
 COMPARATIVE REPORT · BENCHMARK V3.1

The RealVuln Benchmark Comparative Report

Python and TypeScript/JavaScript stratum

Prepared for: security and risk leadership in regulated financial institutions

Companion briefing: Rings of Risk: Evaluating Defense-in-Depth in the Era of
Agentic AI Exploits

AUTHORS

John Pellew

john@kolega.ai

Faizan Raza

faizan@kolega.ai

ISSUED BY



Kolega Labs

DATA

RealVuln v3.1.0, generated 11 September 2026
140 repositories · 35 scanner runs
Ground-truth hash sha256:0754363...5d23585

ISSUED

23 October 2026
Public data

Executive summary

This report asks one question, and it is narrower than a product comparison: if an autonomous adversary can chain several small, individually unremarkable flaws into a working attack path, how many of those flaws would your current scanning have surfaced at all? Every flaw a scanner misses is a link an agent can use.

RealVuln answers that question with measurements on real application code rather than synthetic test cases. Version 3.1 covers 140 repositories, 66 in Python and 74 in TypeScript/JavaScript, holding 4,138 scored vulnerabilities and 280 false-positive traps. 35 configurations across three categories scanned them. All artifacts, ground truth and scoring code are public.

84.4 F3 (strict), security-specialized Kolega DevSec Max, 90.2% recall	79.5 Best general-purpose LLM GPT Daybreak Blue, 81.8% recall	10.0 Best rule-based SAST Semgrep, 9.9% recall	0/232 Broken-access-control findings detected by Semgrep
---	--	---	---

- The tiers are far apart.** On the like-for-like two-language stratum, the security-specialized scanner leads at F3 84.4, the best general-purpose LLM reaches 79.5 and the best rule-based tool reaches 10.0. The Python stratum alone gives the same order: 86.4 / 80.0 / 23.1.
- Rule-based tools go blind where chaining happens.** Semgrep detected 0 of 232 broken-access-control/IDOR findings and 0 of 173 missing-authentication findings. SonarQube also scored 0%. Rowan reached 6.0% and 3.7%. These are the families an agentic attacker composes into a path.
- The median scanner misses seven of every ten access-control flaws.** Across all 35 runs, median recall on broken access control is 29.7% and on missing authentication 38.2%. The distribution is wide, from 0% to 87.9%, so a vendor's category label tells you little.

- 4 **Scale breaks detection.** Provisional large-application results show aggregate recall falling from 80.55% on small Python applications to 1.77% on five systems between 263k and 816k lines, and 0.00% on three of them. On five large Java systems the same configuration matched 3 of 295 entries (1.02%), and 0 of 47 on the largest, at over 1M lines.
- 5 **Cost is not the constraint.** The strongest general-purpose LLMs complete the full 741,034-line corpus for \$74–\$140 per 100,000 lines, and a complete sweep costs \$548–\$1,041 in API spend. Spending more does not buy detection: the most expensive run in the benchmark scores 52.1.
- 6 **Validation is the only real fix.** Every coverage claim in an RFP or audit pack should be reproduced on repositories of your own size before you rely on it. If a tool cannot demonstrate recall on a 600,000-line system, it has not demonstrated coverage of your estate.

WHY THIS MATTERS NOW

The recent global PHP remote-code-execution disclosures made an old argument urgent. Autonomous agents do not inspect code; they compose flaws. A defense-in-depth architecture is only as strong as its weakest ring, and this report measures detection directly, at the layer where access control actually lives.

■ SECTION 1

Method, and what this benchmark does not claim

RealVuln runs each scanner against the same pinned commits of real applications and matches the findings to hand-labeled ground truth. A match requires agreement on three fields: **file path** (normalized, exact), **CWE** (present in the entry's accepted-CWE list) and **line number** (within ten lines of the labeled range). Each ground-truth entry can be matched once. Findings left over are false positives; vulnerabilities left over are false negatives. A scanner that reports an attack chain rather than a single location may declare alternative locations, and it gets credit for each distinct vulnerability the chain genuinely identifies.

