Web Security — S03E02: Browser Isolation Boundaries

Origin vs. Site, SOP, CORS, COOP/COEP/CORP, Strict CSP & SRI

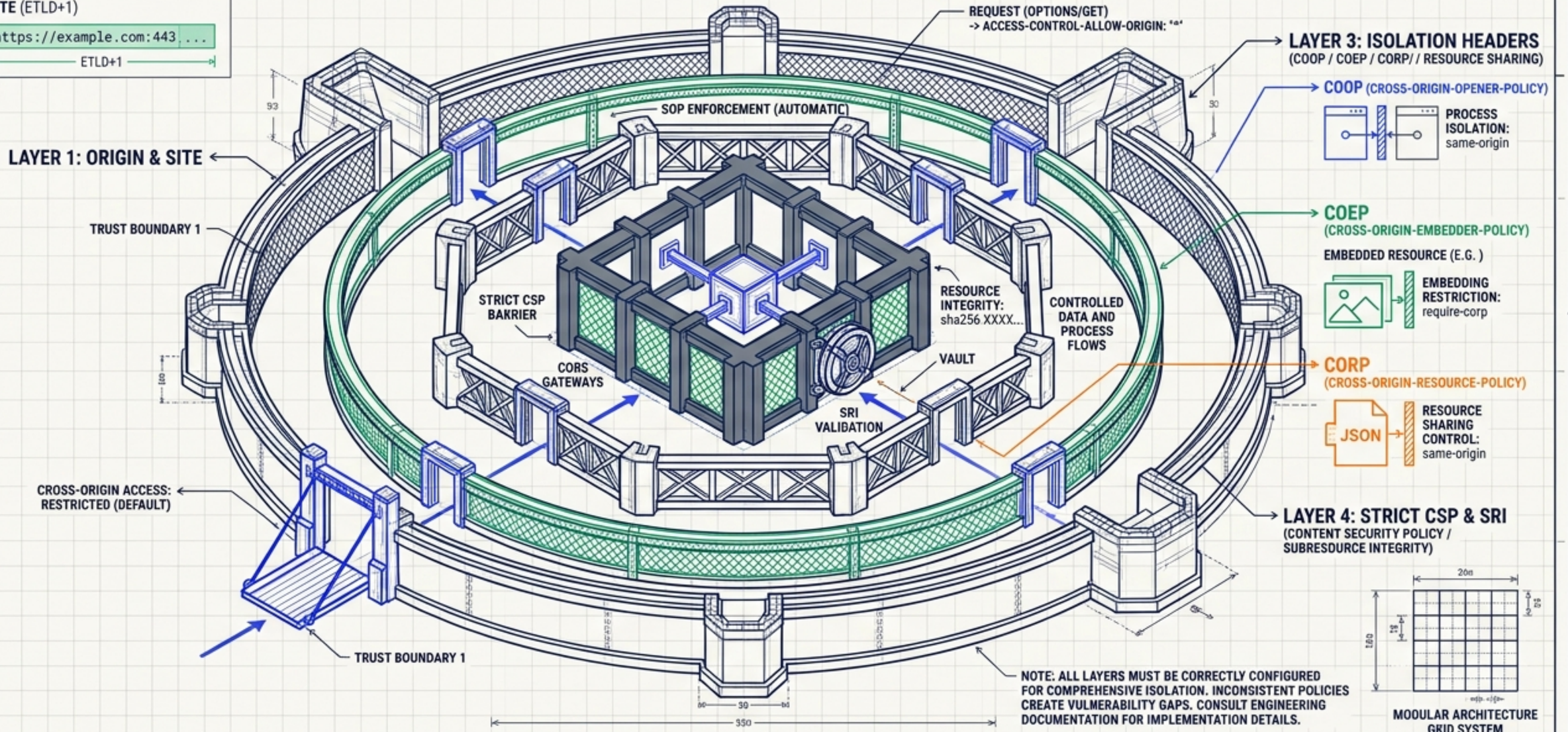
The Concentric Fortress: Architectural patterns for frontend, backend, and platform engineers.

LEGEND:

- **ORIGIN** (SCHEME://HOST:PORT)
  - `https://example.com:443`
  - URL
- **SITE** (ETLD+1)
  - `https://example.com:443 ...`
  - ETLD+1

LEGEND:

PRIMARY STRUCTURAL (NAVY/SLATE) | AUTHORIZED FLOW (COBALT)
SECURITY ENFORCEMENT (EMERALD) | RESOURCE CONTROL (ORANGE)

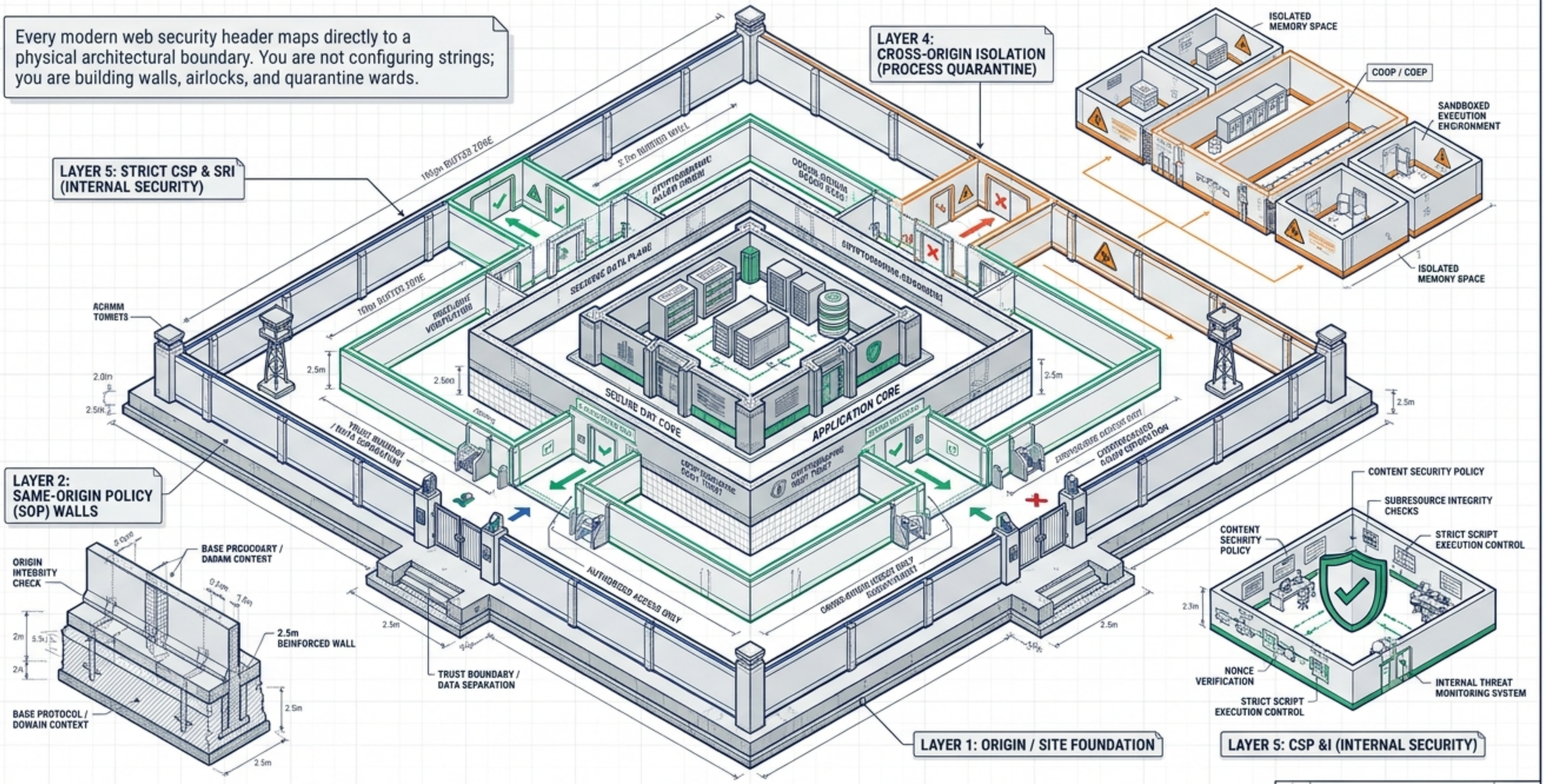


DRAWN BY: WEB SEC ARCHITECTS | DATE: OCT 26, 2024 | REV: 1.0 (FINAL)

