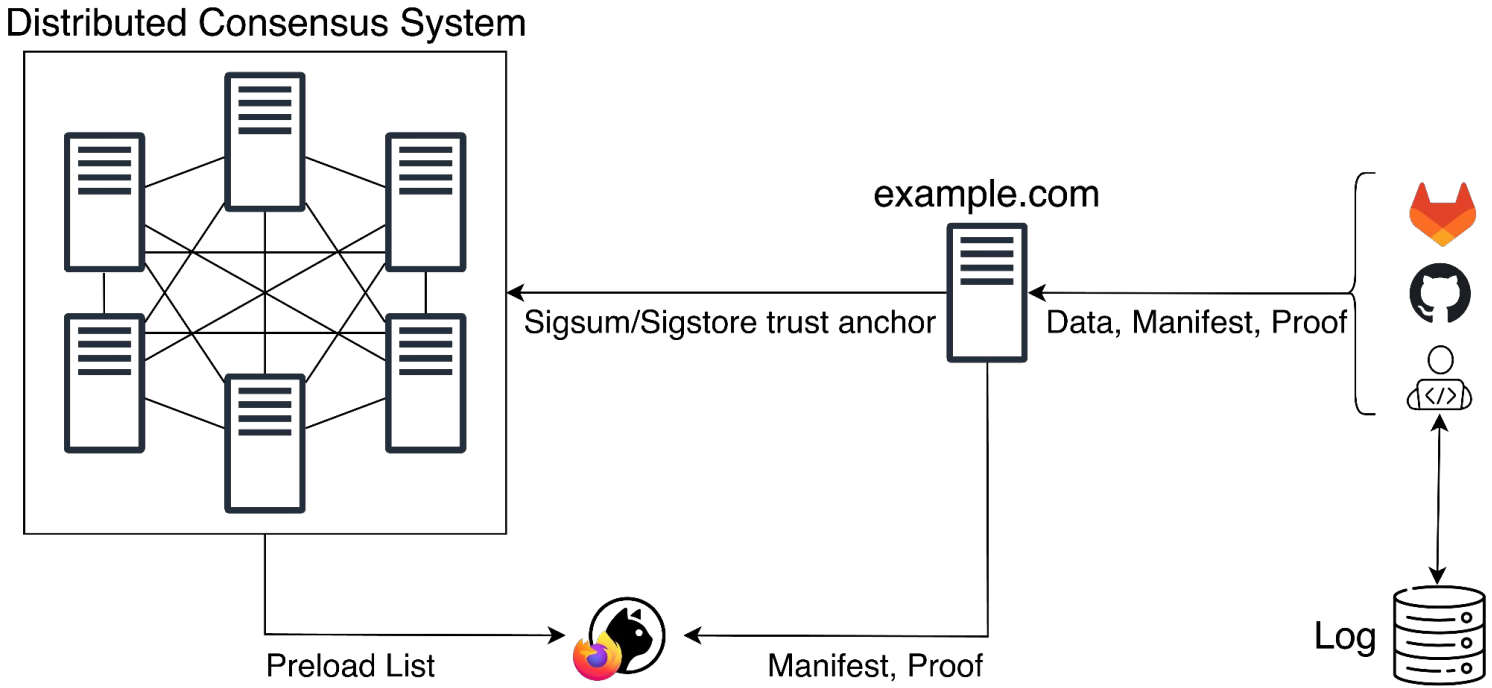
Firefox Browser ADD-ONS

Let’s talk requirements

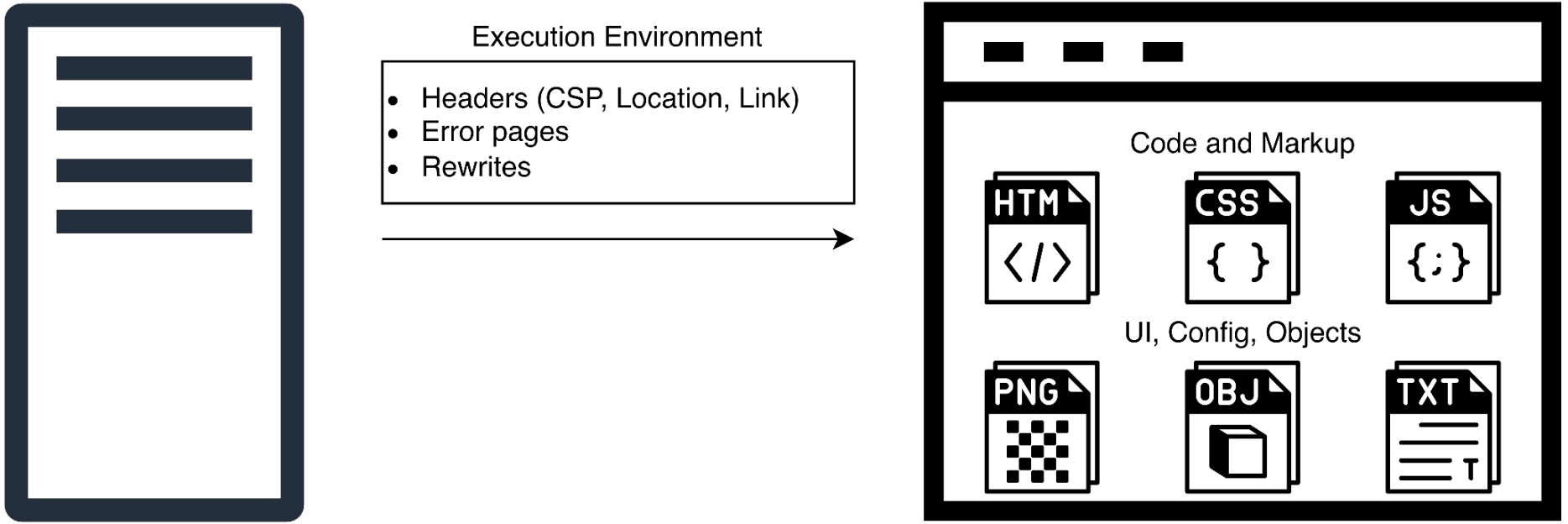
1. General-purpose
2. Fail-closed by default
3. Privacy-preserving
4. Compatible with existing applications
5. No single point of failure