TABLE 1 Corpus composition, RealVuln v3.1.0

STRATUM	REPOS	LINES OF CODE	SCORED VULNS	FP TRAPS
Python	66	133,782	1,902	—
TypeScript / JavaScript	74	607,252	2,236	—
Total	140	741,034	4,138	280

False-positive traps are counted at corpus level.

The corpus holds two kinds of target. Fifty repositories are teaching and CTF applications, written by humans and vulnerable on purpose. The other ninety are company-style applications generated by named coding models (Claude Opus 4.7, GPT-5.5, Kimi K2.6 and others) and then seeded with reviewed vulnerabilities. That second kind matters most to institutions now shipping AI-assisted software. The corpus covers eighteen CWE families.

Scoring

The primary metric is **F3 (strict)**, the harmonic mean of precision and recall with $\beta = 3$, scaled to 100. That weighting puts recall nine times above precision. The asymmetry is deliberate and fits a bank: a false positive costs an analyst a few minutes, while a missed vulnerability can cost a breach.

Strict aggregation counts any repository a scanner failed to finish as if it had reported nothing, so an unreliable tool cannot hide behind a partial run. F1 and F2 are published alongside F3, so anyone can re-test the conclusions under a different weighting.

DISCLOSURE: READ THIS BEFORE CITING THE NUMBERS

Kolega Labs wrote this benchmark, and one of the scanners it evaluates, Kolega DevSec Max, is our own product. That is the same conflict of interest we criticize in vendor-run benchmarks, and it should temper how you read our first-place result.

Disclosure is the only mitigation we can offer. **Every ground-truth label, every raw scanner output, the matching code, the scoring pipeline and the interactive dashboard are public**, so the result can be reproduced or falsified without asking us. We have also reported results where a competitor outperforms DevSec Max (Section 3), and results that are unflattering to the benchmark's own corpus (Section 5).

SECTION 2

Results on the like-for-like stratum

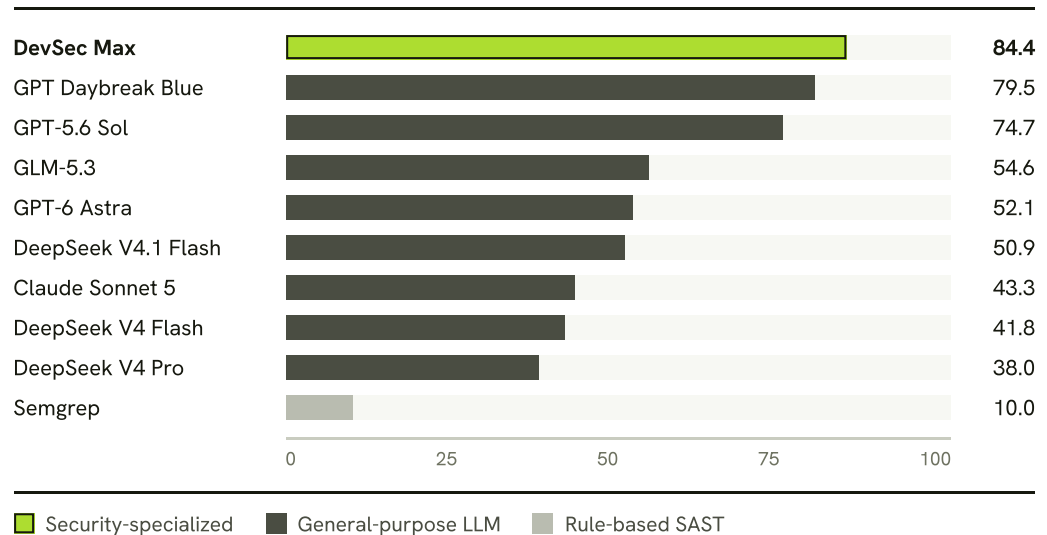
Thirty-five scanner runs are recorded in v3.1. Ten completed both language strata, and only those ten support a direct comparison. The rest ran on Python alone and appear on the Python leaderboard. Table 2 and Figure 1 show the comparable set.