Gemini Notebook

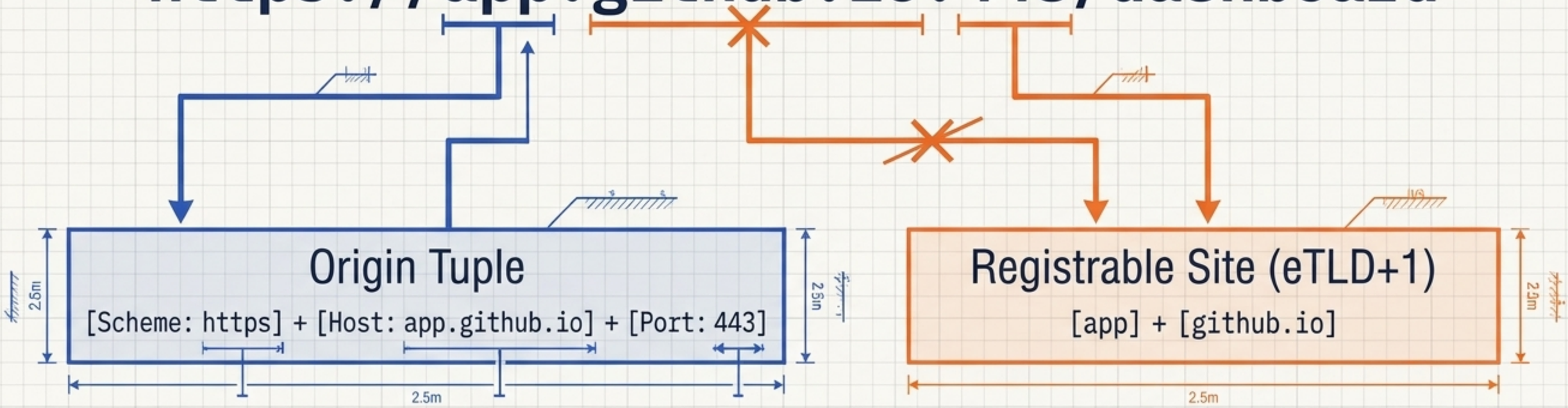
THE BROWSER IS A HIGH-SECURITY CONTAINMENT FACILITY

Every modern web security header maps directly to a physical architectural boundary. You are not configuring strings; you are building walls, airlocks, and quarantine wards.



Gemini Notebook

https://app.github.io:443/dashboard



Key Concept:
RFC 6454 defines an **Origin** as an exact triple-match (Scheme, Host, Port).
The HTML Standard defines a **Site** as the Effective Top-Level Domain plus one (eTLD+1).

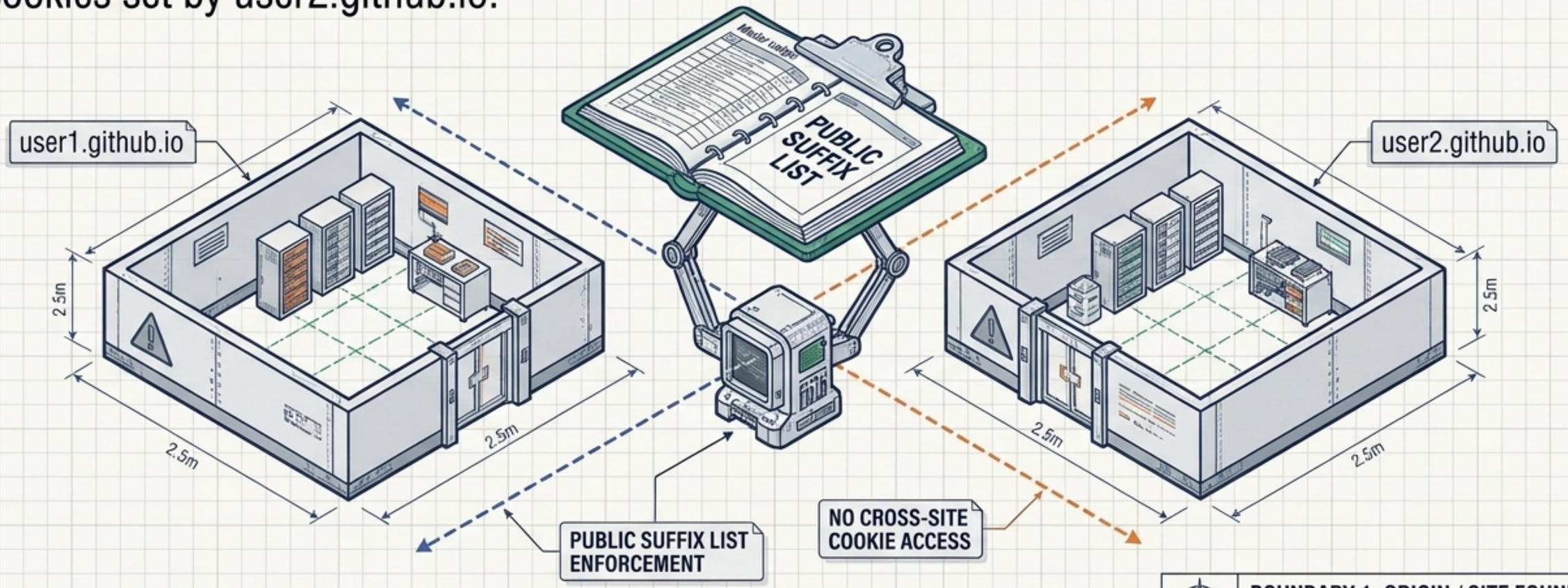
Impact:
Origins dictate DOM/Fetch boundaries.
Sites dictate Cookie and Storage partitioning.

BOUNDARY 1: ORIGIN / SITE FOUNDATION

Gemini Notebook

THE eTLD+1 PROBLEM & THE PUBLIC SUFFIX LIST

- **Rule:** .com is a Top-Level Domain (TLD). company.com is a Site.
- **The Exception:** Shared hosting domains like github.io or s3.amazonaws.com are mathematically subdomains, but practically function as TLDs.
- **The Fix:** Browsers hardcode the Public Suffix List to prevent user1.github.io from reading cookies set by user2.github.io.