*Focused on **prevention of execution**, detection as a fallback*

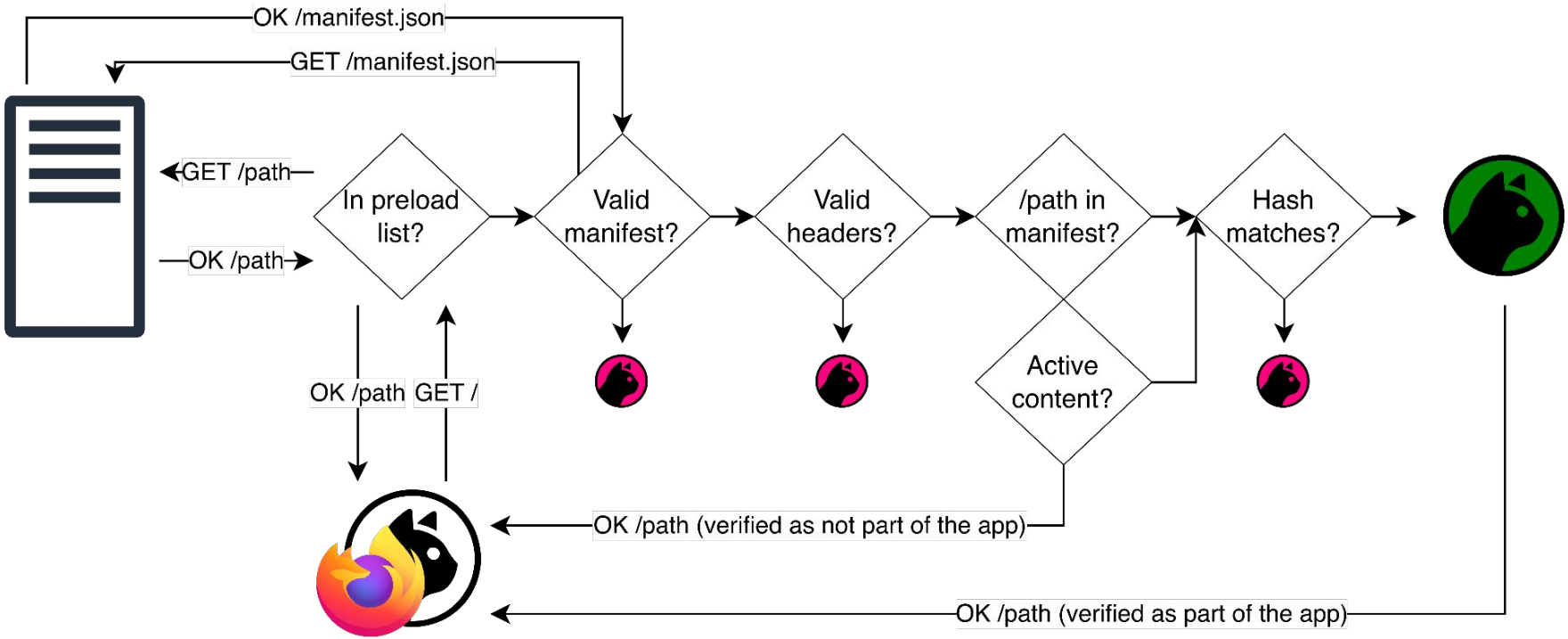
Architecture



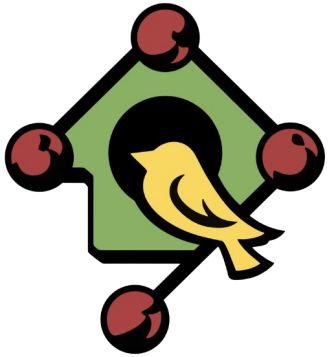
What is a Web Application anyway?



Verification Logic



