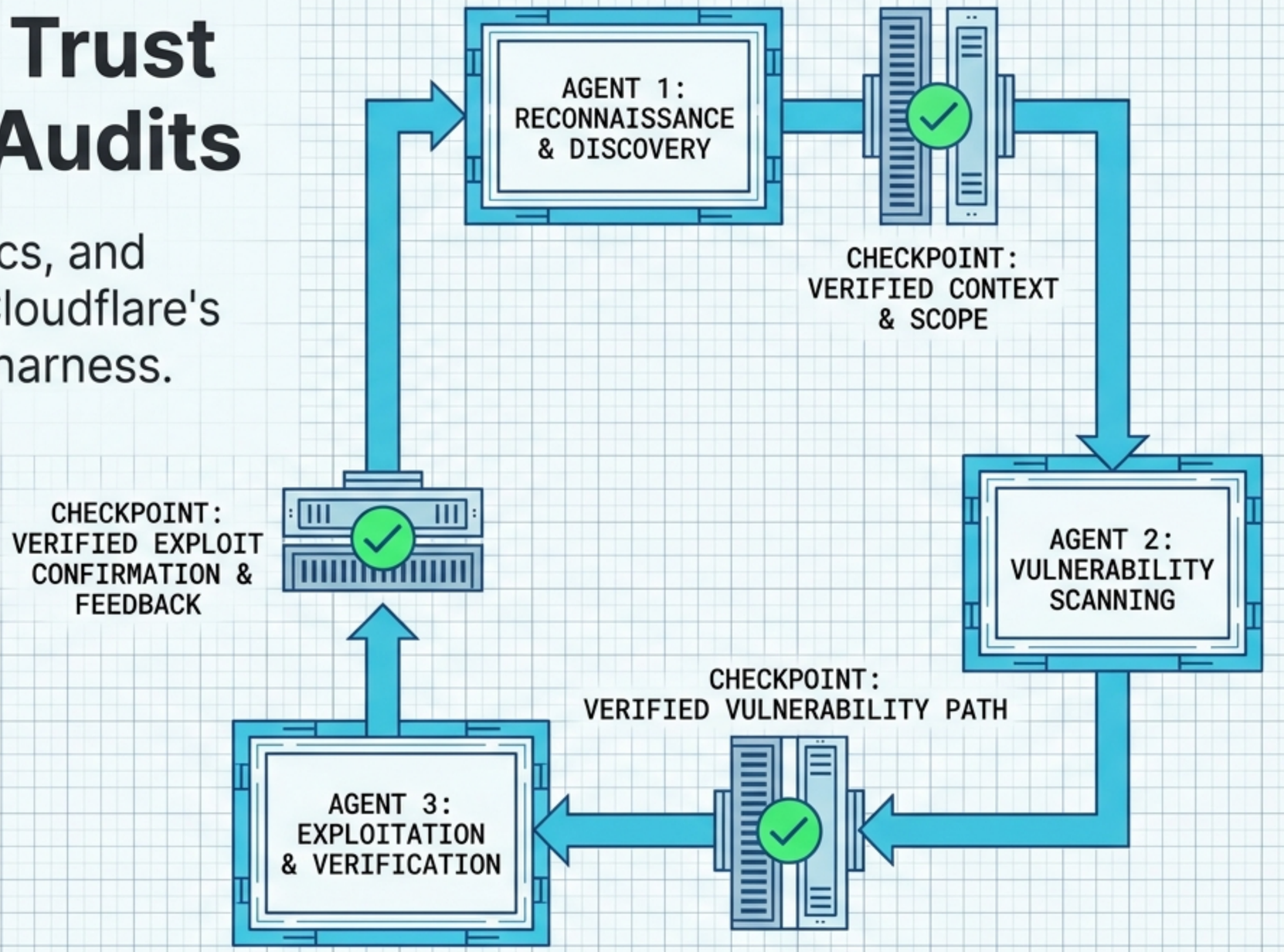
Orchestrating Trust in AI Security Audits