TABLE 2 Comparable leaderboard, both strata (140 repositories), ranked by F3 strict

SCANNER	CATEGORY	F3	RECALL	PRECISION	MISSED	REPOS	\$/100K LOC
Kolega DevSec Max v0.1.0	SECURITY-SPECIALIZED	84.4	90.2%	53.2%	405	140/140	2.39
GPT Daybreak Blue	GENERAL LLM	79.5	81.8%	64.0%	755	140/140	73.90
GPT-5.6 Sol	GENERAL LLM	74.7	76.9%	59.3%	955	140/140	75.60
GLM-5.3	GENERAL LLM	54.6	55.9%	44.9%	1,823	136/140	14.80
GPT-6 Astra	GENERAL LLM	52.1	53.1%	44.7%	1,941	140/140	140.40
DeepSeek V4.1 Flash	GENERAL LLM	50.9	50.7%	52.3%	2,039	140/140	0.00
Claude Sonnet 5	GENERAL LLM	43.3	42.5%	52.6%	2,380	140/140	17.60
DeepSeek V4 Flash	GENERAL LLM	41.8	40.7%	56.3%	2,455	140/140	0.50
DeepSeek V4 Pro	GENERAL LLM	38.0	36.4%	60.9%	2,630	140/140	2.00
Semgrep	RULE-BASED SAST	10.0	9.9%	11.5%	3,728	140/140	—

Rule-based tools are licensed rather than metered per token, so their figures are not comparable on this axis. Cost is list-price API spend for the scored run, not total cost of ownership.

FIGURE 1 F3 (strict) across the 140-repository, two-language corpus



Recall is weighted nine times above precision. The gap between the top of the general-purpose tier and the rule-based tier is 69.5 points, and the gap between the two leaders is 4.9.

Reading the table

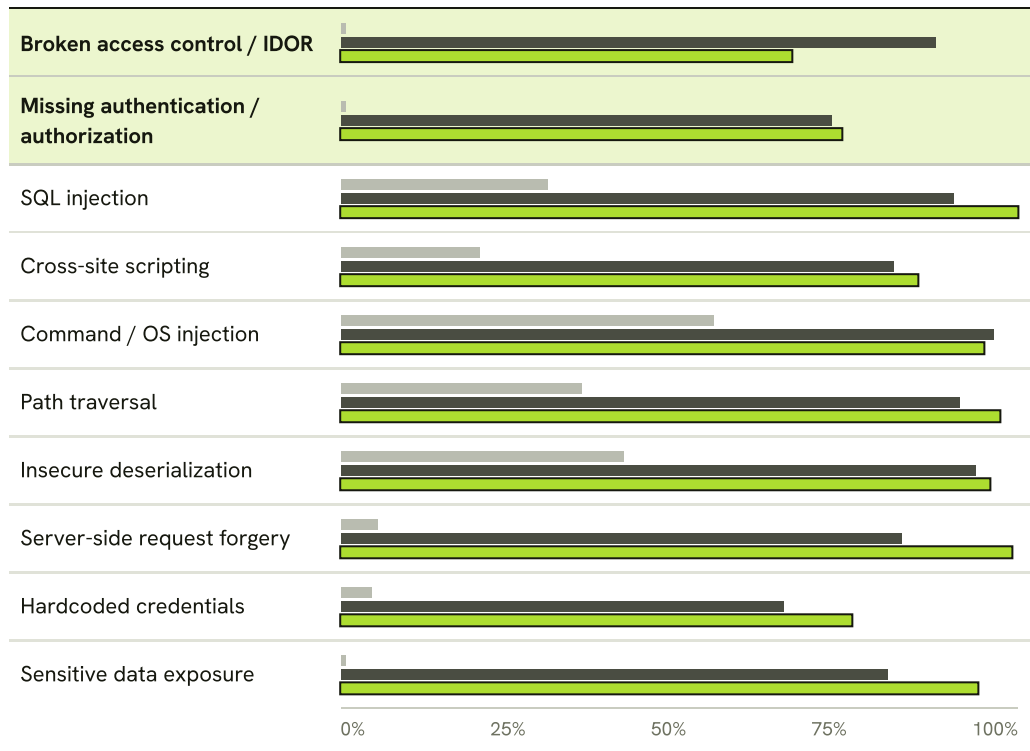
The ranking is less informative than the detail behind it. At the top of the general-purpose tier, Daybreak Blue's 81.8% recall and 64.0% precision would have counted as a good result for a specialist tool two years ago. The claim that large models cannot reason about real code no longer holds. Within that same tier, recall runs from 81.8% down to 36.4%, so saying "we use an AI scanner" describes capabilities that differ by a factor of two. Rule-based tools sit apart from that scale. Semgrep's 9.9% recall means it reported one vulnerability for every ten labeled ones in the corpus.

