HEXAWARE

Whitepaper August 2026

# Your code scanners' blind spot – measured

Author:
**Satyajith Mundakkal, CTO, Hexaware,**
with Agalya Mythily TM and Gowtham C P,
Companion to "Mythos, Measured"

RESEARCH PAPER · AI & CYBERSECURITY

# Your code scanners' blind spot — measured.

Nine detectors, four approaches, three codebases — and only 14 of 334 retained candidate records overlapped.

We ran nine detector configurations across four detection paradigms — traditional scanners / static-dataflow analysers, general-purpose / code-capable models, a security-specialised model, and a tool-augmented review workflow — over three Python/FastAPI–React codebases. Their retained candidate findings overlapped surprisingly little. That supports layered discovery; it does not, on its own, establish which approach is best.

By Satyajith Mundakkal, CTO, with Agalya T, Mythily T M, and Gowtham C P | Companion to "Mythos, Measured" | ~35 min read

---

STATUS — EXPLORATORY

This is an exploratory comparative study of detector *outputs*, not a benchmark of detector accuracy. Unequal output sampling and the absence of blind adjudication prevent precision, recall, and detector-ranking conclusions; a separate 262-finding product-owner review does supply developer-validated false-positive rates, reported as scoped, non-blind results. Results are described as **retained consolidated candidate records**: one row in the canonical 334-row master after de-duplication/sampling, before validation, scoped to the tested configurations. The 334 rows encode 349 detector contributions because 14 rows carry multi-detector support. Data freeze: 18 Jun 2026.

FOR THE DECISION-MAKER

If you read nothing else — the three things this study will support in front of a skeptic.

1. **Layer; don't pick a winner.** Of 334 consolidated master records — 349 detector contributions once the multi-detector rows are counted — only 14 carried more than one detector. The four approaches mostly see different things, so coverage comes from combining methods — not from choosing one model.

2. **Keep your existing scanners.** The database-backed lane supplied nearly all the named dependency-advisory identifiers represented in the canonical master (pending CVE reconciliation) — the kind of evidence your compliance and SBOM processes depend on. No AI approach replaced that lane.

3 **The bottleneck is downstream of discovery.** More candidates do not help if validation and remediation cannot keep up. Instrument time from validated finding to verified production fix before you add AI discovery.

**Do:** build the layered discovery-and-verification workflow (Section 10); measure remediation throughput first (Section 11). **Don't expect here:** a "best detector" verdict or a product recommendation — the data supports neither (see Status, above).

## THE SHORT VERSION

**What this is.** An exploratory comparison of retained candidate findings and their overlap across nine detector configurations on three Python/FastAPI–React codebases. The Status box above defines what it does and does not measure. A separate, non-blind 262-finding product-owner review reports scoped false-positive rates (Section 2).

**What changed our recommendation.** In this run, the four approaches produced substantially different retained findings — coverage emerged from the whole detection system, not the model label. (The thesis below states this in full.)

**The overlap result — the strongest, most defensible result.** Under strict record matching, only **14 of the 334** consolidated master records (349 detector contributions) carried multi-detector support. Under a broader repository–CWE–category grouping, **35 of 145** groups involved two or more detectors. Low observed overlap — though the broader grouping can combine separate vulnerabilities.

**What it does not establish.** Output samples were unequal — the three general-purpose models were each truncated to 50 retained rows — and blind adjudication is still pending. The study supports no detector ranking and no precision or recall comparison. A separate 262-finding product-owner review does provide developer-validated false-positive rates (Section 2), scoped and non-blind, on a different record set than the 334 master.

**The operational implication.** Do not replace established scanning with a model. Build a layered discovery-and-verification workflow, then measure validated risk reduction, analyst effort, and time to production remediation — the question the companion brief, *Mythos, Measured*, is really about.

CENTRAL THESIS

The experiment does not establish which detector is best; the current data cannot answer that. Its defensible contribution is narrower. Under the tested configurations, different detection systems produced substantially different retained findings. **Coverage is a property of the whole detection system — its analysis method, context, tooling, vulnerability intelligence, prompting, and verification — rather than of the model name.** Low overlap between candidate outputs makes layering rational. The layer that matters most operationally is the one that converts a candidate into a validated, fixed, regression-tested, deployed risk reduction.

**What surprised us.** We expected the detectors to disagree. We did not expect the strict overlap to be this low: after consolidation, only 14 of 334 master records had more than one detector behind them. That shifted the recommendation. What began as a "which approach is best" question became a workflow-design one, and the honest answer stopped being "replace your scanner."

CONTENTS



WHY NOW

**Security detection is changing with repository-scale Mythos-class models.** For years, vulnerability detection meant a fixed toolchain: SAST, dependency and secrets scanners, dataflow analysis, run on a set cadence. Capable code models changed the ground under that: they can read a whole repository and reason about it in ways a rule-based scanner cannot. That raises the question every security team now faces: has detection fundamentally changed since Mythos, or has it simply added a new lane? And what does LLM-based detection actually add to the stack you already run? This study set out to map that gap on real production code: the space between what the traditional

toolchain catches and what LLM-based detection surfaces. It is a gap-mapping exercise, not a contest. We compared what each approach found, and they mostly found different things. The gap is visible in this run; the response is to layer the old and the new, not to replace one with the other.

### WHAT WE SCANNED

Three production codebases on a roughly constant stack (Python / FastAPI back end, React front end). Bars show relative *size / footprint* — lines of code, not a complexity metric.

#### APP A · SMALL FOOTPRINT

**~20.5k** lines

235 files · 85 deps · 0 test files · single Dockerfile.

#### APP B · MEDIUM FOOTPRINT

**~52.4k** lines

390 files · 294 deps · 84 tests · SQLAlchemy 2.0.

#### APP C · LARGE FOOTPRINT

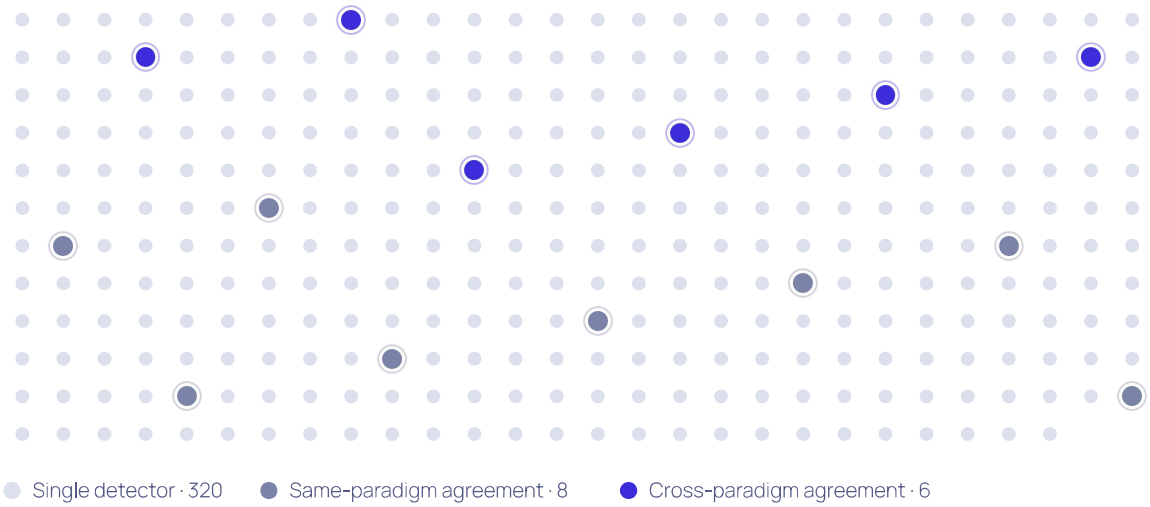
**~106.5k** lines

574 files · 438 deps · 121 tests · K8s manifests (count in dispute)†.

† Source conflict on the App C manifest count — see Appendix D.

THE SIGNATURE RESULT — 14 OF 334

Each dot is one of the 334 consolidated master records, coloured by how much the detectors agreed: light for a single detector (320), mid-tone for a same-paradigm agreement (2+ detectors from one paradigm; 8), and bold for a cross-paradigm agreement (2+ detectors from different paradigms; 6). Only **14 of the 334** carried more than one detector at all, so the four approaches mostly saw different things. That is why layered detection matters.




Dot position is arbitrary; colour is the only variable that encodes data (construction: Appendix A).

Low overlap means different coverage, not noise — each detector caught records the others missed. False positives are measured separately (Section 2), not implied by this chart.

**334**
consolidated master records

**14**
records by 2+ detectors (strict)

**35 / 145**
groups by 2+ detectors

**9**
detector configs · 4 paradigms

**3**
codebases · one stack

01 · WHY THIS STUDY

## What to add to an existing AppSec program

The companion brief, **Mythos, Measured**, argues that AI is collapsing the time and skill required to *discover* vulnerabilities while the machinery to *fix* them stays on human pace. That is a strategic hypothesis from the brief, not something three repositories can establish; we treat it as motivation, not as a result of this experiment.