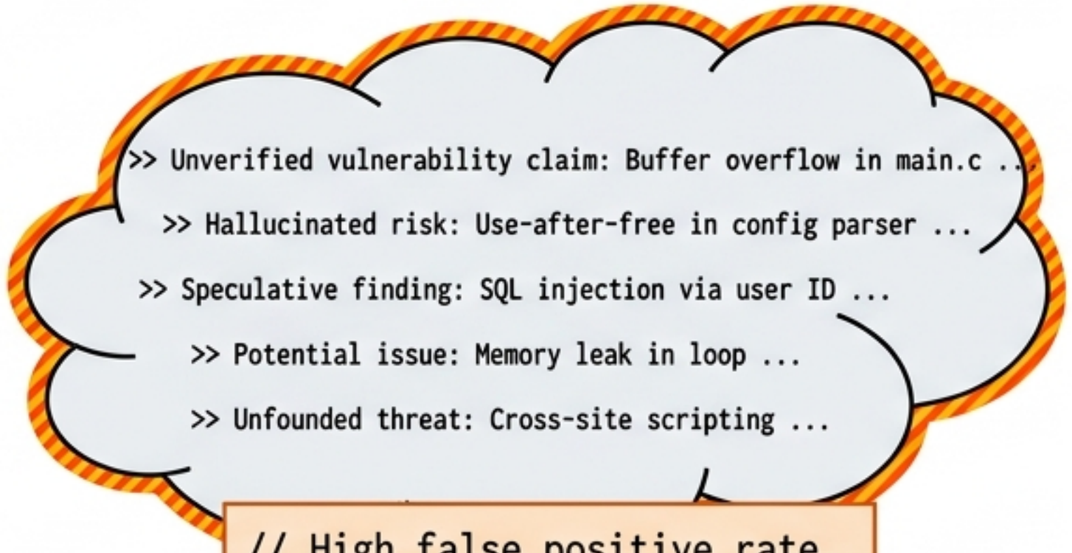
The architecture, mechanics, and design principles behind Cloudflare's multi-phase vulnerability harness.



Gemini Notebook

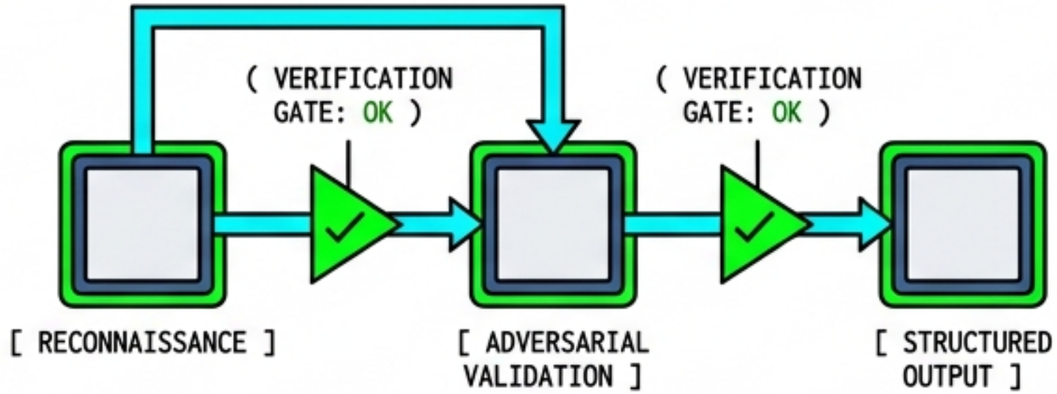
Machine-driven audits require deterministic boundaries

The Naive Approach



// High false positive rate.
Unverifiable claims.

The Orchestrated Skill



// Bounded execution.
Independent verification.

The **security-audit-skill** is an open-source orchestration engine that transforms a standard coding agent into a rigorous security auditor by enforcing **isolated reconnaissance**, **adversarial validation**, and **structured output**.

Gemini Notebook

Five principles that eliminate AI hallucinations



Established Boundary Failures

Only confirm proven breaks. Keep blocked leads as `needs_validation` with exact unresolved facts.



Adversarial Validation

The agent that checks a finding is never the agent that found it.



Impact-Driven Severity

Likelihood x Impact.