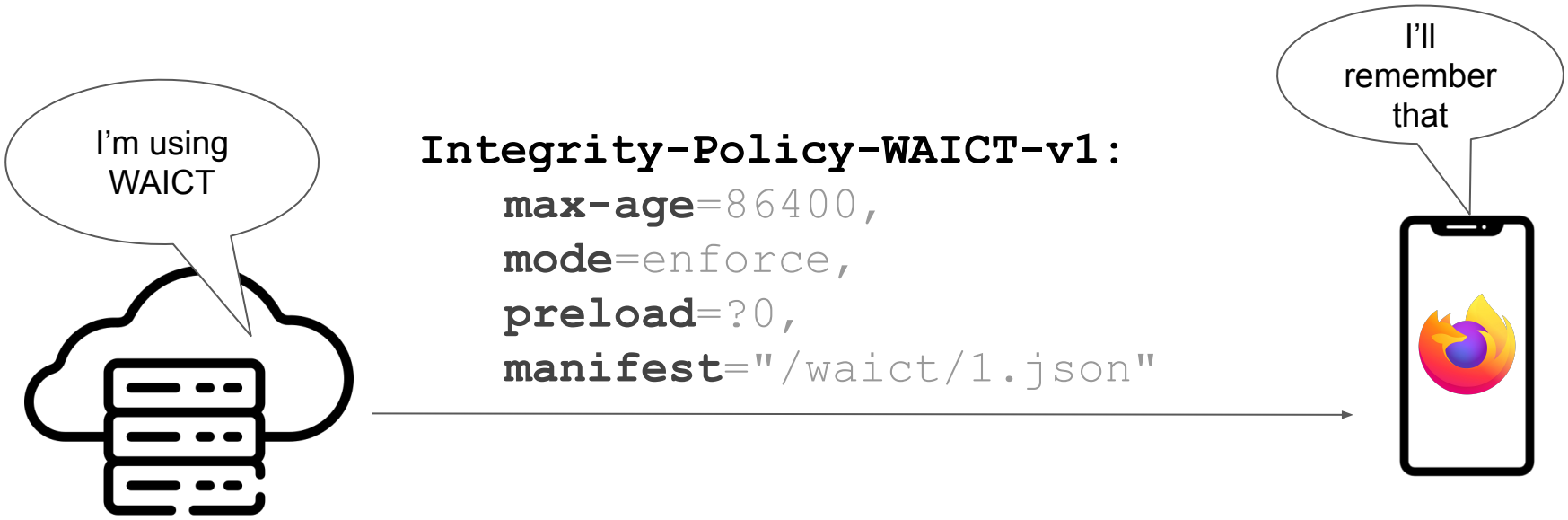
WAICT



Web Application Integrity, Consistency and Transparency

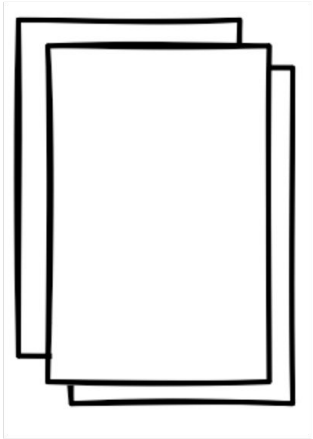
1. Standardize these approaches
2. Integrate into the browser