The practical question for a CXO is narrower and answerable in part: **"I have traditional application security testing today. What kinds of findings does each newer approach add, and how much do they overlap with what I already run?"** This study addresses that in terms of retained candidate output and overlap, not accuracy.

02 · METHOD

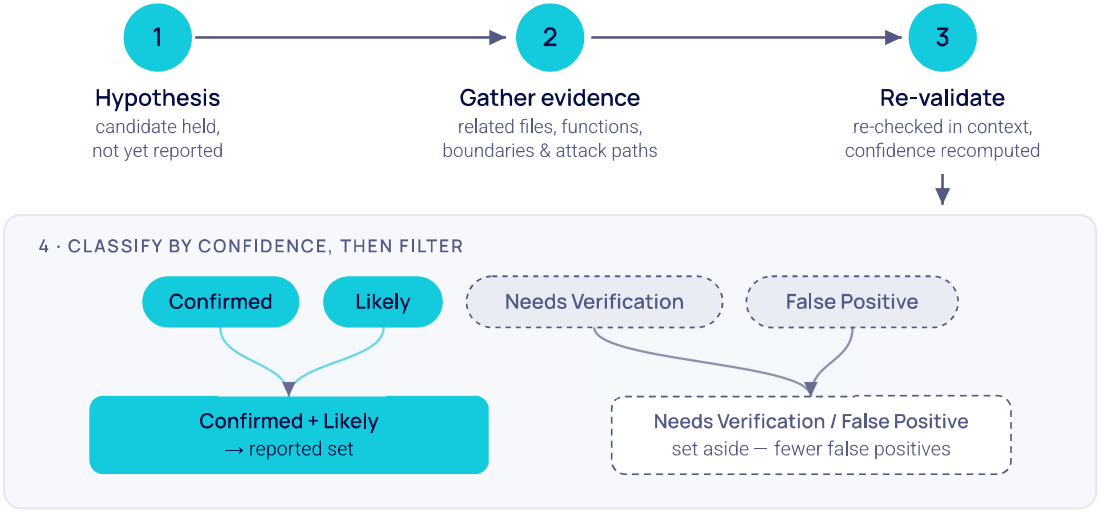
## Four classes of detector, on three codebases

Three codebases vary size while holding the stack roughly constant. Nine detector configurations fall into four paradigms, colour-coded throughout. Treat the paradigm labels as an organising convenience: the experiment does not isolate whether a result is due to the base model, the harness, the tools, database access, context selection, the prompt, or a verification loop. The four-way split is a useful lens for describing *what* each kind of detector surfaced, not a claim that the label itself *caused* the difference.

| PARADIGM | DETECTORS (THIS RUN) | WHAT IT IS |
|---|---|---|
| Traditional | Semgrep, Trivy, SonarQube, GitLeaks | Database-backed, rule-based, and static/dataflow analysis depending on product and configured edition. Generally reproducible for a fixed version, ruleset, database snapshot, and command. |
| General LLM | Qwen3-Coder-30B, GPT-5.3-Codex, grok-4.3 | General-purpose / code-capable models given repository context and run through a hypothesis-and-evidence workflow (see note): each candidate is re-checked against related code, trust boundaries, and attack paths, then confidence-classified. No integrated security database or runtime execution. Grok 4.3 was tested here as a general model applied to code review; xAI documents grok-4.3 as its general model and points to a separate dedicated coding model for code-specific use. |
| Security-trained | Fable 5 (Mythos-class) | A security-specialised model. A single model represents this paradigm here, so its results cannot establish a class effect. |
| Hybrid | Claude Code Security (research preview) | Model reasoning plus repository navigation, execution, history, and verification tooling. |

### The general-LLM scanning pipeline

Each candidate is treated as a hypothesis, evidenced, re-validated, and confidence-classified; only sufficiently-evidenced findings reach the reported set. Workflow labels such as *Confirmed* and *Likely* are evidence-confidence labels, not adjudicated truth.




False positives in this study were controlled in **two independent reduction layers: one automated, one human**. Neither alone is sufficient; the paper reports both.

**Layer 1 — automated, before the master.** Before any finding reached the candidate master, every detector ran a documented false-positive-reduction step: multi-pass scanning with automatic de-duplication and a mandatory verbatim-code-quote-plus-attack-scenario rule for the general-purpose LLMs (raw output roughly halved; Codex and Grok reductions ~43–53%), cross-tool confirmation and root-cause de-duplication in the traditional lane, and an at-source grounding rule for Fable. This removed the bulk of raw noise, but what survived are *candidates*, not confirmed findings. The per-detector mechanisms, the raw-to-deduped reduction chart, and the pre-validation estimate table are collected in Appendix C.

**Layer 2 — human developer validation.** A product owner then reviewed a 262-finding validation list and marked each finding Valid or False Positive. These are the developer-validated rates charted below. The result is the point of the two-layer design: automated reduction was necessary but not sufficient. After Layer 1, **29% of the reviewed findings (77 of 262) were still false positives**, rising to roughly half for the general-purpose models (Codex 47%, Qwen 49%) and 64% for validated SonarQube, while the most tightly grounded lanes held (Trivy 0%, Fable 9%). Automated filtering narrows the pile; only human validation establishes what is real, which is why the workflow in Section 10 pairs layered discovery with an independent verification step before remediation.

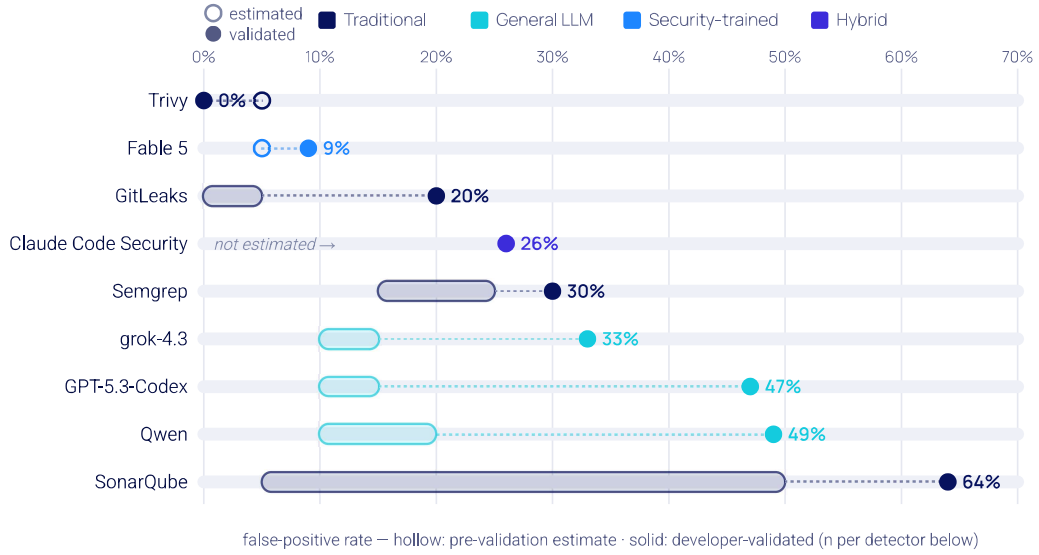
#### FALSE-POSITIVE RATES · PRE-VALIDATION ESTIMATE VS DEVELOPER-VALIDATED

This is Layer 2 — the human pass. Each rate began as a researcher estimate from raw scan output (Layer 1); a product-owner review then validated 262 of the findings — a *separate* pass from the 334-record discovery master, not the same rows — letting each estimate be checked against a developer verdict for the first time. It is

product-owner review, not the blind two-reviewer adjudication Appendix B specifies, and it measures false positives only: recall is still unmeasured.

### Estimated vs developer-validated false-positive rate

For each detector, the hollow marker is the pre-validation researcher estimate; the solid dot is the developer-validated rate. For detectors with an estimate, every validated rate except Trivy met or exceeded the estimate (Claude Code Security had no pre-validation estimate) — and the general-purpose LLMs ran two-to-five times higher than estimated.




Validated on the 262-record review pass, separate from the 334-record master. Product-owner review, not blind two-reviewer adjudication; volumes are unequal and small for some detectors (n: Trivy 19, Fable 53, GitLeaks 10, Claude Code Security 43, Semgrep 23, Grok 33, Codex 34, Qwen 33, SonarQube 14). SonarQube estimate spans its Vulns (5–10%) and Hotspots (30–50%) rows. Recall is still not measured.

#### DEFINITIONS USED THROUGHOUT

**Consolidated master record** — one row of the 334-row canonical master. The 334 rows encode **349 detector contributions** (14 rows carry more than one detector: 13 two-detector, 1 three-detector), so "334 records" is not 334 separate emitted findings; the detector-output sheets hold a larger pre-master row count and are not the denominator for the main figures. **Candidate** — a finding before validation. **Retained record** — a candidate kept after the workbook's de-duplication/sampling. **Strict match** — two records linked only when they share location and consequence. **Group** — records sharing repository + CWE + raw category (a heuristic that can merge unrelated bugs). **Paradigm** — the four organising classes above; an attribute of the run, not a proven cause of coverage.