Deviation from a checklist does not equal severity.



Defense-in-Depth Reality

If Layer A prevents the attack, the absence of Layer B is a hardening note, not a vulnerability.



Additive Coverage

Multiple runs improve coverage. Single runs only catch roughly 50% of what repeated runs uncover.

Gemini Notebook

The Validation Paradigm Matrix

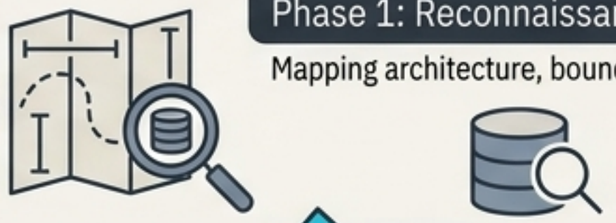
System Dimension	Traditional AI Prompting	Skill Orchestration (Cloudflare)
Actor Count	Single omniscient agent	✅ **Parallel, isolated sub-agents**
Verification	Self-checked (high hallucination)	✅ **Adversarial (Checker is not Finder)**
Output Format	Unstructured Markdown text	✅ **Immutable JSON schemas (`findings.json`)**
Code Coverage	Random / Attention-biased	✅ **Deterministic ledger (`coverage-ledger.json`)**

Gemini Notebook

The 6-Phase Engine

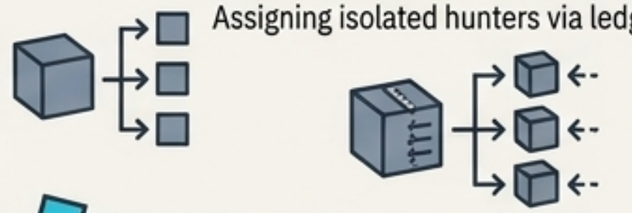
Phase 1: Reconnaissance

Mapping architecture, boundaries, prior evidence



Phase 2: Coverage-led hunting

Assigning isolated hunters via ledger units



Phase 3: Candidate validation

Giving unique candidates to fresh verifiers



Phase 4: Structured output

Writing records to JSON schemas



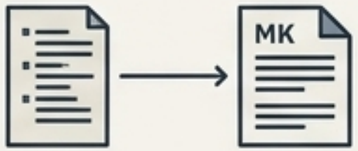
Phase 5: Independent record verification