Deploying WAICT

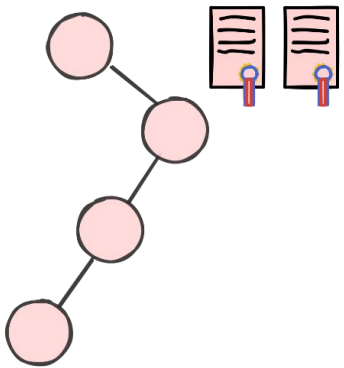


Opt-in Security Policy via HTTP Header

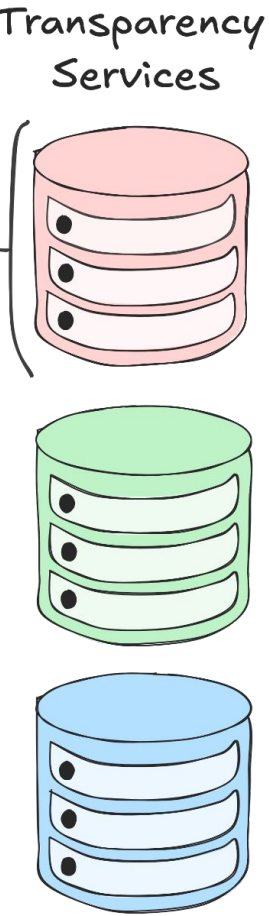
Deploying WAICT



Manifest

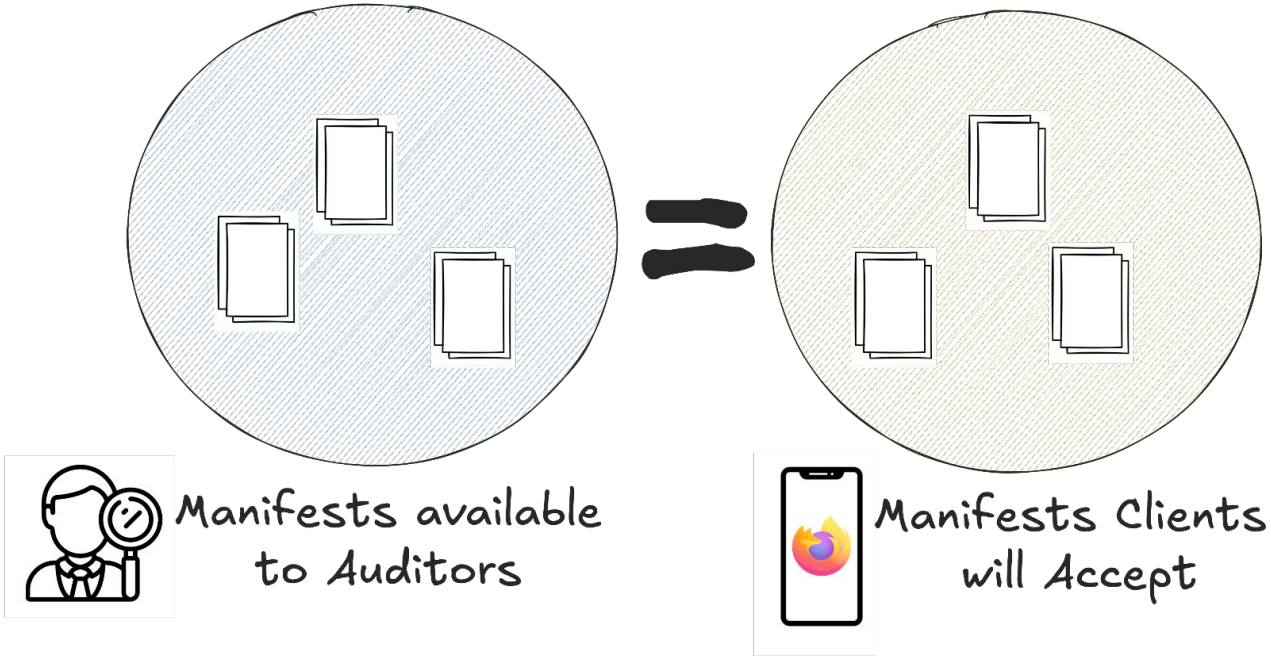