WHAT CONSTRAINS EVERY NUMBER BELOW

**Unequal sampling.** The three general-purpose models were each truncated to **50** retained rows; Fable (79) and Claude Code Security (57) were not. Cross-detector and cross-paradigm *volumes* are therefore not comparable, and no detector ranking is drawn. The caps can also affect observed overlap and category composition, so the low overlap is a property of the sampled retained set, not an estimate of detector-level true overlap. **Partial adjudication.** A separate 262-finding product-owner review now yields a validated false-positive rate (charted above); recall is still unmeasured, so coverage counts here remain contribution and overlap, not recall. **Single run.** One run per stochastic configuration; treat every figure as a single-run snapshot scoped to *modern Python/FastAPI AI applications*. **Data version.** The workbook also contains an older 368-record generation; only the 334-row master is used here. **Availability.** Claude Code Security is a research preview. Access to Fable / Mythos-class models was suspended on 12 Jun 2026 under a US export-control directive; the Department of Commerce lifted those controls on 30 Jun 2026, with Fable 5 returning globally from 1 Jul 2026 and Mythos 5 restored to approved US organisations. The Fable scan here predates the suspension, so its results stand; neither model is prescribed as an available production stage, and availability should be re-checked at publication time. A full run manifest (model slugs, prompts, commits, tool versions) is still required — see Appendix B.

03 · THE SHAPE OF THE GAP

## An interpretive map of where findings landed

The map below organises the discussion: each category is placed by where, interpretively, it was caught.

## Figure 1 · Coverage map (conceptual synthesis)

Bubbles are vulnerability categories, sized by retained findings and coloured by the paradigm that surfaced the most of each.


The observed category distribution suggests two tendencies: dependency and known-pattern records cluster toward the catalogued / pattern corner, where the database-backed lane operated; authorization, business-logic, and architectural records toward the contextual / reasoning corner, where the model-based reviews operated. The sparse corners are a property of this configuration, not a proof about tool classes.

The data behind the placement — which paradigm surfaced which category, by retained count:

### Figure 2 · Retained records by selected major category × paradigm

Counts of retained records for selected major categories, per paradigm (not a full category matrix). A dot means that paradigm surfaced none of that category in this configuration. Counts are sampling-affected and are not a performance measure.

| | Traditional | General LLM | Security-trained | Hybrid |
|---|---|---|---|---|
| Dependency / CVE | 15 | 1 | 4 | · |
| Hardcoded creds | 17 | 26 | 5 | 4 |
| SQL injection | 4 | 23 | 3 | 2 |
| Command injection | · | 11 | · | · |
| Authentication bypass | 1 | 11 | 5 | 1 |
| Information disclosure | 3 | 15 | 15 | 1 |
| Security misconfig | 3 | 7 | 17 | · |
| Cryptographic issues | 1 | 2 | 6 | 1 |
| Path traversal | · | 5 | 3 | 2 |
| Access control / IDOR | · | 5 | 2 | 12 |

The dots mark categories a paradigm did not surface here. No column is dense across every row — consistent with the low-overlap result, and with the value of layering. Because the general-purpose models were capped at 50 rows, their cells understate raw output.

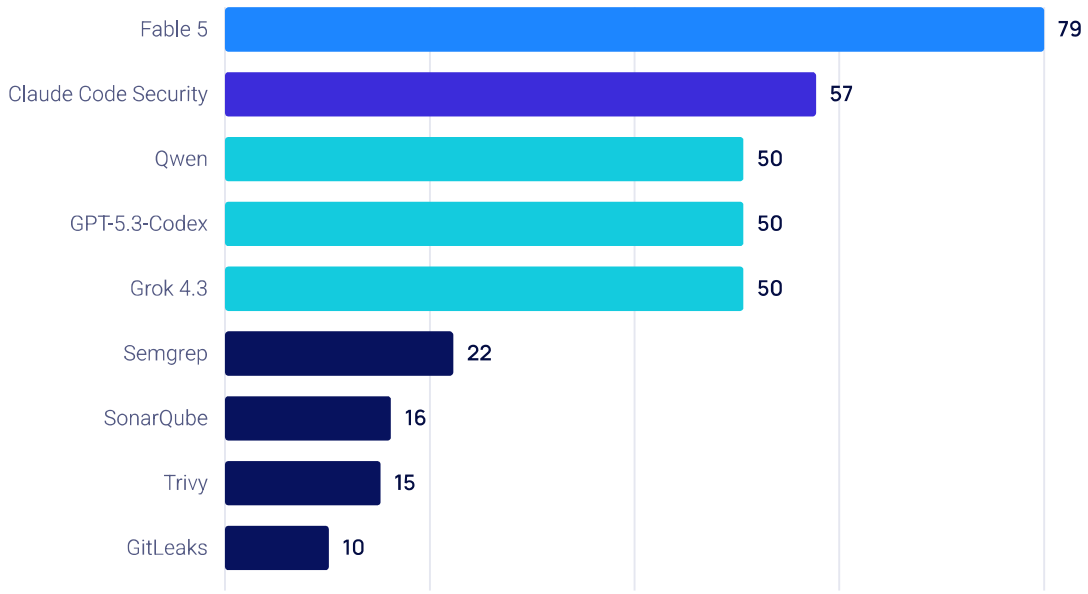
04 · OBSERVED GAPS IN THIS CONFIGURATION

## Where each approach surfaced little or nothing

First, an honest look at the retained-record counts, and at the sampling that shapes them.

### Figure 3 · Retained records per detector

Retained-record counts, **not** a performance ranking. The three general-purpose models sit at an identical 50 because each was truncated to a 50-row cap — those three bars mark the sampling limit, not an observed detector total — so cross-detector totals are not comparable.




Fable (79) and Claude Code Security (57) were not truncated; Qwen, Codex, and Grok were each capped at 50. The chart therefore shows the sampling rule as much as detector behaviour — which is precisely why no "leading detector" claim is made.

## Gap one: the configured baseline surfaced few logic flaws

Across the three codebases, the configured traditional runs surfaced few authorization / IDOR and business-logic records, while the model-based reviews surfaced many such candidate records. Because the model candidates are unadjudicated and the samples unequal, this is a difference in *retained output*, not a measured recall gap. Independent literature (Section 9) reports low real-world recall for some SAST tool sets, but those results are language- and benchmark-specific and are not a baseline for this study.

## Gap two: CVE provenance

The database-backed run (Trivy) accounted for the dependency-CVE records in the master inventory (13 IDs); one model run (Fable 5) also emitted two CVE IDs. A separate CVE matrix lists **19** IDs versus **15** in the master (an unreconciled discrepancy), and at least one ID is joined to the wrong product.

WITHHELD · CVE-COVERAGE FIGURE

We have withheld this figure rather than publish a number we cannot yet stand behind: a deliberate choice, not a gap. It awaits a full trace of every ID to its source detector output, package or product, advisory source

(NVD / OSV / GHSA), repository commit, and affected-version evidence, and resolution of the 15-vs-19 and Trivy/Fable attribution conflicts. See Appendix B.

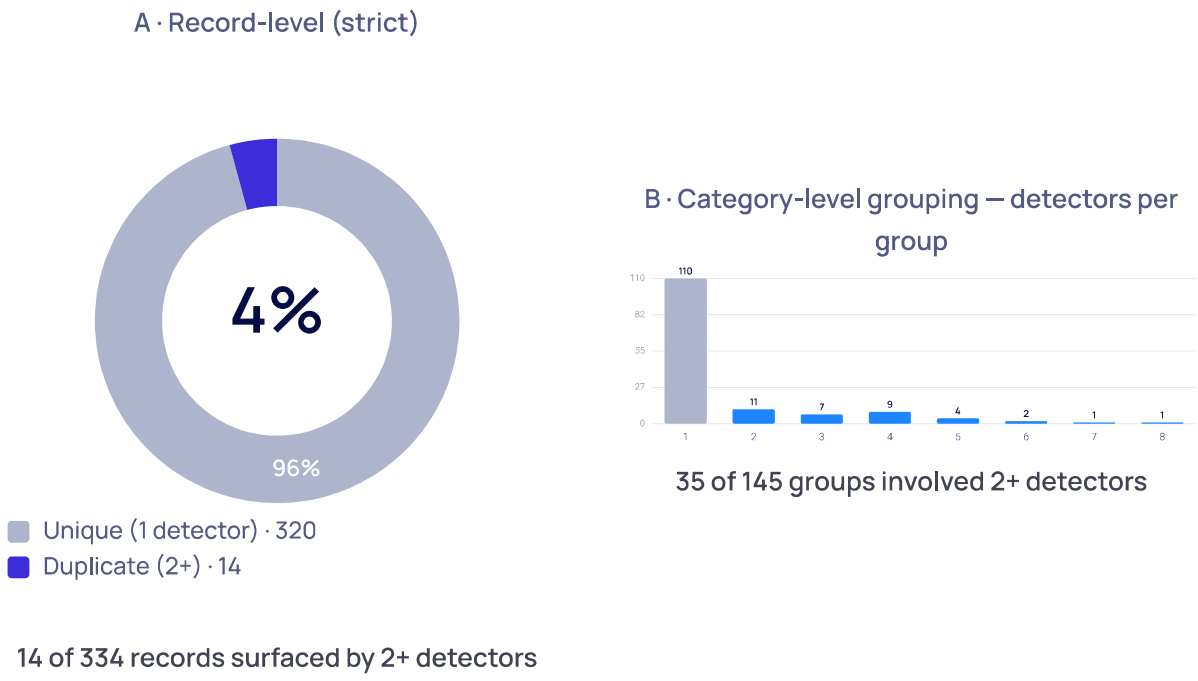
05 · HOW MUCH THEY AGREE

# The corroboration-granularity problem

How often did more than one detector surface the same thing? The data answers two different ways, and the choice matters. (Two units recur below: a *record* is one row of the 334-row master; a *group* merges records that share repository + CWE + category; 145 groups in all.)