Fresh agents verifying final claims



Phase 6: Target-neutral reporting

Deriving Markdown reports

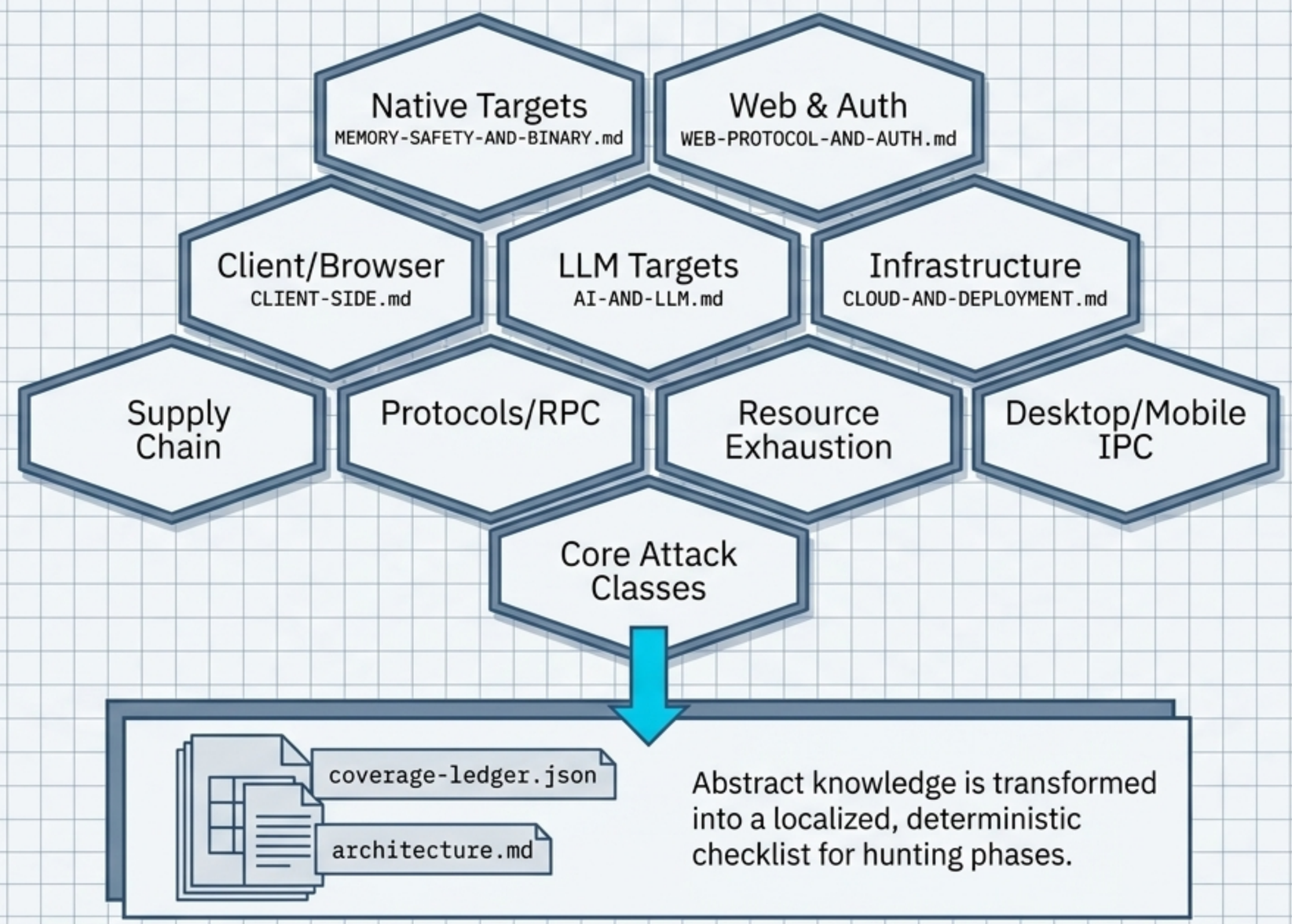


validate-findings.cjs
validate-coverage-ledger.cjs

Zero-dependency Node.js validators acting as ultimate referees.