Reliability is a separate axis. Incomplete runs are common among frontier models. GLM-5.3 finished 136 of 140 repositories, and outside the comparable set several configurations finished only the Python stratum. Strict scoring counts an unfinished repository against the scanner, which is the right treatment for a regulated environment: a scan that intermittently fails to complete is not a control anyone can attest to.

SECTION 3

Where coverage disappears

Aggregate scores hide the shape of the failure. Figure 2 breaks recall out by vulnerability family for three archetypes: the best rule-based tool, the best general-purpose LLM and the security-specialized leader. The shortfall is not spread evenly. It concentrates on the families an attacker chains together.

FIGURE 2 Recall by vulnerability family, all 140 repositories

■ Semgrep
 ■ Daybreak Blue
 ■ DevSec Max
 ■ Families an adversary composes into a path

Semgrep detects none of either highlighted family. Full family counts and per-scanner values are in Table 3.

TABLE 3 Recall by family (n = labeled vulnerabilities), selected scanners, all repositories

FAMILY	N	SEMGREP	DAYBREAK	DEVSEC MAX
Broken access control / IDOR	232	0.0%	87.9%	66.4%
Missing authentication / authorization	173	0.0%	72.3%	74.0%
SQL injection	154	30.5%	90.3%	100.0%
Cross-site scripting	206	20.4%	81.6%	85.0%
Command / OS injection	136	55.1%	96.3%	94.9%
Path traversal	104	35.6%	91.3%	97.1%
Insecure deserialization	48	41.7%	93.8%	95.8%
Server-side request forgery	93	5.4%	82.8%	98.9%
Hardcoded credentials	216	4.6%	65.3%	75.5%
Sensitive data exposure	349	0.0%	80.8%	94.0%

Bold marks the leading value per family.

Rule-based detection is confined to syntactic pattern matches. It keeps partial recall on command injection (55.1%), where the pattern is a recognizable call, and falls to zero on authorization, where the flaw is a missing check spread across files. A missing check has no text to match. It is a property of the call path.

The families where rule-based coverage is zero are also the ones that make chaining cheap. An access-control gap needs no memory-corruption primitive to be useful. Put it next to an SSRF (5.4% rule-based recall) or a deserialization flaw and you have a path from the perimeter to a data layer without a single exotic exploit. That is the composition an autonomous agent performs routinely, and it is what the Rings of Risk argument is about.

No scanner dominates every family, ours included. Daybreak Blue beats DevSec Max on the flagship access-control family by 21.5 points (87.9% against 66.4%). DevSec Max leads on SSRF, sensitive data exposure, path traversal and SQL injection, and it leads on overall recall through breadth rather than through superiority in any one class. A reader looking for a single winner is looking for the wrong conclusion. The useful question is which failure modes your estate can absorb.

■ SECTION 4

The two language strata behave differently

TypeScript/JavaScript is the harder stratum, though not for the usual reason. The TS/JS corpus is 4.5 times the size of the Python corpus (607,252 lines against 133,782) but carries only 18% more vulnerabilities, 2,236 against 1,902. Density is therefore about a quarter as high. Precision falls for that reason alone, since a scanner's false positives land in far more clean code.