### Figure 4 · The same question, two answers

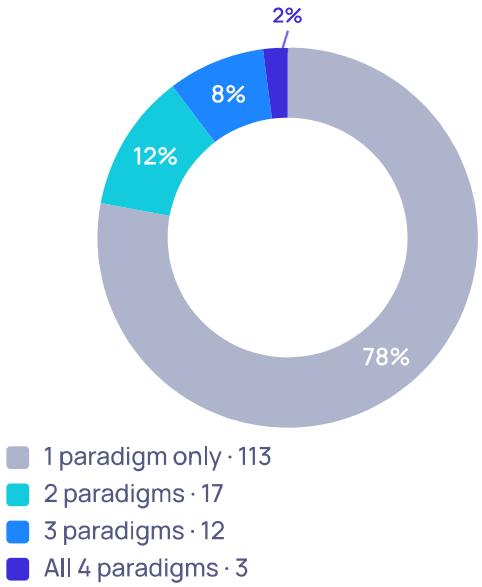
Left: strict record matching. Right: detectors appearing once records are grouped by repository + CWE + category.




**Precisely stated:** under strict record matching, only 14 of the 334 consolidated master records carried multi-detector support; under the broader repository–CWE–category grouping, 35 of 145 groups involved two or more detectors. Both indicate *low observed overlap*. Of those 14 records, only 6 cross paradigm lines; the other 8 are agreements between detectors of the *same* paradigm (for instance two general-purpose models), which is why Section 7's exclusive counts leave just 6 records shared across paradigms. Granularity B is a **cross-detector agreement cue**, not proof of truth: the grouping can merge unrelated bugs; for example, two distinct hardcoded-secret records in the same repository share repository + CWE + category and are counted as one group though they are different findings. The 145 groups exclude three aggregate *N/A / Various* groups; that exclusion is stated here explicitly.

### Figure 5 · How many paradigms surfaced each group

Of 145 groups, the large majority were surfaced by a single paradigm.




113 groups by one paradigm, 17 by two, 12 by three, 3 by all four. Note the two denominators that recur in this section: 35 of 145 groups were surfaced by two or more *detectors* (Figure 4B), but only 32 by two or more *paradigms* — three multi-detector groups drew their detectors from a single paradigm, which is why 110 single-detector groups become 113 single-paradigm groups. Read as exploratory overlap, not measured proof, but the direction is the paper's most robust observation.

06 · ANATOMY OF THREE FINDINGS

## What contextual reasoning surfaced here

Three candidate records the configured pattern scanners did not surface. Each is the kind of issue that is a property of behaviour across requests, state, or history rather than a single dangerous line. **All three are candidate records — not independently adjudicated or reproduced in this study.**

### 1 · Anonymous account takeover via a shared token cache

`CWE-287 · App A · surfaced by Hybrid · candidate`

THE REPORTED ISSUE

A public token-exchange endpoint accepted a "silent login" code and returned a session derived from a process-global identity-provider token cache, such that an unauthenticated

WHY THE CONFIGURED BASELINE DID NOT SURFACE IT

No dangerous call or tainted-input pattern in any single file; the issue is how identity flows through shared state across requests, outside the configured pattern rules, and dependent on whole-flow reasoning.

caller could receive a session for whichever user was last cached.

**If valid, blast radius:** takeover of any cached account, administrators included. **Status:** candidate; requires reproduction. **Suggested fix:** remove the silent-login path; do not derive identity from a shared global cache.

## 2 · One-time passcode reported as reusable

`CWE-287 · App C · surfaced by Hybrid · candidate`

### THE REPORTED ISSUE

The OTP was validated but reportedly not invalidated after a successful use, so the same code could be replayed to authenticate again.

### WHY THE CONFIGURED BASELINE DID NOT SURFACE IT

The code present is correct; the reported flaw is a missing single-use state transition. Pattern rules check what is present, not what is absent.

**If valid, blast radius:** a captured passcode reusable until expiry. **Status:** candidate; requires reproduction. **Suggested fix:** invalidate the OTP atomically on first use.

## 3 · JWT signing key reported recoverable from git history

`CWE-798 · surfaced by Hybrid · candidate`

### THE REPORTED ISSUE

A JWT signing key had reportedly been committed earlier and removed later, but remained recoverable from the repository's git history.

### WHY THIS RUN DID NOT SURFACE IT ELSEWHERE

Secret scanners such as Gitleaks *can* scan commit history; a full-history run is explicitly designed to do so. This run not surfacing the historical key likely reflects clone depth, scan mode, or rule configuration, not an inherent tool-class limitation. The exact scan command and history depth must be disclosed before attributing the miss.

**If valid, blast radius:** anyone who clones could forge tokens. **Status:** candidate; requires reproduction. **Suggested fix:** rotate the key, rewrite history, move to a secrets manager.

# What each paradigm surfaced without cross-paradigm support

Stripping out the records that crossed paradigm lines (the 6 cross-paradigm agreements) leaves what each paradigm surfaced without corroboration from another paradigm: 328 records in all, the 334-record master minus those 6, split 55 / 139 / 77 / 57 across the four paradigms. These are **paradigm-exclusive counts, not a performance ranking** — records no other paradigm surfaced, same-paradigm agreements included, not detector-level exclusives. They are sampling-affected: the general-purpose total is built from three models each capped at 50 rows. The value is in the *kinds* of records each paradigm surfaced, not in comparing the totals.

### Traditional

**55**

records, paradigm-exclusive

- Named dependency advisories — code-execution CVEs in deepdiff, nltk, unstructured, plus a DSA key-recovery flaw in jsrsasign (CVE-2026-4599, an incomplete-comparison bug, not RCE)
- HTTP request smuggling in h11
- Prototype-pollution-class dependency issues
- JWT used without signature verification (pattern)

Mostly named dependency CVEs — the database-backed lane's distinctive contribution in this run.

### General-purpose LLM

**139**

records (3 models, each capped at 50)

- Inconsistent access control across API routes
- Auth bypass by trusting client-side state
- Command injection via kubectl init container & ffmpeg subprocess
- Arbitrary code execution via a malformed PDF
- WebAssembly module loaded without integrity verification

Widest spread across injection, auth-logic, and novel execution paths — volume inflated by three models; unadjudicated.

### Security-trained (Fable)

**77**

records, single model

- Hardcoded AES key in the browser bundle
- Employee health PDFs (PII/PHI) reported committed to the repo
- Crypto-config endpoint reported exposing PBKDF2 inputs
- Live service keys reported in code comments
- System prompt instructing the model to "never decline" over confidential data

### Hybrid (research preview)

**57**

records, paradigm-exclusive

- Account takeover via shared-cache silent login
- Session revocation reported not enforced on protected routes
- Missing ownership check on a usage endpoint (IDOR)
- One-time passcode reported replayable

Concentrated on cryptography, secret exposure, privacy, and AI-specific logic. One model; not a class effect.

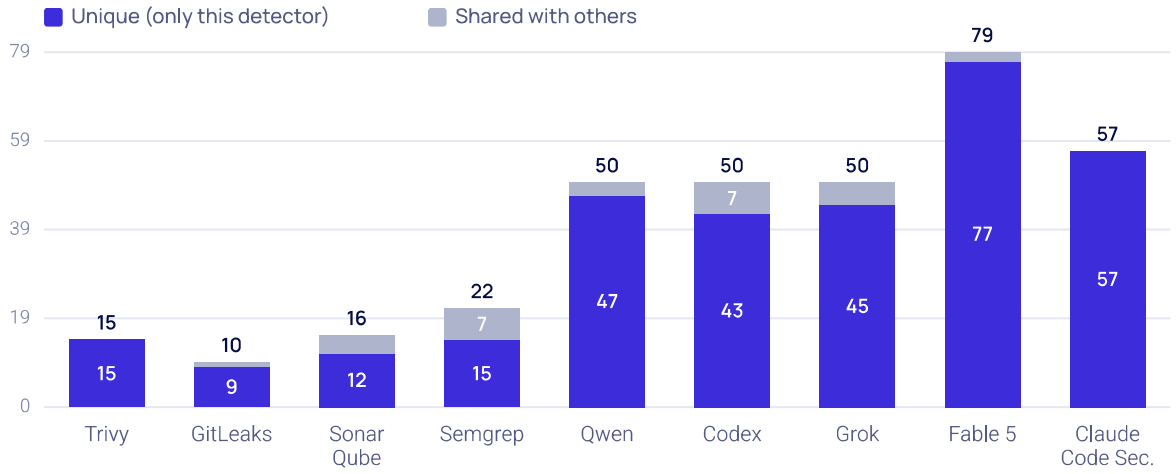
JWT signing key reported recoverable from history

WITHHELD · AI-VS-TRADITIONAL HEADLINE PERCENTAGE

It would be tempting to headline that ~82% of records were surfaced by an AI configuration and no traditional tool. The figure reconciles to the 334-row table, but because the underlying sampling is unequal (which affects de-duplication and category composition, not only volume), it is withheld until the model outputs are normalised under one rule. See Appendix B.