Gemini Notebook

Mapping the attack surface via coverage ledgers

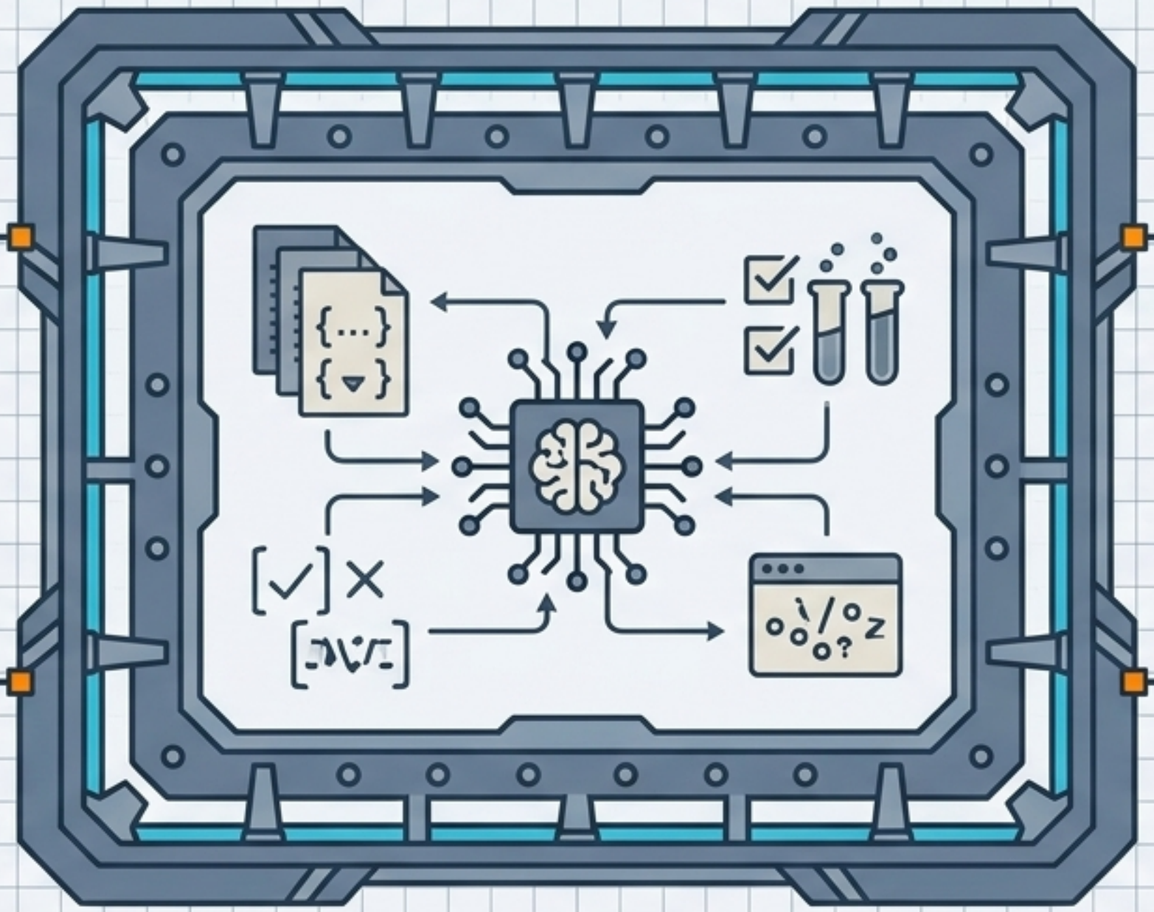


Gemini Notebook

Enforcing reality through the Adversarial Sandbox

No external networking

Enforced resource limits



Sanitized allowlisted environment

Writes restricted only to assigned scratch paths

Without these OS-enforced controls, the workflow is physically incapable of executing target code, forcing all unverified leads to safely default to `needs_validation`.

Gemini Notebook

Hunter vs. Verifier: The Adversarial Hand-off

Agent A: The Hunter



Discovers a vulnerability candidate. Strips away all context except the raw candidate data.

RAW DATA:
CANDIDATE BYTES

```
02DE 001011001
02DE 931000100
02DE 003010100
02DE 031010100
02DE 010011121
02DE 003010100
02DE 831010100
02DE 031010101
```