BOUNDARY 1: ORIGIN / SITE FOUNDATION

Gemini Notebook

The Origin vs. Site Matrix

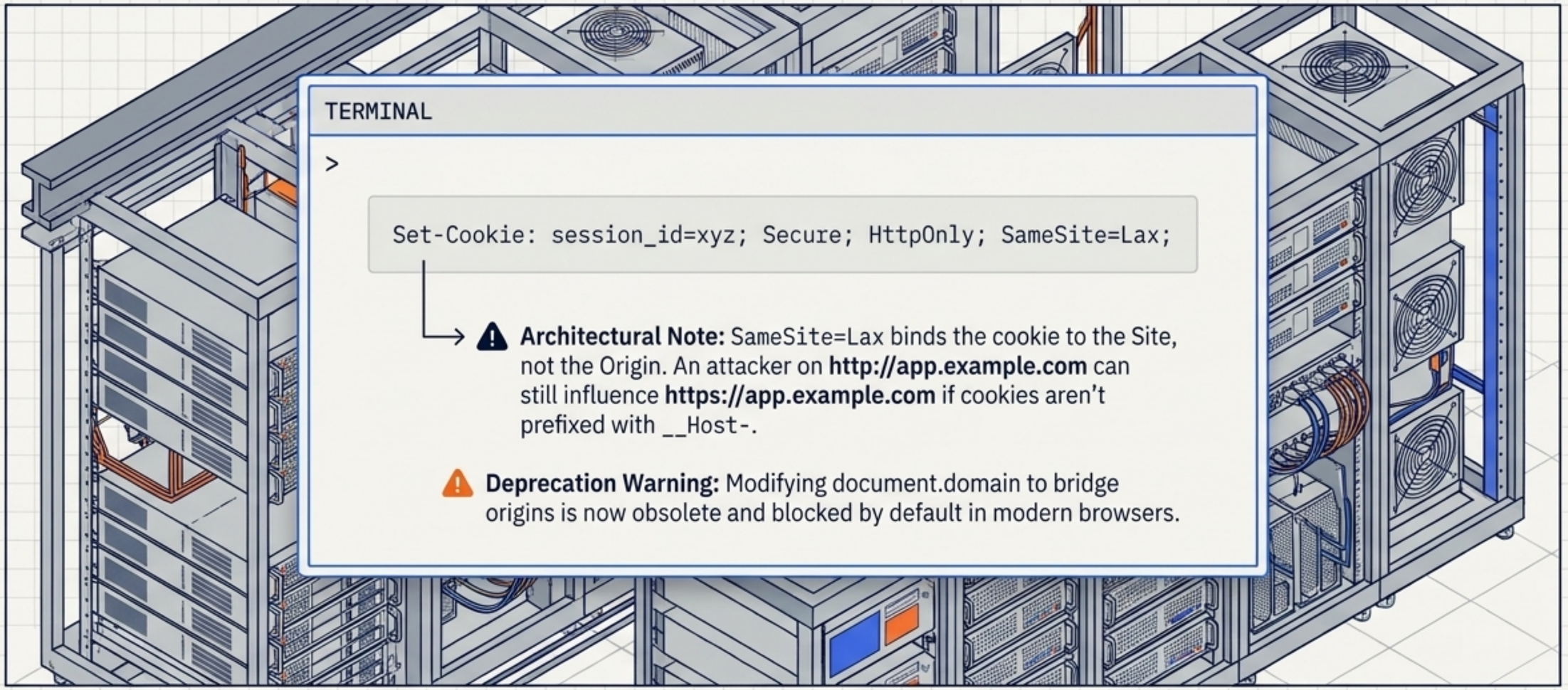
Target URL: https://app.github.io/

	Test URL	Same Origin?	Same Site?
1	http://app.github.io	✕ No (Scheme)	✕ No (Scheme)
2	https://app.github.io:8080	✕ No (Port)	✓ Yes
3	https://api.app.github.io	✕ No (Host)	✓ Yes
4	https://other.github.io	✕ No (Host)	✕ No (PSL Boundary)

Takeaway: Insecure schemes (**HTTP**) no longer share a **Site** with secure schemes (**HTTPS**).

Gemini Notebook

Platform Blueprint: Securing the Foundation



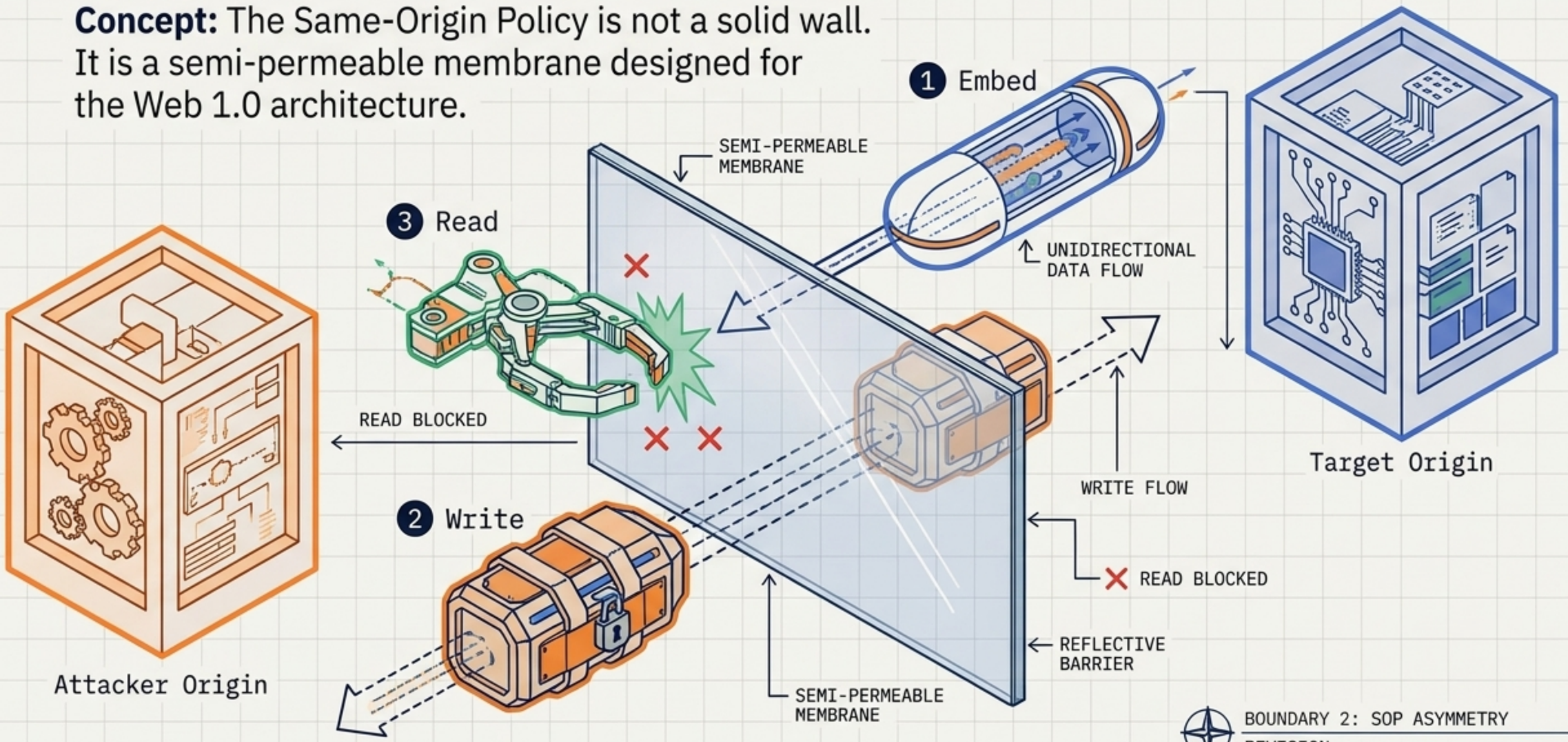
BOUNDARY 1:
ORIGIN / SITE FOUNDATION

REVISION
1

Gemini Notebook

Boundary 2: SOP Asymmetry

Concept: The Same-Origin Policy is not a solid wall. It is a semi-permeable membrane designed for the Web 1.0 architecture.

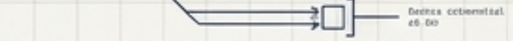


Gemini Notebook

The Network Action Matrix

Insight: SOP blocks reading cross-origin data by default, but permits cross-origin writes. This asymmetry is why Cross-Site Request Forgery (CSRF) is possible out of the box.

Network Action	Default SOP Behavior	Reasoning & Impact
READS (XHR, Fetch, DOM Access)	**BLOCKED**	Prevents unauthorized data exfiltration.
WRITES (Links, Form POSTs)	**ALLOWED**	Enables the fundamental hyperlinked web. **(Creates CSRF vulnerability).** ⚠
EMBEDS (Img, Iframe, Script)	**ALLOWED**	Enables CDNs and composable UI. **(Creates side-channel risks).** ⚠



BOUNDARY 2: SOP ASYMMETRY

REVISION 1

Gemini Notebook

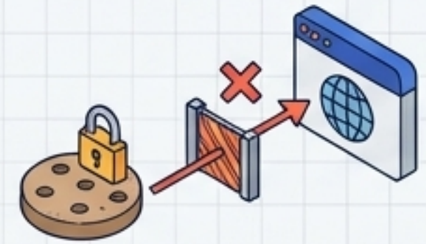
Platform Blueprint: SOP Asymmetry Defense

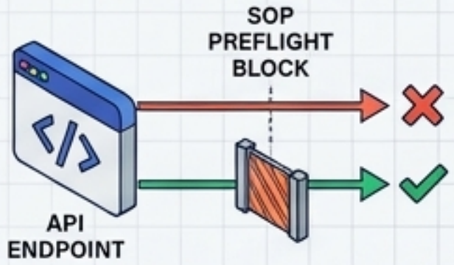
Defense Stack:

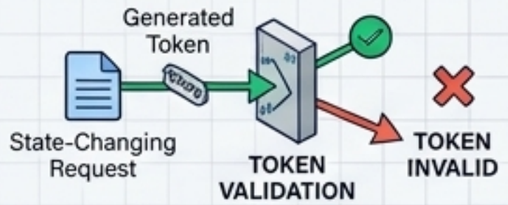
1. **Primary:** `SameSite=Lax` or `Strict` on all session cookies to prevent the browser from appending credentials to cross-origin writes.