## Figure 6 · Records matched vs unmatched, per detector

"Unmatched" = not linked to another detector under the strict rule. Not a measure of correctness or novelty.




Most of each detector's retained records were unmatched under the strict rule. This reflects the low-overlap result and the strictness of the matching rule, not validated uniqueness.

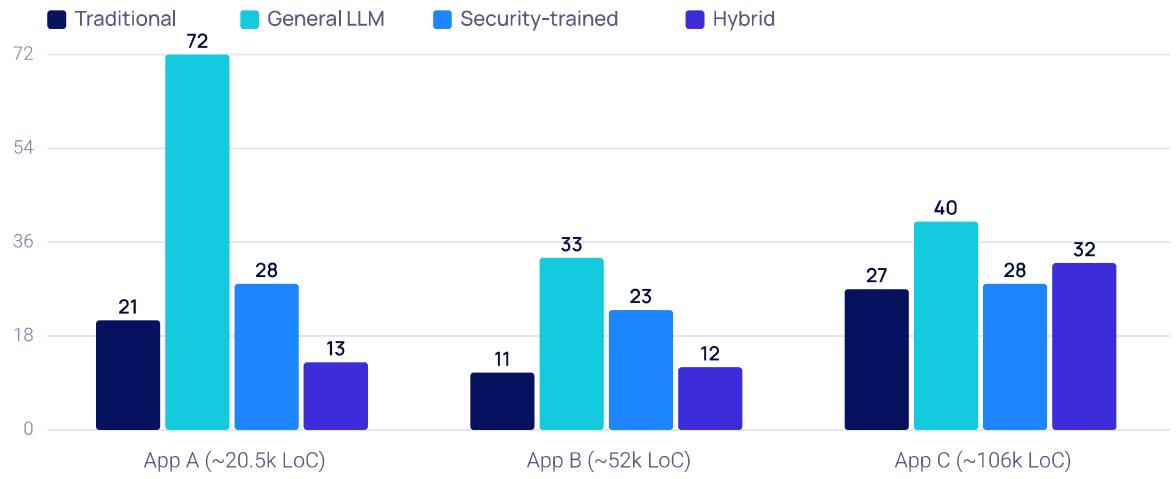
A NOTE ON THE SECURITY-SPECIALISED MODEL

In this run the security-specialised model concentrated on cryptography, secret-exposure, privacy, and AI-specific records, including a "never decline" system-prompt issue that the other sampled model outputs did not surface in this run. This is suggestive of a distinct contribution, but a *single* model cannot establish a paradigm-level effect; a second security-specialised model and blind adjudication are required before any class claim.

# Category mix varied; this design cannot isolate size

### Figure 7 · Retained records per repository, by paradigm

Counts vary by repository, but with one codebase per size band and capped model samples, this design cannot isolate a size effect.




General-purpose counts (72 / 33 / 40 across App A / B / C) do not increase monotonically with size, and the model samples are capped, so no "AI scales with codebase size" claim is drawn. The general-purpose bars sum to 145, not the 3 × 50 = 150 raw retained: where two general models flagged the same record, the master keeps a single general-paradigm row, collapsing a handful of same-paradigm agreements. Fable (79) and the hybrid (57) carry through unchanged, having one detector each; across all paradigms the bars total 340 — the 334 master plus the six cross-paradigm records counted in two paradigms. The defensible statement is that the category mix differed substantially by repository.

WITHHELD · SEVERITY DISTRIBUTION CHART

Better no chart than a misleading one. The severity chart is withheld because a defensible one cannot yet be drawn: the available severity data came from the older 368-record generation (its values summed to 366, not 334) and aggregated detector labels from different rubrics (for example SonarQube's Blocker / Major / Minor / Hotspot alongside Critical / High / Medium). Raw severity labels are not comparable across tools; severity will only be charted after a single normalised, adjudicated rubric is applied. See Appendix B.

More decision-useful than severity counts would be operational metrics this study did not capture: scan runtime and cost, analyst triage minutes per candidate, valid findings per 100 candidates, and time to verified fix. Those are listed as required follow-on work.

09 · EXTERNAL EVIDENCE & ITS LIMITS

## What independent work does and does not support

Four external strands get cited alongside studies like this. Each helps only when stated precisely, and none is a baseline for this Python/FastAPI run.

**SAST recall (peer-reviewed).** Li et al. (ESEC/FSE 2023, DOI 10.1145/3611643.3616262) found seven Java SAST tools detected ~12.7% of real-world vulnerabilities, most missed even when combined — language- and benchmark-specific context, not a baseline here.

**Vendor false-positive figure.** Ghost Security (2025) reported ~91% false positives scanning Go/Python/PHP for three vulnerability classes — vendor-authored and narrow; it does not license "traditional tools miss the logic layer."

**AI discovery needs validation.** Anthropic reported 500+ open-source vulnerabilities under triage; separately, the curl maintainer found five Mythos-labelled "confirmed" findings collapsed to one real bug after review — an argument for human validation, not a universal AI false-positive rate.

**Remediation telemetry.** GitHub states Copilot Autofix clears two-thirds of supported alerts, and (separately) that median fix time fell from 1.5 hours to 28 minutes — distinct populations, not one causal claim.

They do not converge on a slogan; the only safe cross-cutting reading is that finding is necessary but not sufficient; validation and remediation carry the operational weight.

10 · A CODE-CENTRIC DISCOVERY WORKFLOW

## Layered by capability, not by product

Because the approaches mostly surfaced different records, completeness argues for layering. The stages below are **capabilities**, not a product ranking, and this is a code-centric discovery-and-verification workflow, *not* a complete application-security programme. It omits threat modelling, DAST / API testing, fuzzing, IAST, runtime / cloud posture, penetration testing, and human review.

| STAGE | CAPABILITY | WHY IT EARNS A PLACE |
|---|---|---|
| 1 | SCA / SBOM, secrets, artifact provenance | Named CVEs, committed secrets, supply-chain integrity — fast, deterministic gates. |
| 2 | SAST / dataflow / IaC | Known dangerous patterns, taint where supported, infrastructure misconfiguration. |
| 3 | Contextual repository review (model-based) | Logic, authorization, and architectural *candidates* that pattern rules miss — requires validation. |
| 4 | Independent reproduction / verification | Converts candidates into validated findings; the operationally decisive layer. |
| 5 | Patch generation + regression tests | Turns validated findings into fixes that do not regress behaviour. |

| | | |
|---|---|---|
| 6 | DAST / API / fuzzing / runtime, where applicable | Behavioural coverage the code-centric stages cannot provide. |
| 7 | Production closure + feedback into rules / tests | Confirms risk reduction and feeds learning back into stages 1–2. |

Where a security-specialised model is used (stage 3), it should be read as "a security-specialised model, where authorized and available" — Fable / Mythos-class access was suspended on 12 Jun 2026 and, per Anthropic, restored from 30 Jun–1 Jul 2026 (Fable 5 globally; Mythos 5 to approved US organisations) — re-check availability at publication time; the Fable scan here predates the suspension. **If prioritising**, strengthen the verification capability (stage 4): converting candidates into validated, fixed, regression-tested risk reduction is the decisive step. This study does not provide the validated-risk, cost, and false-positive data needed to rank specific products, so no single-product recommendation is made.

### The trade behind each layer

Qualitative and directional only — not benchmarked here, and not a ranking.

| PARADIGM | TYPICAL RUNTIME | RELATIVE COST | OUTPUT VOLUME / NOISE | WHAT IT SURFACED HERE |
|---|---|---|---|---|
| **Traditional** | Seconds–minutes | Very low | Lower volume; CVE / pattern records | Dependency CVEs, secrets, known patterns |
| **General LLM** | Minutes | Moderate | High raw volume (capped here); unadjudicated | Broad logic & injection candidates |
| **Security-trained** | Minutes | Moderate–high | Moderate | Crypto / secret / privacy / AI-specific candidates |
| **Hybrid** | Minutes–longer | Higher (compute + tooling) | Moderate; integrated verification | Architecture & authorization candidates |

11 · THE HAND-OFF

## Discovery is the part this study measured

A note on what is, and is not, established here. **Measured in this experiment:** retained candidate records and their overlap. **Imported from the companion brief:** the finding-to-fix gap and the industry-

scale evidence below. **External / directional:** the timeline figure. **Author inference:** the workflow recommendation in Section 10.