NO CONTEXT

FILTERED
RAW DATA

```
02DE 001011000
02DE 831010100
02DE 010011131
02DE 001010100
02DE 031010160
```

Agent B: The Verifier

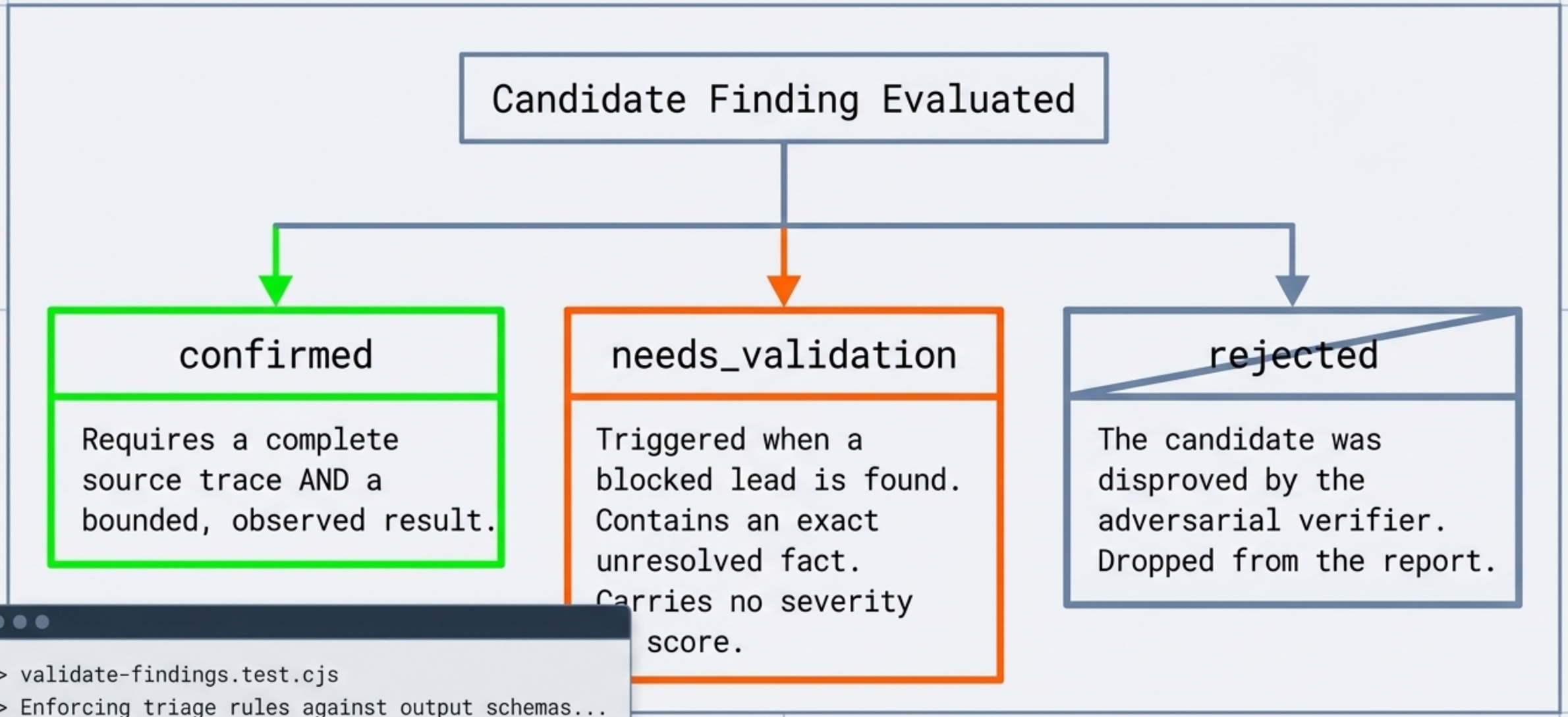


Completely isolated environment. Receives a dedicated, antagonistic prompt: Disprove this finding.

⚠ Material replacements trigger yet another independent verifier. The system is designed to kill its own darlings.