2. **Secondary:** Require custom headers (e.g., `X-Requested-With`) on API endpoints. **SOP** blocks cross-origin requests from adding custom headers without preflight approval.

3. **Fallback:** Cryptographic CSRF Anti-Forgery **Tokens** validated on state-changing requests.







Implementation requires backend framework configuration
Preflight checks (OPTIONS requests) must be handled.

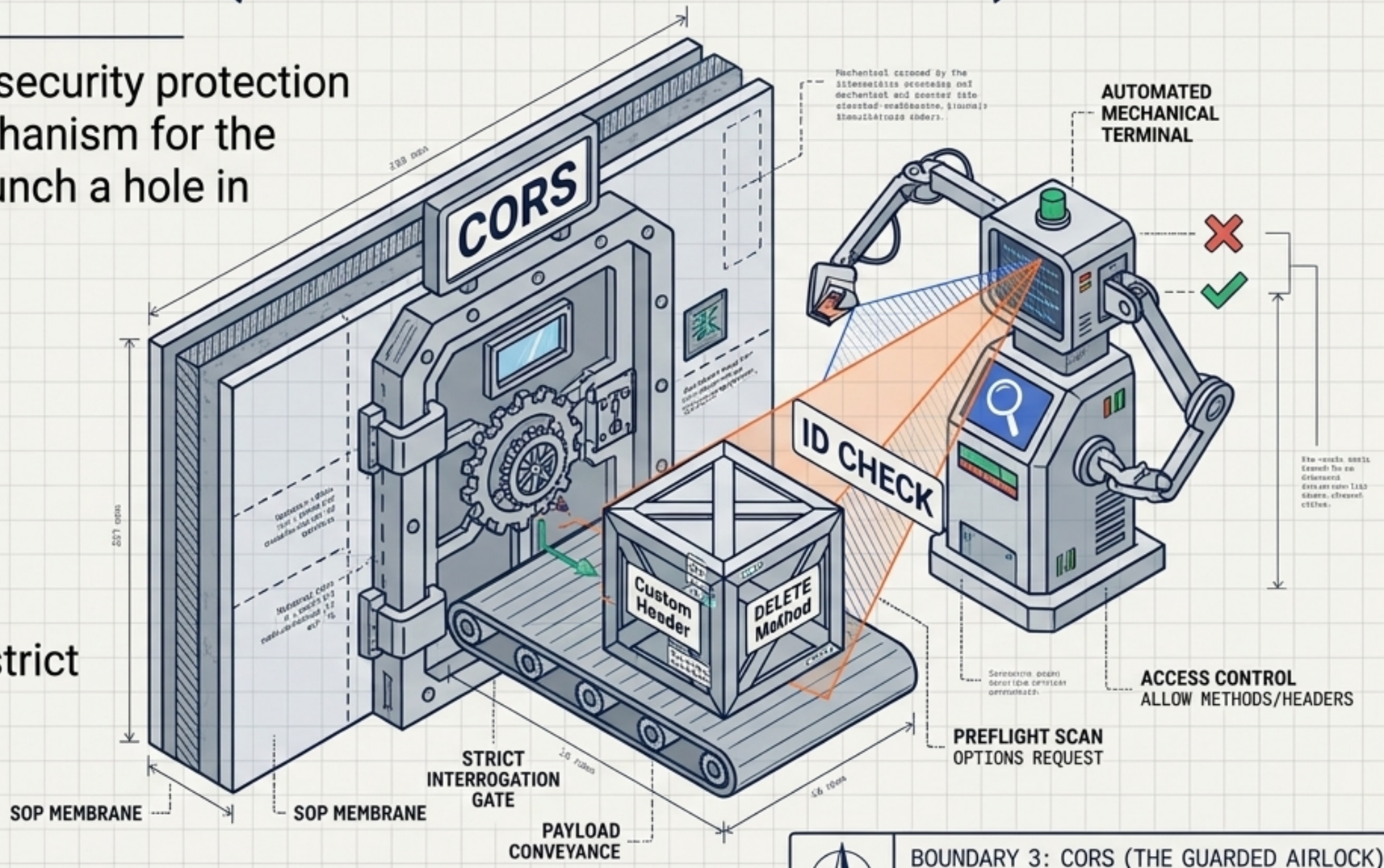
Gemini Notebook

Boundary 3: CORS (The Guarded Airlock)

Concept: CORS is not a security protection for the client. It is a mechanism for the server to intentionally punch a hole in the SOP membrane.

The Trigger:
"Simple" requests (standard GET/POST) bypass the guard.

"Complex" requests (custom headers, PUT/DELETE) trigger a strict Preflight interrogation.



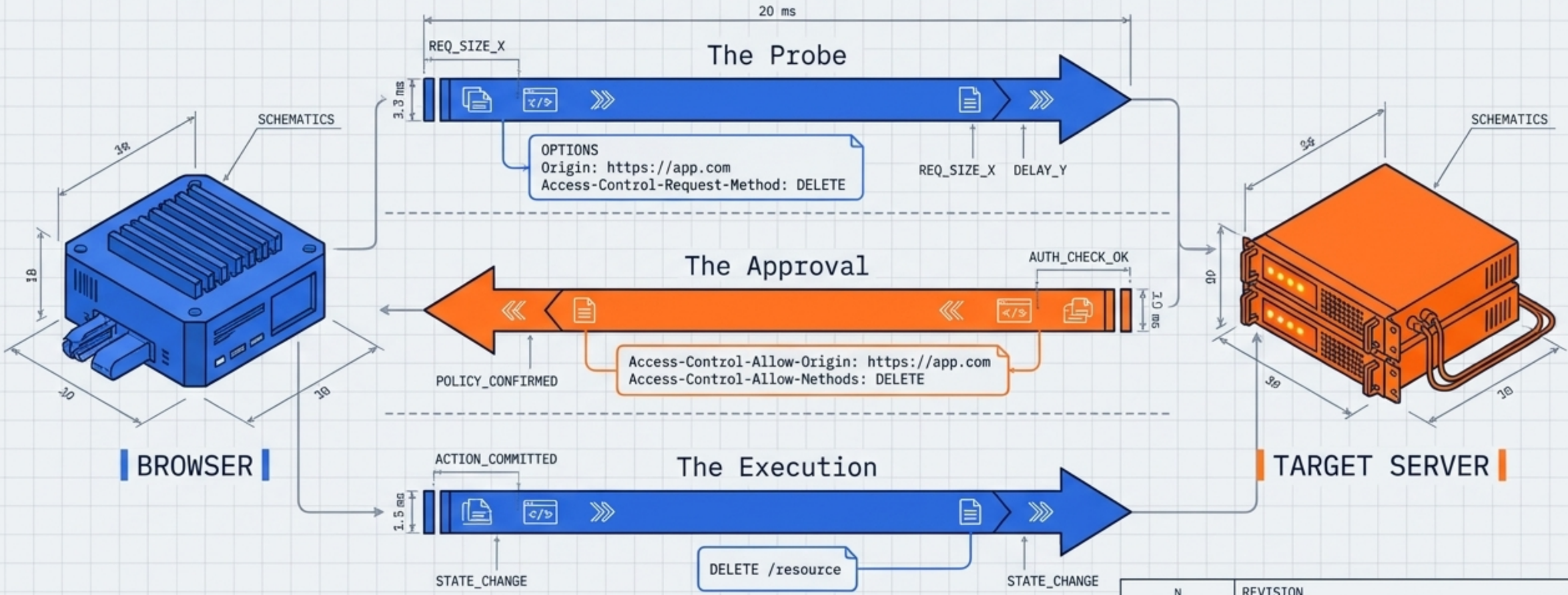
BOUNDARY 3: CORS (THE GUARDED AIRLOCK)

REVISION 3

Gemini Notebook

The Preflight Sequence

Rule: The browser holds the actual request hostage until the server explicitly authenticates the Origin via the OPTIONS exchange.