This study measured discovery records, not validation time, patch capacity, or remediation throughput. It therefore *illustrates* the brief's finding-to-fix gap but does not test it: we did not measure whether the resulting candidates could be serviced at any particular cadence. The brief itself reports industry-scale signals: external evaluators confirming a frontier model completing a 32-step network attack, and 271 fixes driven into a single Firefox release while the great majority of candidate findings stayed unpatched.

**Figure 8 · Disclosure-to-exploitation timeline (imported)**

Reproduced from the companion brief; directional. Not measured in this experiment.


Imported from *Mythos, Measured* (IAPS / zerodayclock as summarised there). The April 2026 figure reflects reported timing, not an AI-attribution measurement in this study. Directional only.

WHERE THIS PAPER HANDS OFF

Add detection layers by capability, but first instrument **mean time from validated finding to verified-in-production fix**, and the validation throughput behind it. Adding AI discovery on top of an un-instrumented remediation pipeline widens the gap rather than closing it. The maturity model and 30 / 60 / 120-day plan live in the companion brief.

12 · OBJECTIONS WORTH ANSWERING

# The pushback, head-on

### Isn't this just model noise dressed up as findings?

Partly, which is why these are called candidate records and why adjudication is required before any precision claim. But the candidate set here is not raw model output: every detector ran a documented false-positive reduction step before a finding reached the master inventory: multi-pass deduplication and mandatory verbatim-code-evidence requirements for the LLMs, cross-tool validation and root-

cause deduplication for the traditional lane. The Codex raw-to-deduped ratio was roughly 2:1 (e.g. 247→122 on App A); Grok was similar. What remains are post-filter candidates, not hallucinations. The open question was whether survivors of those internal filters are true positives. A 262-finding product-owner review gives a first, scoped answer: it held best for the most tightly grounded lanes (deterministic Trivy at 0% and the security-trained Fable run at 9%) but confirmed the objection for the general-purpose models, where roughly half of the validated candidates were false positives (Codex 47%, Qwen 49%). Noise is not unique to models either: validated SonarQube was 64% on these codebases. The response is layering plus verification; verification is what removes that half, not picking a "clean" tool.

### Won't one frontier model eventually do all of this?

This run does not settle that. One model run did emit two CVE IDs, but the database-backed lane accounted for the CVE identifiers in the observed table. CVE / SBOM provenance is a distinct capability (database access, not just reasoning), and remains necessary regardless of model strength.

### Is "hybrid" just a model with a wrapper?

The design cannot isolate the cause. What is testable: integrated repository navigation, execution, history, and verification *may* change outcomes, but this study cannot attribute the hybrid's distinct records to any one component, and does not claim a false-positive advantage.

### Three codebases, one stack, one run — why believe it?

Treat it as exploratory and scoped: Python / FastAPI AI applications, single run, unequal sampling, incomplete adjudication. The one result that survives that scoping is the low observed overlap, and it points the same way as the broader case for layering.

13 · CONCLUSION

## Not one tool — a layer of tools, plus verification

The single result that survives every caveat in this study is the low observed overlap: across these three codebases, the four paradigms surfaced largely *different* candidate records, with only 14 of 334 carrying more than one detector. Read conservatively, that says no single detector, and no single paradigm, saw the whole picture on these three codebases. Coverage came from combining methods rather than from choosing one. On the discovery question, the defensible conclusion is not "pick the best scanner" but "run a layer of them," because each surfaced a different slice of the candidate set.

The equally important half is what the overlap result does *not* license. These are candidate records, not confirmed vulnerabilities, and where they were adjudicated the false-positive share was substantial: roughly half of the general-purpose model candidates, and 64% of validated SonarQube findings, did not hold up. More detectors therefore mean more to triage, not automatically more real bugs fixed. Layering broadens what you *find*; it does nothing on its own to confirm or remediate. The value only

materialises when each candidate passes through an independent verification step before it reaches a fix.

So the operational takeaway is a conjunction, not a single move: **layer for discovery, then verify before you remediate.** A conglomerate of complementary detectors widens coverage; independent verification converts that wider candidate set into fixed, regression-tested risk reduction without drowning the team in noise. This study measured only the first half, the discovery overlap, and argues the second half is where the operational weight sits. It does not rank the detectors, and it does not claim that any single one of them is enough on its own.

---

14 · LIMITATIONS

## What this paper is willing to be checked on

These are the open items a reviewer could still probe, distinct from the recomputation and source checks already completed in Appendix D. Appendix D records what was verified and held; the list below records what remains unresolved or untested.

1. **Unequal / capped sampling.** The three general-purpose models were each truncated to 50 retained rows; Fable and the hybrid were not. Detector and paradigm *volumes* are not comparable, and all ranking language is withdrawn.
2. **Mixed data generations.** The workbook contains an older 368-record generation alongside the 334-row master. Only the 334 master is used here; any figure that cannot be reconciled to it has been withdrawn (CVE coverage, severity).
3. **CVE table not reconciled.** 15 IDs in the master vs 19 in the matrix, conflicting Trivy / Fable attribution, and at least one wrong-product join. The CVE figure is withdrawn pending reconciliation.
4. **Adjudication is partial.** A 262-finding product-owner review provides validated false-positive rates (a separate pass from the 334 master, not blind two-reviewer adjudication; recall unmeasured). Detector-ranking and precision / recall language stays withdrawn pending blind adjudication; the false-positive rates are reported as scoped, non-blind results.
5. **Single run; single stack.** Non-deterministic models, one run each, three Python/FastAPI codebases. No generalisation to other languages or stacks is claimed.
6. **No run manifest yet.** Exact model slugs, prompts, commits, tool versions, and commands are required for reproduction (Appendix B).
7. **What would change the read.** A validated ground-truth set showing high inter-paradigm overlap, or a traditional toolchain matching model logic-finding recall under adjudication, would undercut

the layering thesis. Neither appeared here — but neither was decisively tested.

A · APPENDIX

## Method & reproducibility

Enough to interrogate the current numbers. The underlying records live in the companion workbook (334-row master).

| ELEMENT | HOW IT WAS DONE |
|---|---|
| **Canonical table** | The 334-row master inventory. An older 368-record generation in the workbook was not used. Data freeze: 18 Jun 2026. |
| **Common schema** | Each record normalised to: codebase, CWE, raw category, severity label, detector, location, description. |
| **Sampling** | General-purpose models truncated to 50 retained rows each; raw outputs were substantially larger (Codex: 247/275/327 raw → 122/146/188 deduped across App A/B/C; Grok: 107/101/129 raw → 53/47/62 deduped). Fable (79) and Claude Code Security (57) were not truncated. (Fable's source reports total 78 findings; the master carries 79 Fable rows because one combined dependency finding was split into two CVE-specific normalized records.) Unequal retained counts mean cross-detector volumes are not comparable — the single largest constraint on comparability. |
| **Strict matching** | Records linked only when they share location and consequence. Yields 320 single / 13 two-detector / 1 three-detector. |
| **Category grouping** | Repository + CWE + raw category, excluding three aggregate *N/A / Various* groups, giving 145 groups. A heuristic that can merge unrelated bugs. |
| **CVE handling** | Withheld. Master holds 15 CVE IDs (13 Trivy, 2 Fable); the matrix lists 19. Unreconciled — see Appendix B. |
| **Severity** | Withheld. Raw detector labels mix rubrics and are not aggregated pending normalisation. |

| | |
|---|---|
| **Adjudication** | Partial. No blind precision / recall figures are reported. Each detector ran a documented FP-reduction step before findings reached the master inventory (see Method note); researcher-estimated FP rates are tabulated there. A separate 262-finding product-owner review then validated those candidates, yielding the developer-validated false-positive rates in Section 2 — product-owner, not blind, covering false positives only (recall unmeasured), and a distinct pass from the 334-record master. |
| **Signature grid** | A unit chart, not a scatter plot: each dot is one of the 334 consolidated master records, laid out in a grid only to show proportion. A dot's position carries no meaning, so there are deliberately no x/y axes. Colour is the only encoded variable — single-detector, same-paradigm agreement, or cross-paradigm agreement. |

B · APPENDIX

## Unresolved data & required work

What must be completed before the withdrawn figures and claims can be restored, and before this becomes a benchmark rather than an exploratory snapshot.