Gemini Notebook

The Verdict Triage System

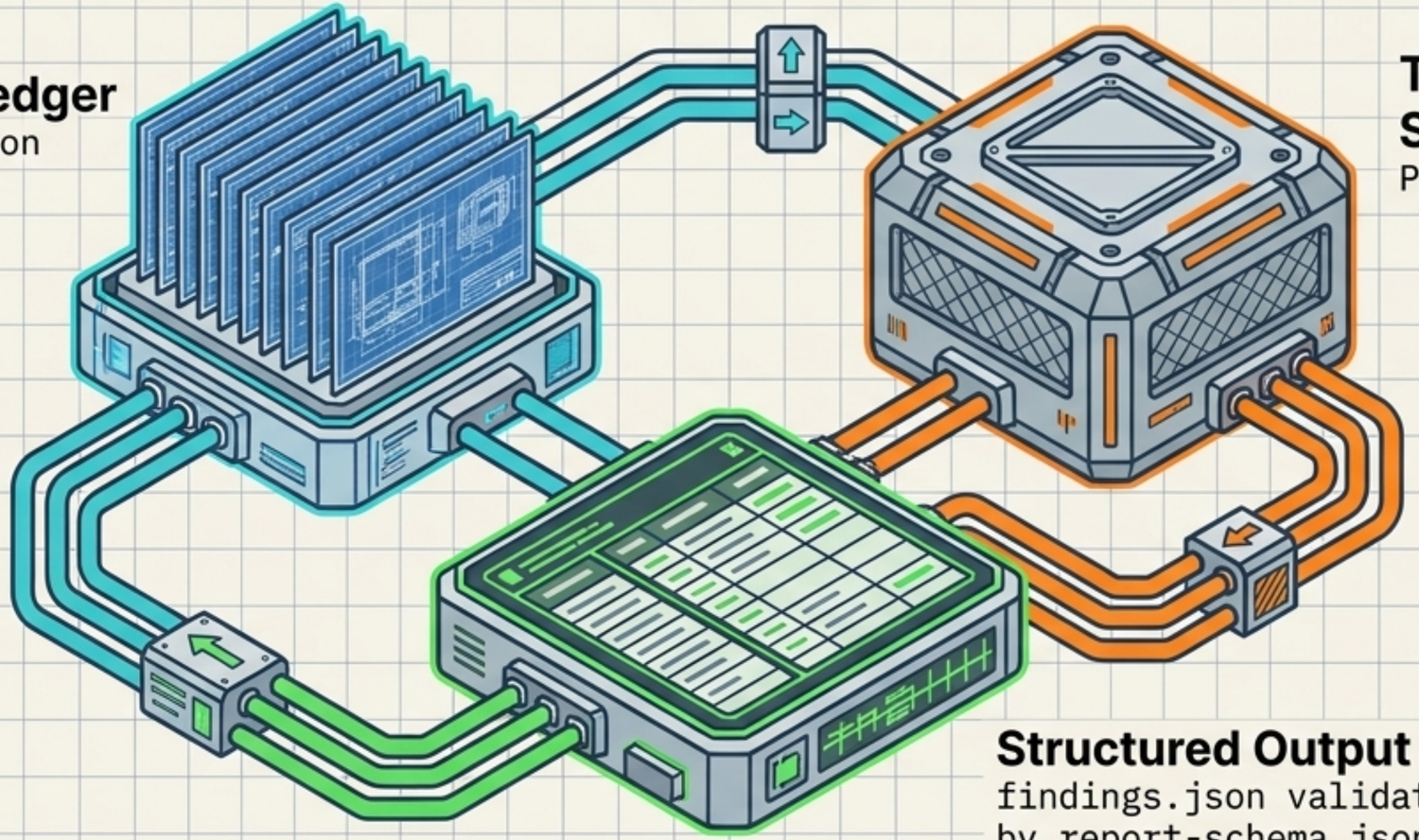



```
> validate-findings.test.cjs
> Enforcing triage rules against output schemas...
```