Proof of Inclusion
& Tree Head Signatures



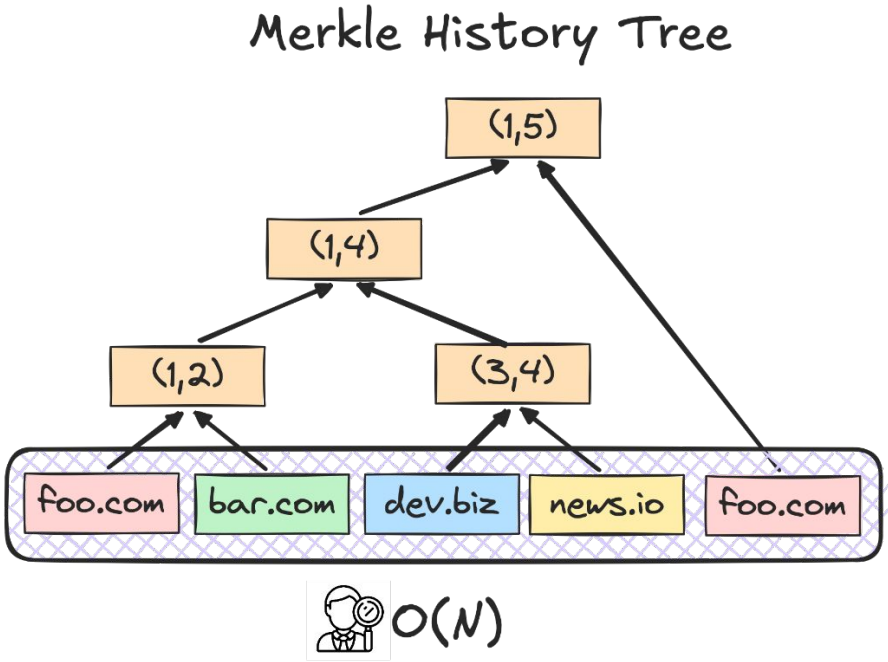
No single point of failure

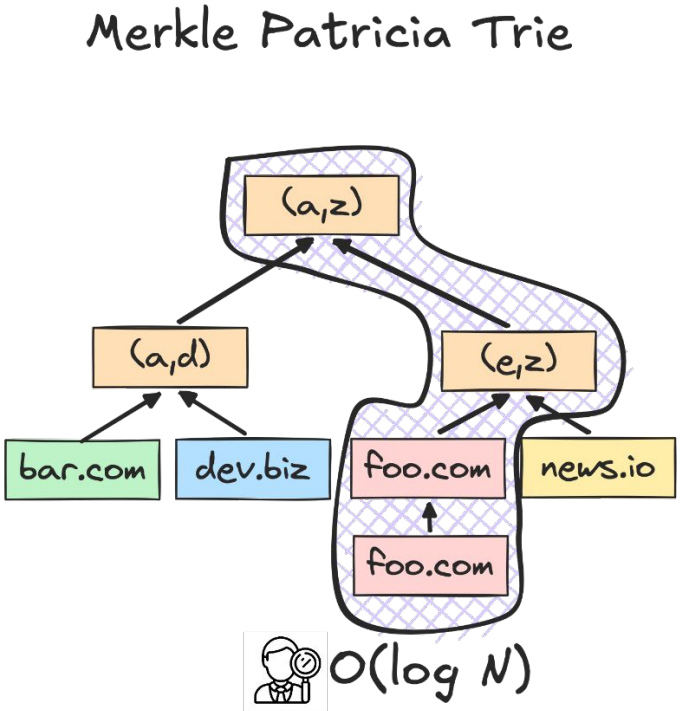
Goal: Transparency



Detection, not prevention!