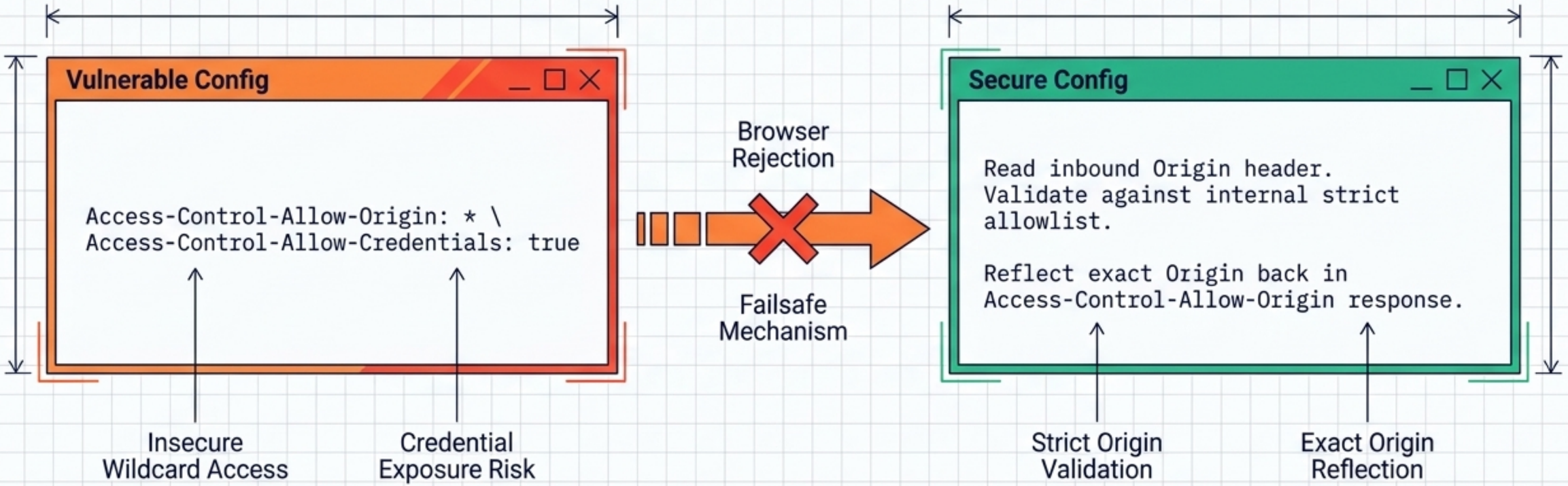


REVISION	
BOUNDARY 3: CORS (The Guarded Airlock)	REVISION 4

Gemini Notebook

Platform Blueprint: Secure CORS Configuration

The Browser Failsafe: Browsers will outright reject the vulnerable combination.
If you allow credentials (cookies), you must specify an exact Origin.

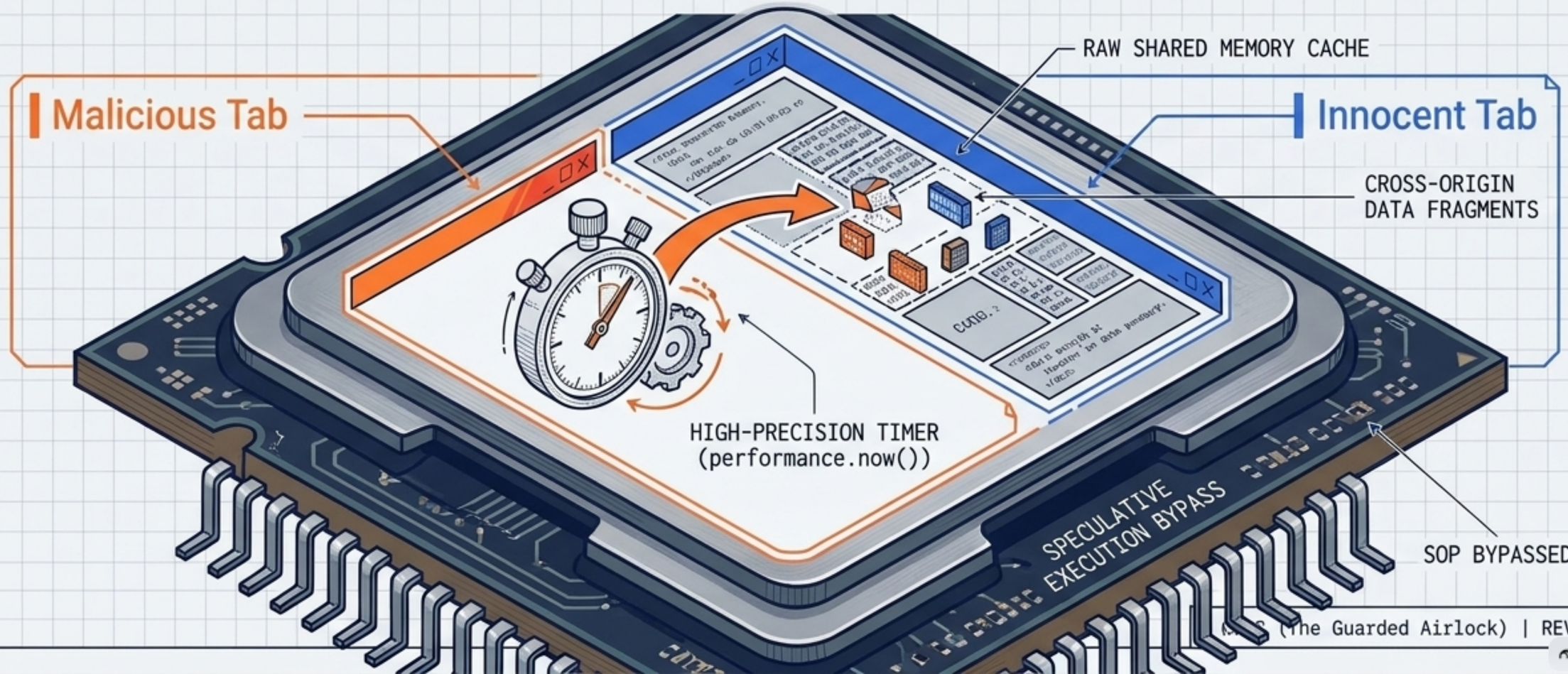


Gemini Notebook

Boundary 4: Cross-Origin Isolation (Process Quarantine)

The Catalyst: Spectre & Meltdown. SOP protects the DOM and network, but modern CPUs use speculative execution.

The Exploit: Attackers can use high-precision timers (`performance.now()`) and Shared Memory to read cross-origin data out of the browser's raw memory space, bypassing SOP entirely.



Gemini Notebook

The Alphabet Soup Decoded

Defense in Depth: COEP forces the lock; CORP provides the key.

Technical Annotation

Header	Target	What it Restricts
COOP (Cross-Origin-Opener-Policy)	Document	Prevents cross-origin windows (popups) from holding a DOM reference to your document.
COEP (Cross-Origin-Embedder-Policy)	Document	Blocks the document from loading any cross-origin resources unless they explicitly opt-in.
CORP (Cross-Origin-Resource-Policy)	Resource (Img/Script)	Protects a specific resource from being embedded by an attacker's cross-origin site.