Gemini Notebook

The Self-Correcting Ecosystem

The Coverage Ledger
`coverage-ledger.json`
`from Phase 1`



The Adversarial Sandbox
`Phases 2/3/5`

Structured Output
`findings.json validated`
`by report-schema.json`

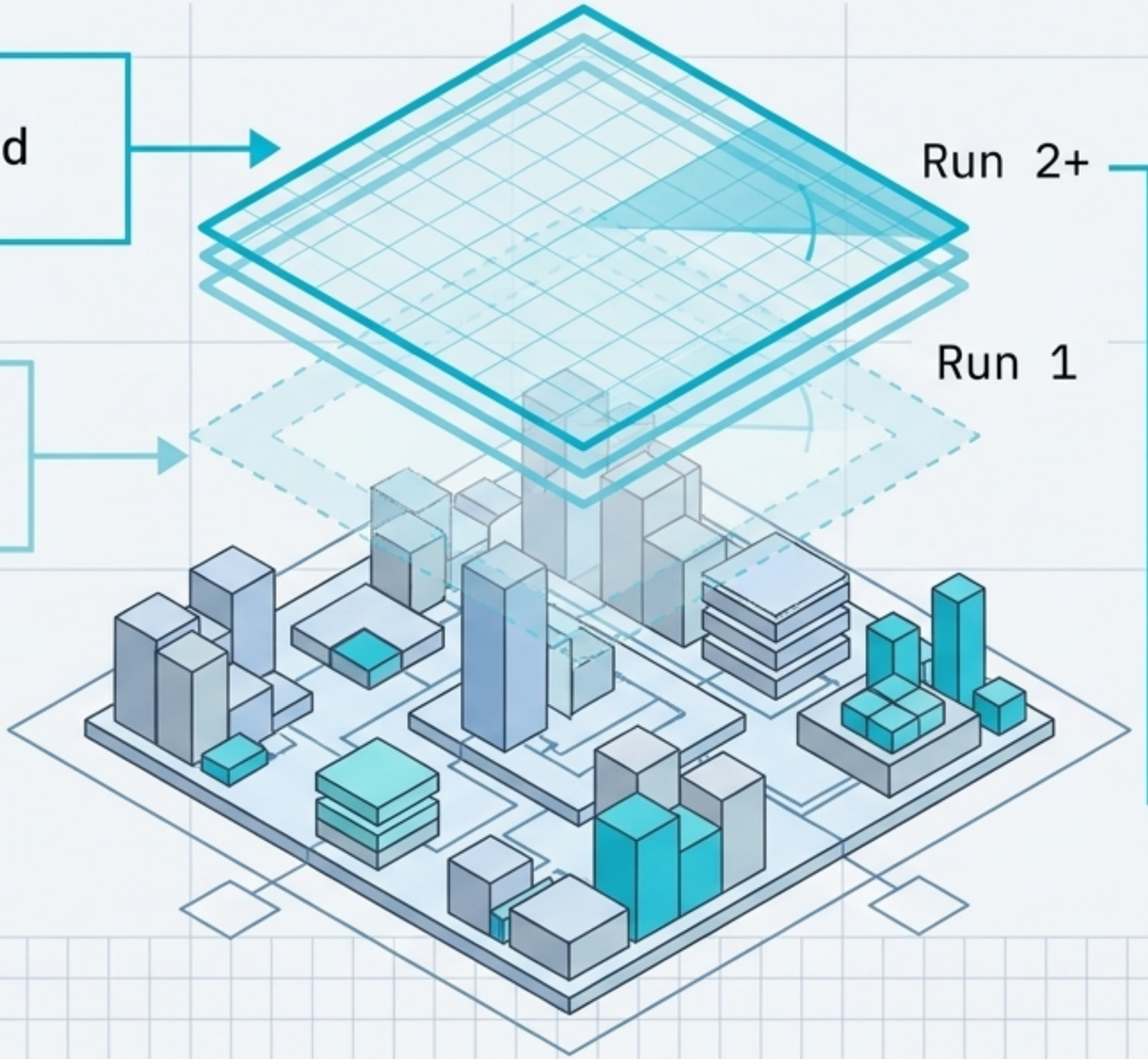
The magic isn't in the AI's ability to hunt. It is in the orchestration. Machine-readable schemas act as the immutable laws of physics—forcing agents to peer-review each other and translating raw deterministic ledgers into target-neutral Markdown (`REPORT.md`, `FINDINGS-DETAIL.md`).

Gemini Notebook

Continuous execution yields additive coverage

Prior ledgers and findings are carried forward.

Identifies approx. 50% of total vulnerabilities.



Run 2+

- Targets new gaps.
- Revalidates changed source code and carries forward current-source evidence.
- **Crucially: Never treats stale or unresolved work as covered.**

Gemini Notebook

Two distinct operating modes

Full Audit Mode

Trigger

> security audit this codebase
> do a security review

Behavior

Direct codebase audit or pen-test execution. Uses all 6 phases.

Output

Generates full artifact directors full artifact directories. Defaults to:

out: ~/security-audit-skill/<repo-name>/run-<N>

Guidance Mode

Trigger

> Asking security questions

Behavior

Contextual, interactive analysis. Skips full ledger generation and reporting file writes.

Output

In-chat responses and localized code suggestions without writing to disk.

Gemini Notebook

Deploying the Audit Skill

```
// Install via Skills CLI
npx skills add https://github.com/cloudflare/security-audit-skill \
   --skill security-audit \
   --global
```

Requirement Checklist

- ✅ Parallel sub-agent support
- ✅ Node.js installed
- ✅ OS-enforced sandbox ready

Open-source orchestration that turns an agent into a verifiable, multi-phase auditor.

Gemini Notebook