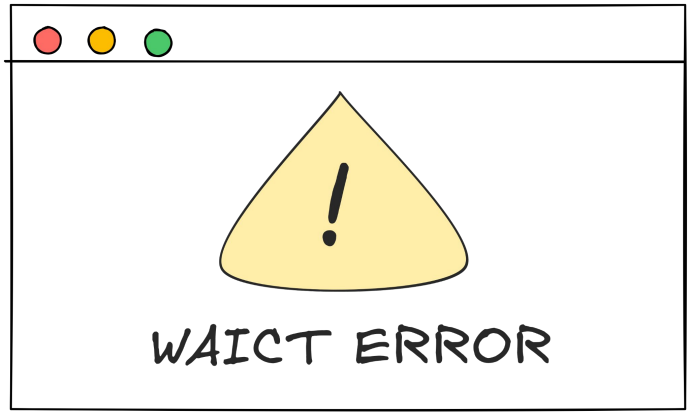
Our Transparency Logs are fancy

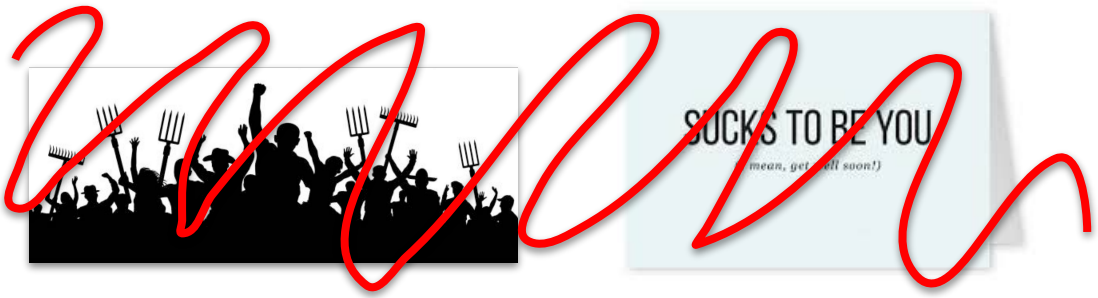


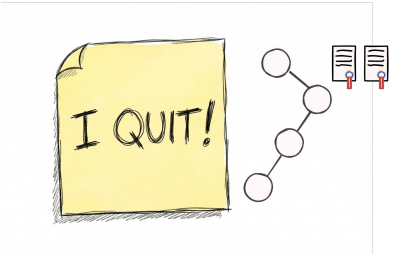


Structured logs enable efficient auditing

Accidents happen





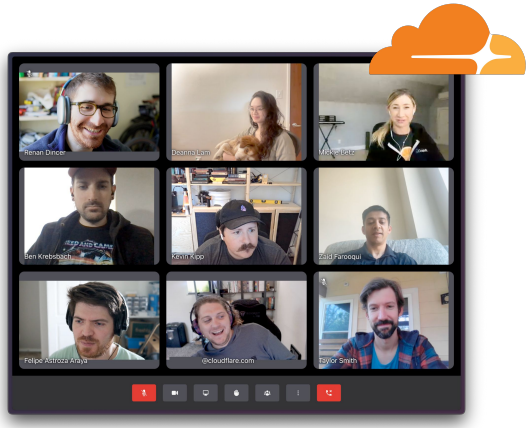


Immediate recovery through transparent opt-out

Running Code and Rough Specs

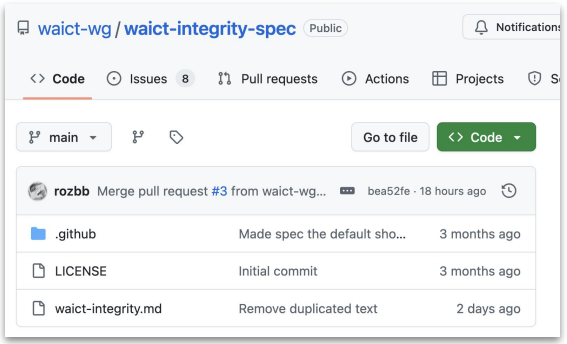
[Firefox Nightly]

Credits:
Tom Schuster
and Anna Weine



[🍊Orange Meets]

Credits: Claude



[Draft Specs]

Trustworthy Javascript for the Web

CodeVerify

WEBCAT

WAICT

Ezzudin Alkotob

Giulio Berra

Demos, Specs and Slides

`rwc26.waict.dev`

Dennis Jackson

Michael Rosenberg

Image Credits