TABLE 4 Recall by stratum for scanners that completed both

SCANNER	PY F3	PY RECALL	TS/JS F3	TS/JS RECALL	Δ F3
Kolega DevSec Max	86.4	88.8%	82.7	91.5%	−3.7
GPT Daybreak Blue	80.0	80.5%	79.2	82.8%	−0.8
GPT-5.6 Sol	77.8	78.7%	72.1	75.4%	−5.7
GLM-5.3	63.3	63.8%	47.4	49.3%	−15.9
Claude Sonnet 5	52.8	51.5%	35.3	34.8%	−17.5
DeepSeek V4 Flash	51.7	49.9%	33.5	32.8%	−18.2
DeepSeek V4 Pro	50.2	48.3%	27.5	26.3%	−22.7
Semgrep	7.4	7.0%	12.2	12.3%	+4.8

Δ F3 is TS/JS minus Python, derived from the published values.

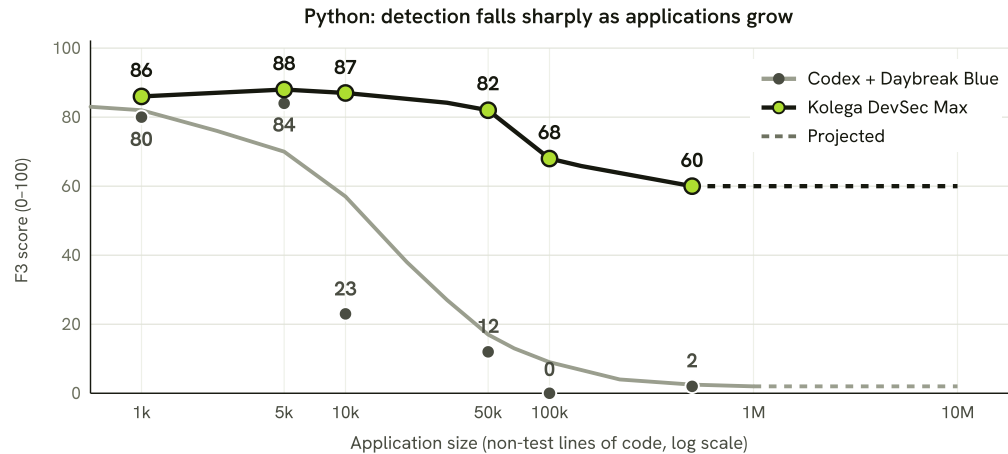
The leaders hold their position. DevSec Max moves from 86.4 to 82.7 (−3.7 points), Daybreak Blue from 80.0 to 79.2 (−0.8) and Sol from 77.8 to 72.1 (−5.7). The middle of the field falls by 16 to 23 points: GLM-5.3 by 15.9, Claude Sonnet 5 by 17.5, DeepSeek V4 Flash by 18.2 and DeepSeek V4 Pro by 22.7. For a technology-selection decision, that asymmetry is the most useful signal in the report. A scanner that is adequate on small Python applications is not adequate on a large TypeScript estate, and the degradation is much steeper in the middle of the market than at the top.

Semgrep is the exception. Its recall rises slightly on TS/JS, from 7.0% to 12.3%, though it still tops out at twelve percent.

SECTION 5

Scale and the repository-size gap

Every number so far comes from repositories that are small by enterprise standards. The Python applications cluster below 5,550 lines, and the corpus was designed to stay tractable for language models and for hand-labeling. A bank's core systems are not tractable. The provisional evidence says this is where general-purpose detection fails almost completely.

FIGURE 3 Provisional: F3 against application size, Python

Detection by a leading general-purpose configuration falls from 84 at 5,000 lines to 23 at 10,000, 12 at 50,000 and 0 at 100,000. The security-specialized configuration degrades far more slowly and is projected to hold near 60 at enterprise scale. Points are measured; the grey line is a fitted curve. **Provisional research result, outside the official benchmark and dashboard**; see the caveat below.