| WORK ITEM | WHAT IT REQUIRES · WHAT IT UNLOCKS |
|---|---|
| **CVE reconciliation** | Trace all 19 matrix IDs and 15 master IDs to source detector output, package / product, advisory (NVD / OSV / GHSA), repository commit, and installed / fixed versions; resolve the 15-vs-19 count, the Trivy / Fable attribution conflict, and at least one wrong-product join. *Restores the former CVE-coverage figure.* |
| **Sampling normalisation** | Recover full raw model outputs (Qwen, Codex, Grok, Fable) and apply one complete, reproducible de-duplication / adjudication rule to every detector. *Unlocks cross-detector volumes and a cross-detector contribution figure after sampling normalisation.* |
| **Run manifest** | Per detector / run: repository commit SHA and history depth; tool version / edition / ruleset / command / image digest; model slug / host / quantization / temperature / context window; system and user prompts; tools, MCP, and network access; iteration and cost budget; repeated-run count and aggregation rule. *Unlocks reproducibility and fair comparison.* |
| **Blind adjudication** | A 262-finding product-owner review has been completed (developer-validated false-positive rates, Section 2) — but it is single-reviewer and non-blind. Still required: two reviewers independently label each candidate (valid / invalid / duplicate / needs-runtime / out-of-scope) with detector identity hidden; third-reviewer resolution; inter-rater agreement. With seeded or known-vulnerability |

| | |
|---|---|
| | benchmarks, recall becomes measurable. *Unlocks blind precision, recall, and detector-ranking claims.* |
| **Severity normalisation** | One rubric after adjudication (exploitability, reachability, impact, exposure, CVSS where applicable). *Restores the former severity figure.* |
| **Operational metrics** | Scan runtime / cost, triage minutes per candidate, valid findings per 100 candidates, time to verified production fix, cost per remediated high-risk finding. *Shifts the study from "who emitted more?" to "which workflow reduces validated risk most efficiently?"* |

**Run manifest — required fields and current status.** The fields a full reproduction needs, and what this snapshot already documents versus what is still outstanding.

| FIELD GROUP | WHAT IT MUST RECORD | STATUS |
|---|---|---|
| **Repository** | Commit SHA, branch, clone / history depth | Pending |
| **Tool scanner** | Version, edition, ruleset / DB snapshot, exact command, container image digest | Pending |
| **Model run** | Exact served slug, provider, date / time, context window, temperature / reasoning effort, quantisation, host | Model families named (GPT-5.3-Codex, grok-4.3, Fable 5, Qwen3-Coder-30B); exact slugs and run parameters pending |
| **Prompting** | System prompt, user prompt, context-selection rule | Pending |
| **Tools & access** | File, shell, and network access; MCP / tool list | Described by paradigm (Section 2); exact per-run config pending |
| **Output rule** | Retention cap, de-duplication rule, confidence labels, exclusion criteria | Documented (Section 2, Appendix C) |
| **Cost / runtime** | Tokens, wall-clock time, approximate cost, repeated-run count | Pending |

**Withdrawn pending the above:** the former CVE-coverage figure, the former severity figure, all detector-ranking statements, the AI-vs-traditional contribution headline, and precision / recall claims. Developer-validated false-positive rates are now reported (Section 2) as scoped, non-blind results from the 262-finding pass, distinct from the 334-record discovery master.

C · APPENDIX

# False-positive reduction, by detector

The per-detector mechanisms behind the false-positive-reduction step summarised in Section 2, the raw-to-deduped reduction chart, and the pre-validation estimate table. These record the research team's read of the elimination step applied before findings reached the master inventory; the developer-validated rates charted in Section 2 are the measured payoff.

HOW EACH MODEL REDUCED FALSE POSITIVES BEFORE FINDINGS WERE COUNTED

**Qwen3-Coder-30B** treated each candidate as a *hypothesis*: the workflow gathered supporting evidence from related files, functions, trust boundaries, and attack paths, re-evaluated the finding in that wider context, recomputed a confidence level, and classified it as **Confirmed, Likely, Needs Verification, or False Positive**. Cross-tool validation was also applied: a finding was treated as confirmed only when a second tool independently flagged the same file and line. Researcher-estimated FP rate: **10–20%** of retained candidates (over-flagging on logging patterns and unconfirmed auth-logic chains).

**GPT-5.3-Codex** ran four sequential passes — *per_file* (each file independently), *deep* (cross-file call-chain tracing), *manifest* (config and dependency files), and *holistic* (full cross-cutting attack chains) — with automatic deduplication collapsing any root cause flagged by more than one pass. Every surviving finding was required to include a verbatim code quote and a numbered attack scenario before it was written to output; findings without traceable code evidence were dropped. Raw-to-deduped reduction: 247→122 (App A), 275→146 (App B), 327→188 (App C), approximately 43–51%. Researcher-estimated FP rate: **10–15%**.

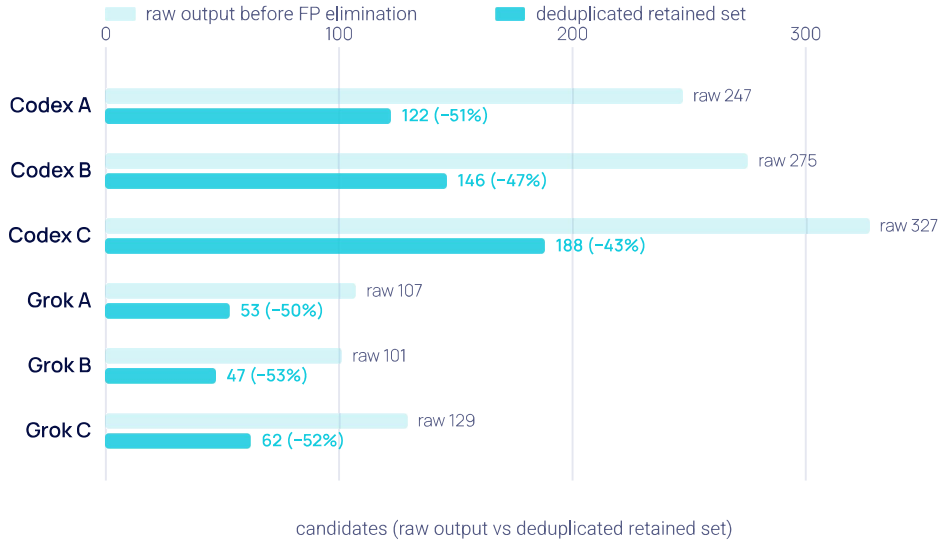
**grok-4.3** ran two passes (*per_file* then *deep*) with the deeper-context version replacing the narrower per-file version where both flagged the same root cause. Same mandatory code-evidence requirement as Codex. Raw-to-deduped reduction: 107→53 (App A), 101→47 (App B), 129→62 (App C), approximately 50–53%. Researcher-estimated FP rate: **10–15%**.

**Fable 5** did not preserve a separate raw-output set in this study; its source reports were already filtered by a grounding rule that required a verbatim code quote pulled directly from the file and a step-by-step numbered attack scenario before a finding was written, so anything that could not be grounded in actual code was not recorded. The priority-first deep file read (fewer files, read in full) appears to have further reduced context-free flagging. That filtering appears to have reduced speculative output, but it did not eliminate false positives: the researcher prior was **~5%**, the lowest estimate of any model-based detector, near the deterministic traditional scanners (GitLeaks 0–5%, Trivy ~5%); the later product-owner validation observed **9%**. Treat grounding as a useful discipline, not as proof of correctness.

All rates above are researcher estimates from direct observation of scan output for these codebases, not independently adjudicated. They document that raw output was substantially filtered before any finding reached the 334-row master inventory, but they do not license a measured precision or recall claim.

## Codex and Grok raw-to-deduped reduction

Raw output before FP elimination vs. retained findings per model per codebase. Dashed bar = raw; solid bar = retained. Reduction ratios (~43–53%) explain why a sampling cap was applied. Qwen's raw output is not charted here.




△ Amber rows indicate detectors with estimated FP rates above 15%. SonarQube appears as two rows — Vulns and Hotspots, one product in two reporting views — so the ten rows here still represent the nine detector configurations. Detail by mechanism in the table below.

The table below is the *pre-validation* estimate and the mechanism behind it — the research team's read of the elimination step applied before findings reached the master inventory, not a precision measurement. The developer-validated rates are the ones charted above.

| DETECTOR | PRIMARY FP-REDUCTION MECHANISM | EST. FP RATE |
|---|---|---|
| Traditional GitLeaks | Literal pattern match against known secret formats; each flagged item physically confirmed | **0–5%** |
| Traditional Trivy | Advisory-to-version match from vulnerability intelligence (NVD / GitHub Advisory); residual FP risk comes from package resolution, version-range interpretation, deployment reachability, and exploit-path applicability | **~5%** |
| Traditional SonarQube Vulns | Pattern-correct; structural inflation from identical code in multiple directory copies deduplicated (9→1) | **5–10%** |
| Traditional SonarQube Hotspots | Flags security-sensitive patterns requiring human decision; 64 hotspots, 0% reviewed at scan time | **30–50%** |
| Traditional Semgrep | Logger-credential rules fire on variable name proximity regardless of live value; 20 of 43 flagged as uncertain | **15–25%** |

| DETECTOR | PRIMARY FP-REDUCTION MECHANISM | EST. FP RATE |
|---|---|---|
| General LLM Qwen | Hypothesis → evidence → re-validate → classify; cross-tool validation for confirmation | 10–20% |
| General LLM GPT-5.3-Codex | 4-pass multi-scan with auto-dedup; mandatory verbatim code quote + attack scenario per finding | 10–15% |
| General LLM grok-4.3 | 2-pass (per_file + deep) with deeper context replacing narrower; mandatory code evidence | 10–15% |
| Security-trained Fable 5 | Grounding rule: no finding written without verbatim code quote + numbered attack scenario; no raw-vs-deduped gap | ~5% |
| Hybrid Claude Code Security | Repository navigation, execution, and verification tooling integrated into finding workflow | Not estimated |

D · APPENDIX

# Independent data-validation & source verification

This appendix records an independent pass over the paper against the raw data files used to build it and against every external figure it cites. Each computable number was recomputed from the primary workbook; each external claim was checked at its origin. Results are reported whether or not they are favourable.