Technical Annotations

Technical Annotation

Revision

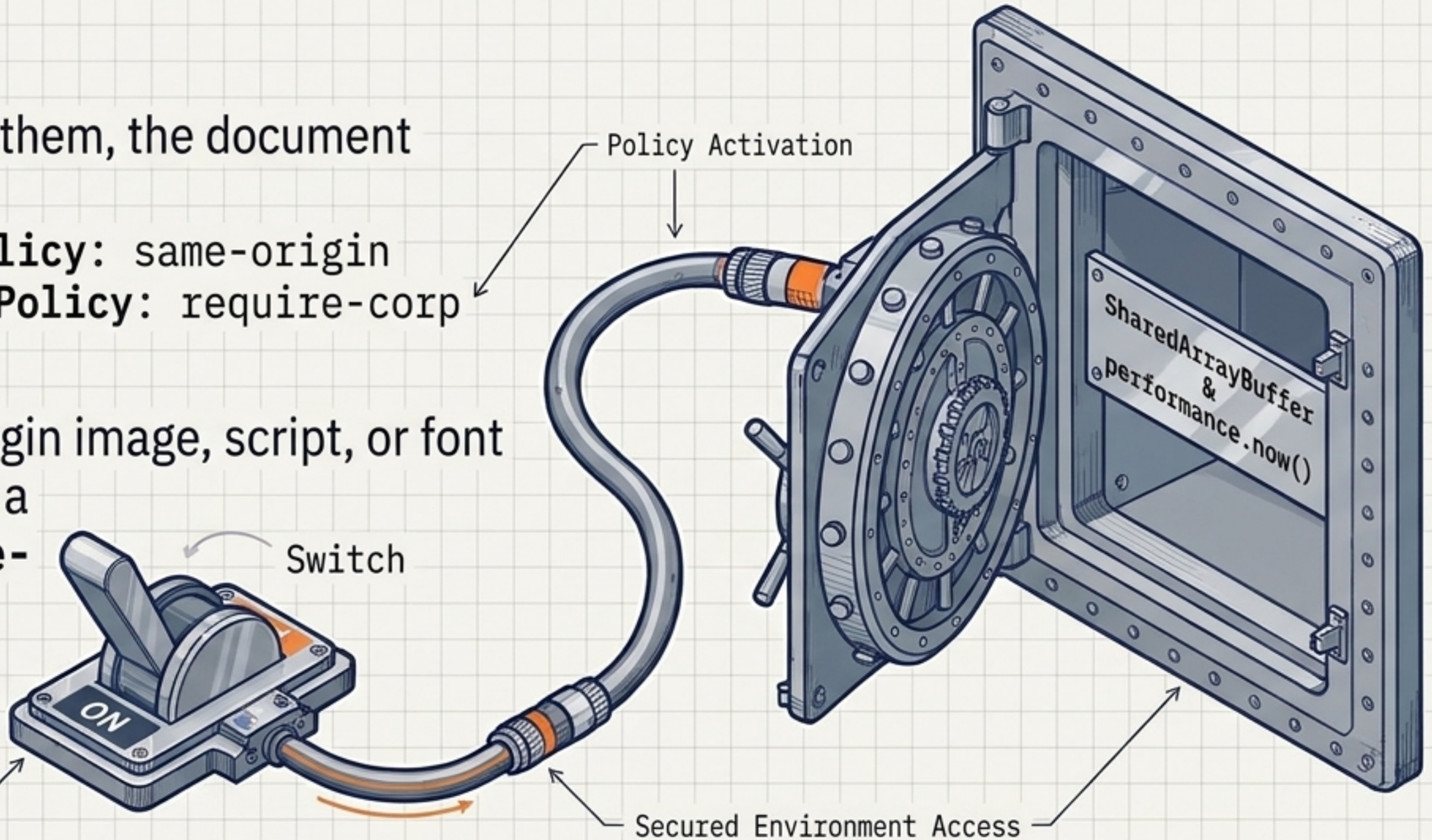
Gemini Notebook

Platform Blueprint: Achieving Process Quarantine

The Requirement: Browsers disable high-precision timers and **SharedArrayBuffer** by default to mitigate Spectre.

The Unlock: To re-enable them, the document must be served with:
Cross-Origin-Opener-Policy: `same-origin`
Cross-Origin-Embedder-Policy: `require-corp`

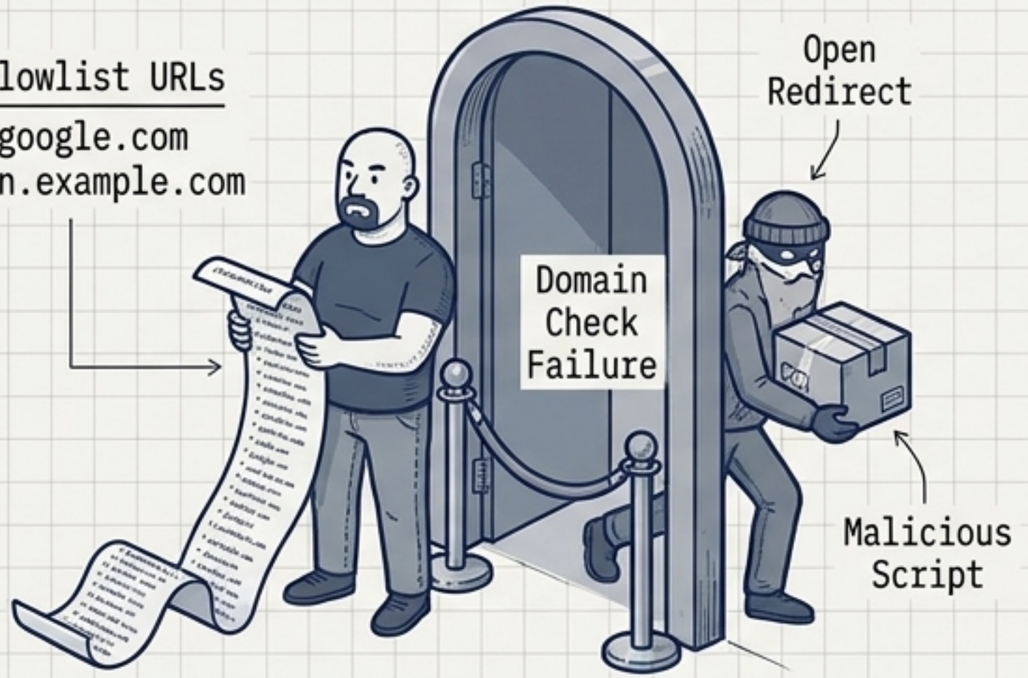
The Cost: Every cross-origin image, script, or font you load must now serve a **Cross-Origin-Resource-** cross-origin header, or the browser will block it.



Gemini Notebook

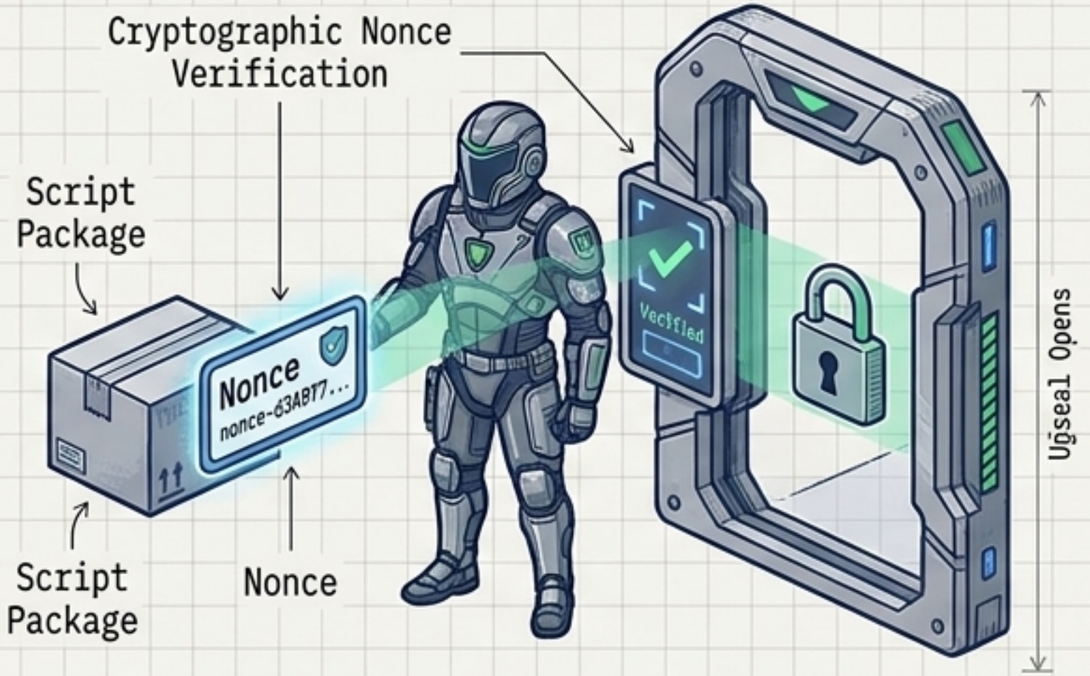
Boundary 5: Strict CSP (The Internal Security Guard)

Old CSP



The Old Way: Domain allowlists are dead. Research shows 95% of allowlist CSPs can be bypassed via JSONP endpoints or open redirects on allowed CDNs.

Strict CSP



The Modern Standard: Strict CSP abandons URLs entirely in favor of cryptographic **Nonces** (for dynamic pages) or **Hashes** (for static assets).