TABLE 5 Provisional large-application results (GPT Daybreak Blue via Codex), outside the scored benchmark

SYSTEM	WHAT IT DOES	LANGUAGE	SIZE (LOC)	GT ENTRIES	MATCHED	RECALL
Marble	Build-automation server	Java	100-250k	36	0	0.00%
Pewter	IoT device-management platform	Java	250-500k	80	0	0.00%
Basalt	Identity and access management server	Java	>500k	84	2	2.38%
Quartz	Business-process automation engine	Java	>500k	48	1	2.08%
Ridge	Enterprise content-management platform	Java	>1M	47	0	0.00%
Slate	Headless e-commerce platform	Python	250-500k	76	2	2.63%
Talus	Network infrastructure-management tool	Python	250-500k	144	8	5.56%
Harbor	Workflow-orchestration platform	Python	>500k	48	0	0.00%
Umbler	Business-management (ERP) suite	Python	>500k	175	0	0.00%
Vale	Application error-monitoring platform	Python	>500k	122	0	0.00%

Applications are pseudonymised, following the Rings of Risk naming. Names, versions and vulnerability identifiers are withheld to respect coordinated vulnerability disclosure; each system is described by what it does, and size is given as a coarse band of non-test lines of code.

The aggregate move is stark: 80.55% recall across the smaller Python applications and 1.77% across the five large ones. Java is no better at 3 of 295 entries (1.02%). Six of the ten large systems returned nothing at all.

CAVEAT: TREAT THIS SECTION AS A RESEARCH SIGNAL, NOT A BENCHMARK SCORE

These are provisional ground-truth results published as a journal entry, outside the official RealVuln dashboard and leaderboard. The inventory of vulnerabilities in systems this large may have gaps, and the scores may move as labeling is completed. Only one general-purpose configuration has been run across the size range so far, and the security-specialized curve above 500,000 lines is projected rather than measured. **The direction of the finding is consistent and large. The precise values are not final.** Base any procurement decision on a validation run on your own repositories (Section 6).

■ SECTION 6

What this means for a regulated bank

Five operational conclusions follow, ordered by the point at which they create exposure.

1 A clean scan history is not evidence of a clean estate

Semgrep-class tooling missed 3,728 of 4,138 labeled vulnerabilities (90.1%), including every access-control and missing-authentication finding in the corpus. If your assurance file holds years of clean rule-based scan results, do not conclude that the estate is clean. Conclude that the instrument was blind to the flaw classes that matter most.

2 The backlog understates the exposure it purports to describe

Remediation runs on a queue, and a queue can only hold what was detected. If roughly nine in ten issues never enter it, the queue is a register of the subset your tooling could see, not a register of your vulnerabilities. The distinction becomes material when a regulator, an auditor or an insurer asks you to demonstrate that the register is complete. Documented exposure you have not scheduled is harder to defend than exposure you have not yet looked for.

3 Prioritize by chainability, not by scanner severity

The families that compose into an agent-executable path (broken access control, missing authorization, SSRF, deserialization) are exactly where syntactic detection returns zero. A backlog ordered by scanner severity, produced by a tool that is blind in those families, will systematically under-prioritize the flaws an autonomous attacker would use first.

4 Validate vendors at your scale, on your code

The benchmark's strongest general-purpose configuration returns 0% to 5.56% recall on systems above 260,000 lines. Test any vendor claim of enterprise coverage on a repository of that size before accepting it. A demonstration on a deliberately vulnerable 800-line sample says nothing about a core banking platform. Ask for recall on a system comparable to yours, with the ground truth open to inspection.

5 Measure remediation closure, not remediation intent

Detection is half the loop. The capability that finds a chain can run it again after the fix, and re-running the attack path is the only evidence that closure is real. Without post-remediation verification, a control is a process control, not a security control.