INTERNAL RECOMPUTATION — CHARTED FIGURES VS THE 334-ROW MASTER

Every quantitative figure derived from the 334-row master inventory was recomputed independently and matched: record structure (334 records: 320 single-detector, 13 two-detector, 1 three-detector; 349 detector contributions); per-detector retained counts (Fable 79, Claude Code Security 57, Qwen / Codex / Grok 50 each, Semgrep 22, SonarQube 16, Trivy 15, GitLeaks 10); paradigm exclusives (Traditional 55, General 139, Security 77, Hybrid 57); the category×paradigm matrix, repository×paradigm table, group counts; and the Codex / Grok raw-to-deduped reductions. No figure derived from the master diverged from it. The App A/B/C footprint table is *not* fully re-derived from the master (it draws on a separate comparative-analysis source), and the App C Kubernetes-manifest count remains a source conflict, flagged separately below.

DEVELOPER-VALIDATED FP-RATE PROVENANCE

The validated false-positive rates in Section 2 trace to the updated 262-row validation list (185 valid / 77 false positive overall); all nine per-detector rates reproduce exactly from that file. An earlier adjudication of the same 262 findings (182 valid / 80 false positive) also exists and is *superseded*: it differs by exactly three findings (one each from Codex, Fable, and Qwen), later re-adjudicated from false positive to valid. The paper uses the later figures throughout. A reader cross-checking the earlier summary should therefore expect Codex 50%→47%, Qwen 52%→49%, and Fable 11%→9%; the discrepancy is a data-version difference, not a computation error.

External figures cited in Section 9 and the CVE labels used in Section 7 were checked against primary sources where available; where only a vendor-authored or secondary summary exists, that status is labelled and the figure is treated as contextual only. Each row below records the source actually consulted.

| SOURCE | CLAIM AS CITED | VERIFICATION |
|---|---|---|
| Li et al., ESEC/FSE 2023 | ~12.7% real-world detection; most missed even combined | Confirmed (DOI 10.1145/3611643.3616262) |
| GitHub Copilot Autofix | Two-thirds of supported alerts; median fix 1.5 h → 28 min | Confirmed (github.blog, 2024) |
| curl maintainer review | Five Mythos-labelled findings → one real low-severity bug | Confirmed (daniel.haxx.se, May 2026) |
| Ghost Security 2025 | ~91% false positives across Go/Python/PHP for selected vulnerability classes | Confirmed via secondary summary (Help Net Security, Jun 2025) of the vendor report; primary report not independently obtained — vendor-authored / contextual only |
| Export-control timeline | Directive 12 Jun; lifted 30 Jun; Fable 5 global 1 Jul 2026 | Confirmed (public reporting) |
| CVE-2026-4599 (jsrsasign) | DSA key-recovery / incomplete-comparison flaw, not RCE | Confirmed — the not-RCE classification matches NVD (DSA key-recovery / incomplete comparison) |
| CVE-2025-43859 (h11) | HTTP request smuggling | Confirmed, correctly attributed |

| SOURCE | CLAIM AS CITED | VERIFICATION |
|---|---|---|
| CVE-2024-22363 / CVE-2023-30533 (SheetJS) | ReDoS / prototype pollution in xlsx | Confirmed, correctly attributed |

RESIDUAL DISCREPANCY CARRIED FORWARD

One source conflict remains unreconciled: the App C footprint line reports 16 Kubernetes manifests (from the comparative-analysis source), while the Fable primary review of the same repository states it contains no Kubernetes, Helm, or Terraform manifests and deploys through an Azure Web App pipeline despite a repository label suggesting Kubernetes-based deployment. The two source documents disagree and the comparative-analysis source could not be independently re-derived here, so the figure is left as reported and flagged for reconciliation with the run-manifest work in Appendix B.

This validation covers arithmetic reproducibility and source attribution only. It is not a re-run of the detectors and is not a full reproducibility package: it does not re-execute any scan, does not measure detector recall, does not re-adjudicate any finding, and does not convert the researcher-estimated rates in Appendix C into a measured precision. A complete run manifest (Appendix B) is still required to reproduce the scans themselves.

WHAT'S NEXT

## Can AI break in?

This study measured what each approach *finds*. The next asks what an attacker could *do* with it. The question is not whether a model can exploit a single weakness in isolation; scanners already flag those one at a time. It is whether a model can take the defects surfaced here and **chain them — combining several across a codebase into a working path to access**. That is the distinctive advantage of an AI attacker: a human tester usually works one finding at a time, while a model can reason over many at once and stitch even low-severity issues into a real breach. Our next paper puts that to the test, driving models to combine the identified vulnerabilities and confirm which chains are genuinely reachable, under authorized and controlled conditions. It is the sharpest form of the verification layer this paper argues for.

## SOURCES & COMPANION MATERIAL

Primary data: the 334-row Vulnerability Research Workbook master inventory (data freeze 18 Jun 2026). Strategic framing: *Mythos, Measured — A CXO Brief on AI-Powered Cybersecurity* (2026).

External evidence (context only, not baselines for this study): Li et al., "Comparison and Evaluation on SAST Tools for Java," ESEC/FSE 2023, DOI 10.1145/3611643.3616262 (the 12.7% figure); Ghost Security 2025 (vendor false-positive figure); Anthropic, Claude Code Security; GitHub, Copilot Autofix telemetry (github.blog, 2024); D. Stenberg / curl, maintainer review of a Mythos-reported finding (daniel.haxx.se, May 2026). Methods differ; figures are not directly comparable to this study. External figures were checked against primary sources where available; where only vendor-authored or secondary summaries were used, that status is labelled and the figure is treated as contextual only; see Appendix D.

## REFERENCES


1. Kaixuan Li et al., "Comparison and Evaluation on Static Application Security Testing (SAST) Tools for Java." ESEC/FSE 2023. doi.org/10.1145/3611643.3616262. Accessed 5 Jul 2026.
2. Ghost Security, *Exorcising the SAST Demons* (2025), as summarised in Help Net Security, "91% noise: A look at what's wrong with traditional SAST tools," 19 Jun 2025. helpnetsecurity.com/2025/06/19/traditional-sast-tools. Vendor-authored; contextual only. Accessed 5 Jul 2026.
3. GitHub, "Found means fixed: Secure code more than three times faster with Copilot Autofix," github.blog, Aug 2024. github.blog — secure-code-more-than-three-times-faster-with-copilot-autofix. Accessed 5 Jul 2026.
4. Daniel Stenberg, "Mythos finds a curl vulnerability," daniel.haxx.se, 11 May 2026. daniel.haxx.se/blog/2026/05/11/mythos-finds-a-curl-vulnerability. Accessed 5 Jul 2026.
5. Anthropic, "Redeploying Claude Fable 5," 30 Jun 2026 — states that export controls on Fable 5 and Mythos 5 were lifted, with Fable 5 returning globally from 1 Jul 2026 and Mythos 5 availability limited to approved US organisations. anthropic.com/news/redeploying-fable-5. Accessed 5 Jul 2026.
6. Anthropic, "Making frontier cybersecurity capabilities available to defenders" (Claude Code Security research-preview announcement), Feb 2026. anthropic.com/news/claude-code-security. Accessed 5 Jul 2026.


## SECURITY & HANDLING NOTE

Repository identities are anonymised throughout: described by neutral labels (App A / B / C) and footprint, never by name or function. No secrets, credentials, or personal data are reproduced anywhere in this paper; the tables and charts report only aggregate, de-identified results. Any sensitive findings surfaced during the study were disclosed privately to the owning teams and handled under responsible-disclosure and data-protection practices, outside this public write-up.

AI & Cybersecurity research · exploratory snapshot: retained candidate records, scoped to the tested configurations, pending blind adjudication and a full reproducibility manifest.

HEXAWARE

## About Hexaware

Hexaware is a global technology and business process services company. Every day, Hexawarians wake up with a singular purpose: to create smiles through great people and technology. With this purpose gaining momentum, we are well on our way to realizing our vision of being the most loved digital transformation partner in the world. We also seek to protect the planet and build a better tomorrow for our customers, employees, partners, investors, and the communities in which we operate.

With offices across the world, we empower enterprises to realize digital transformation at scale and speed by partnering with them to build, transform, run, and optimize their technology and business processes.

Learn more about Hexaware at https://www.hexaware.com.

Get in touch