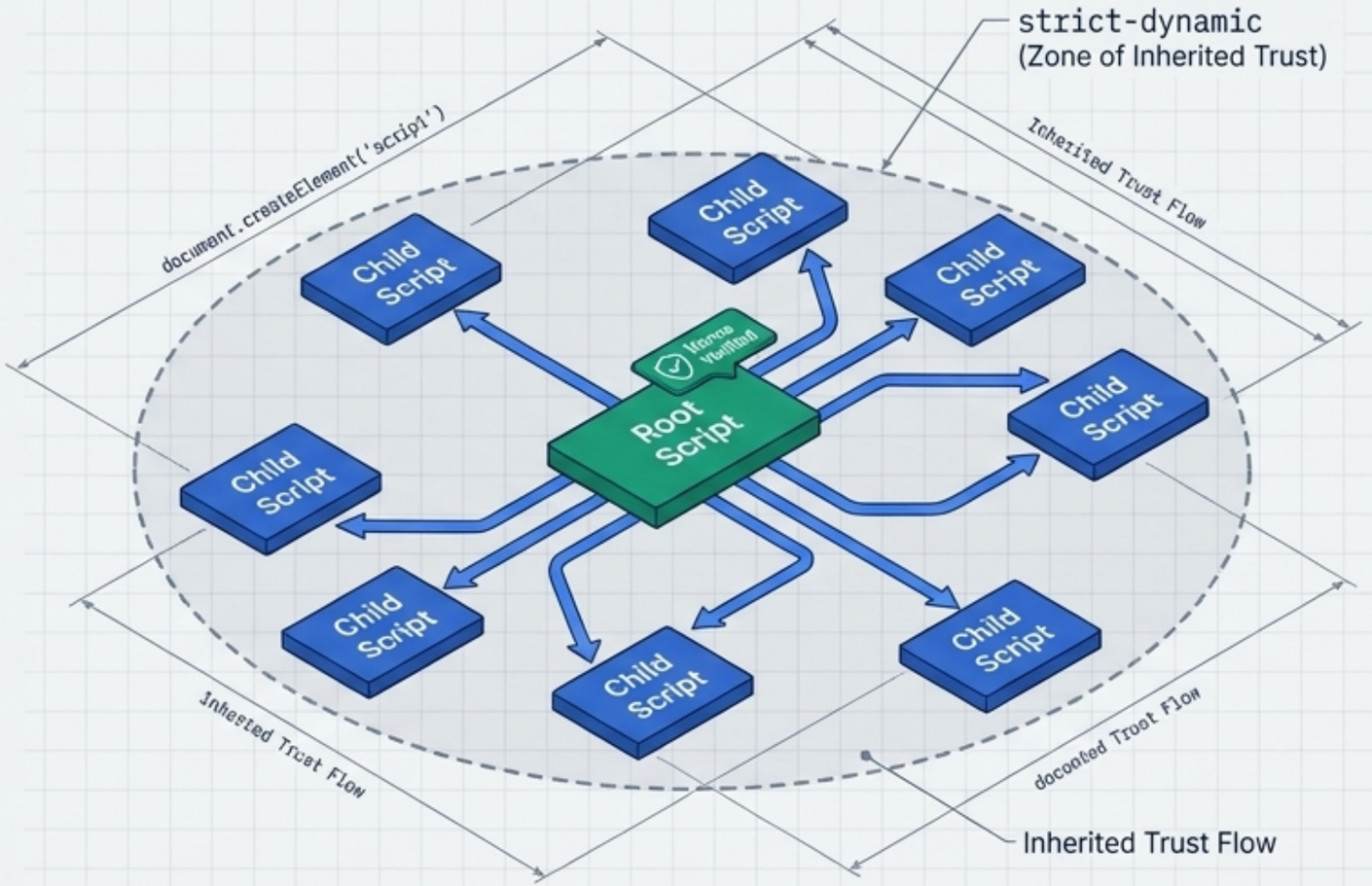
(Strict CSP Implementation) | REVISION 6

Gemini Notebook

The Power of `strict-dynamic`

The Problem:
Modern widgets (like Google Maps or Twitter embeds) dynamically inject dozens of child `<script>` tags.
You cannot nonce them all.

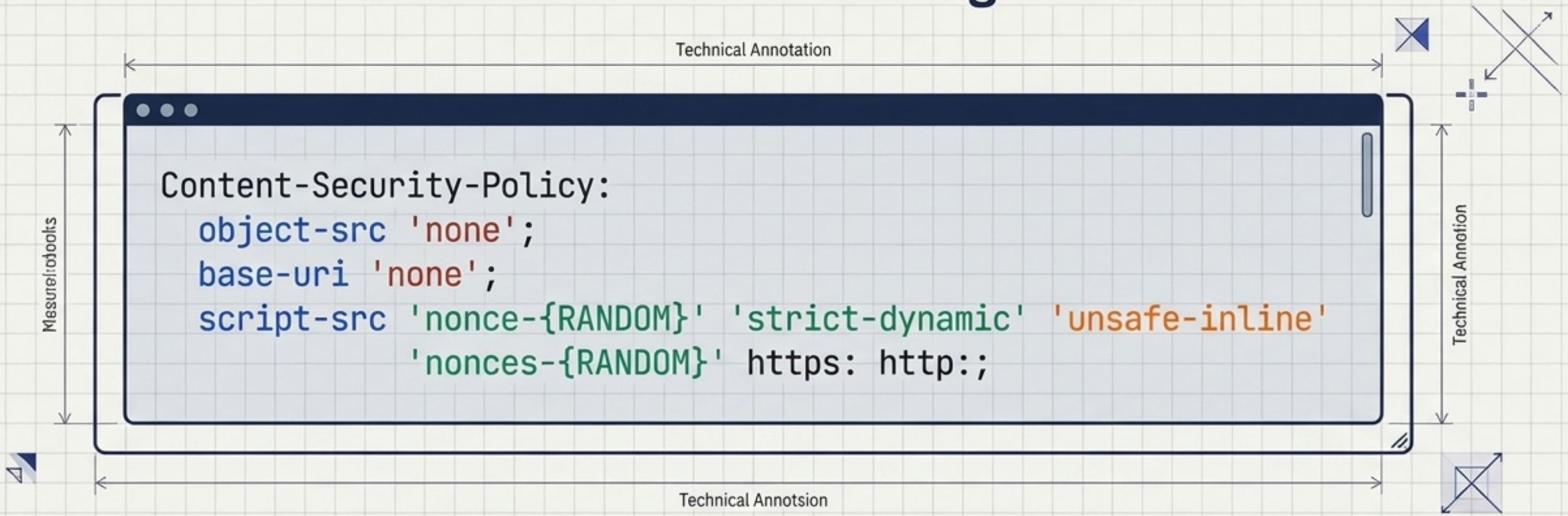
The Solution:
strict-dynamic tells the browser:
“If a script was trusted via a nonce or hash, automatically trust any new scripts it generates via non-parser-inserted methods (like `document.createElement('script')`**).”**