REGULATORY CONTEXT

DORA (Regulation (EU) 2022/2554), the FFIEC Information Technology Examination Handbook and the MAS Technology Risk Management Guidelines use different mechanisms, but they expect the same thing: **vulnerability handling that is continuous, evidenced and proportionate to the institution's risk profile**, backed by independent testing rather than a point-in-time attestation.

A metric that moves only when a vendor's rule set changes is hard to reconcile with continuous assurance. The benchmark data gives that discussion an objective input, the completeness of the detection layer, which until now has usually been asserted rather than measured. This report is evidence about tooling. It is not a compliance certification and it is not legal advice.

Limitations and threats to validity

Any evaluation has limits. These are ours, and they matter for how far the results should be pushed.

We are a vendor in our own benchmark

Kolega Labs wrote RealVuln and fields the leading scanner. We disclosed that conflict above. The only mitigations are full release of the data and code, and our reporting of results where competitors outperform us.

Scope is two languages, one target class

Version 3.1 covers Python and TypeScript/JavaScript only. The scored corpus is made of intentionally vulnerable and seeded applications: high density by construction, not production code with its realistic sparsity. Java and Go appear only in the provisional size study. Nothing here supports conclusions about memory-unsafe languages.

Ground truth is hand-labeled

Hand-labeling involves judgment. The accepted-CWE lists and the published evidence for each entry are designed to make disagreement possible, not to prevent it. Community challenges are welcome through the issue tracker.

LLM results are non-deterministic

Scanners built on language models produce different output from run to run, and several completed fewer than 140 repositories. Read single-run results with that uncertainty in mind. Strict scoring penalizes incomplete runs, but it cannot remove run-to-run variance.

Cost is API spend, not total cost of ownership

Cost figures reflect list prices at the time of the run. They exclude integration, triage labor and licensing for rule-based tools, and they are not comparable between categories.

What is measured is detection	The benchmark scores whether a scanner finds a labeled vulnerability in the right place with the right classification. It does not measure exploitability, business impact or the quality of a remediation recommendation.
Contamination	Some corpus repositories are public and may sit in model training data. The 280 false-positive traps are the main control against memorization inflating recall.

■ SECTION 8

Reproducing this

Every figure here comes from the released v3.1.0 dataset. The manifest locks the ground-truth content hash, the commit SHA of all 140 repositories and the default prompt hash, so a rerun can be checked against this document instead of taken on trust.

RELEASE	RealVuln 3.1.0, data generated 11 September 2026
GROUND-TRUTH HASH	sha256:0754363572503562e692414f826aad6f135f254851b8d12f70148bd235d23585
DEFAULT PROMPT HASH	sha256:3481f1432c23
SCANNING MODE	Agentic (tool-use loop) with a shared prompt and structured JSON output
METRIC	$F3 = 10 \cdot P \cdot R / (9P + R)$, strict micro-averaged, scaled to 100

SOURCES

RealVuln benchmark and live dashboard: realvuln.com.

Repository, ground truth, scanner outputs, scoring pipeline: github.com/kolega-ai/RealVuln-Benchmark.

Methodology paper: RealVuln: Benchmarking Rule-Based, General-Purpose LLM, and Security-Specialized Scanners on Real-World Code, arXiv:2604.13764.

Large-application results: The repository-size gap: our first provisional large-app results, RealVuln Journal, Issue 20, 30 September 2026.

Language-stratum analysis: The Python-to-TypeScript drop, RealVuln Journal.

Disclaimer. This report is provided for information. It is not legal, regulatory or investment advice. Benchmark results describe how named scanner configurations performed on a specific published corpus at a specific time. They do not predict performance on any institution's estate. Kolega Labs develops one of the scanners evaluated and wrote the benchmark, so readers should weight the corresponding results accordingly and are encouraged to reproduce or challenge them. All product names belong to their respective owners.

© 2026 Kolega Labs. Data from RealVuln v3.1.0 (2026-09-11). Report issued 23 October 2026.