Gemini Notebook

Platform Blueprint: The Strict CSP Header

The Golden Standard Configuration



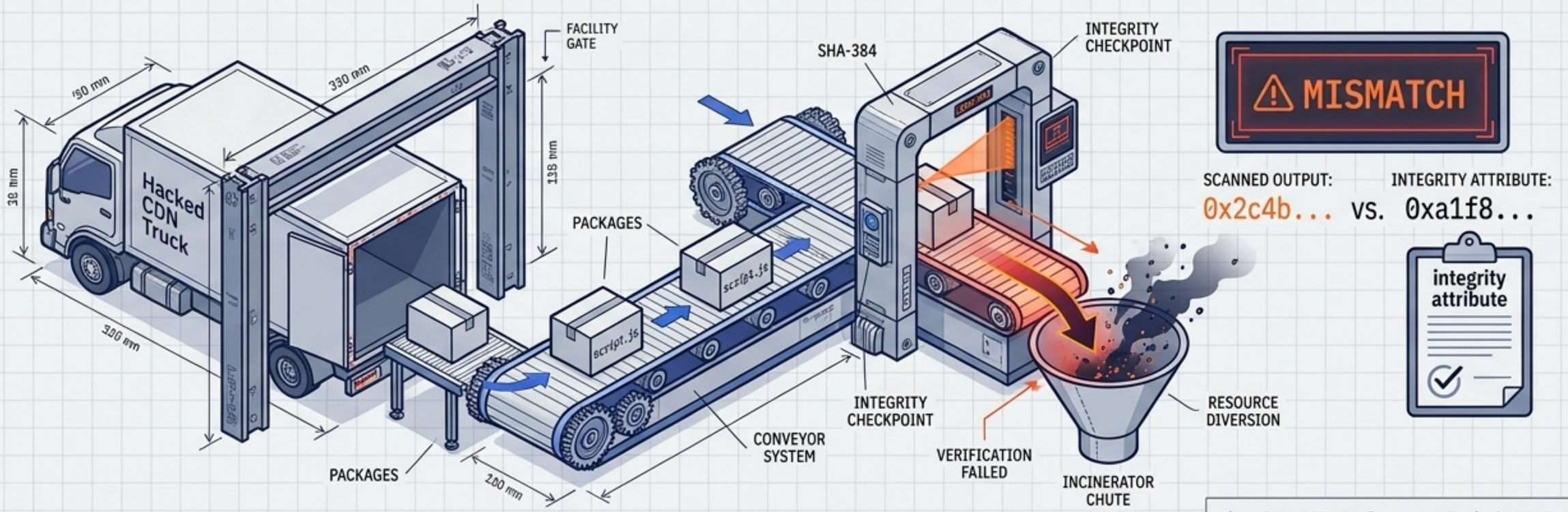
Backward Compatibility: Modern browsers see strict-dynamic and ignore ‘unsafe-inline’ and the http: fallbacks. Older browsers ignore strict-dynamic and fall back to the URL allowlist.

Gemini Notebook

Boundary 6: Subresource Integrity
(The Tamper-Evident Seal)

The Threat:
You trust a CDN via CORS and CSP.
The CDN gets hacked and silently replaces analytics.js with a keylogger.

The Defense:
SRI enforces cryptographic verification on the client side. If a single byte of the external resource changes, the browser refuses to execute it.

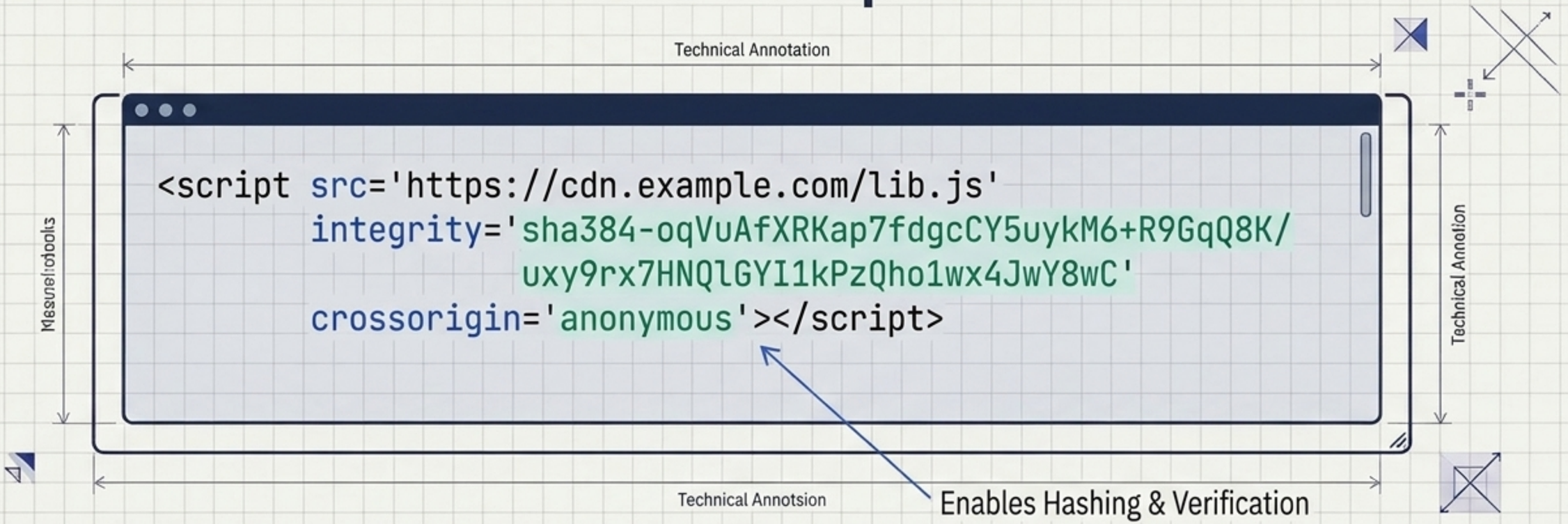


(Strict CSP Implementation) | REVISION 9

Gemini Notebook

Platform Blueprint: SRI Implementation

The Markup

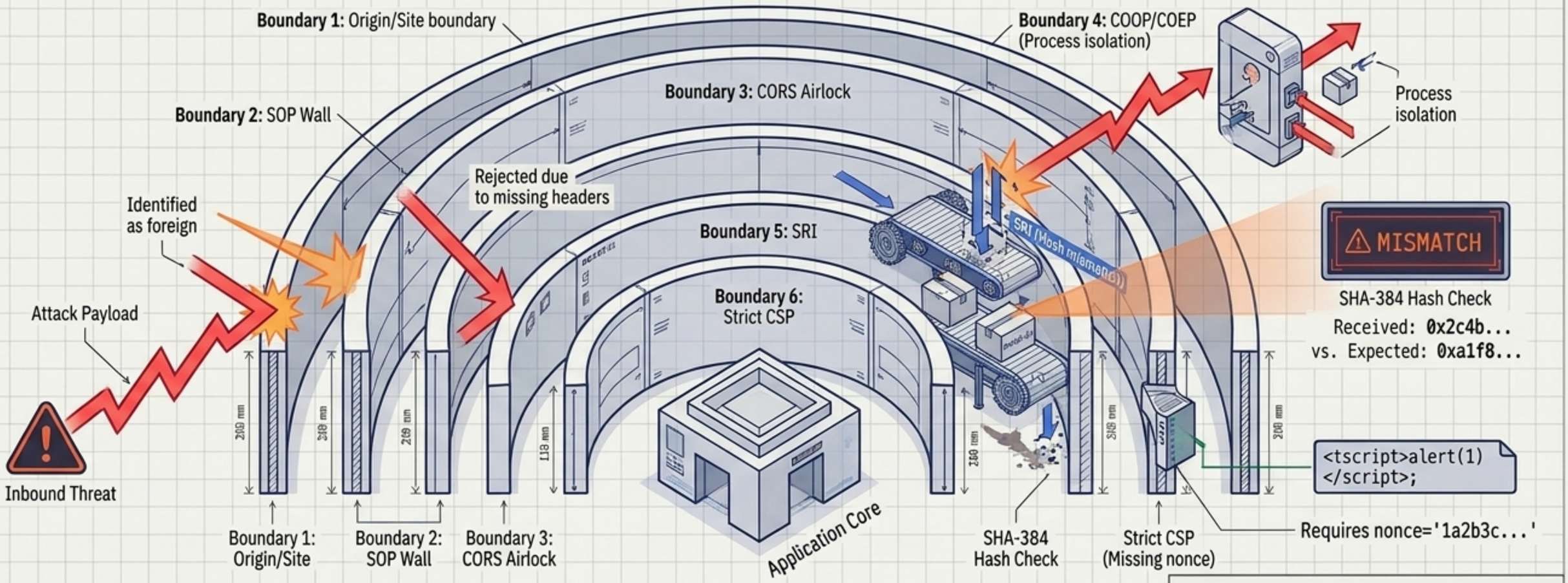


The CORS Requirement: You cannot hash an opaque response. By default, cross-origin scripts load in no-cors mode. To read the bytes and verify the hash, you must include **crossorigin='anonymous'**, and the CDN must respond with **Access-Control-Allow-Origin: ***.

Gemini Notebook

Browser Defense-in-Depth Architecture

Takeaway: No single header provides perfect security. Modern web architecture relies on overlapping perimeters. An attacker must bypass the foundational Site boundaries, navigate SOP asymmetry, trick the CORS airlock, evade process quarantine, forge a cryptographic nonce, and crack a SHA-384 hash to compromise a hardened application.



(Strict CSP Implementation) | REVISION 11

Gemini Notebook