Which LLM Should Score Your CVEs?

A Cost-Aware Benchmark for Context-Conditioned Vulnerability Prioritization

Fahed Dorgaa

venslabs | `github.com/venslabs/vens-benchmark`

Abstract

A scanner hands you 400 CVEs and their CVSS base scores. Most of them do not matter for *your* deployment, and CVSS was never meant to tell you which. `vens` closes that gap by asking an LLM to re-score each finding against a short project-context file (exposure, data sensitivity, criticality, controls, compliance) and the scanner report, producing an OWASP Risk Rating from 0–81. The obvious question from operators is: *which model do I put behind it, and is it worth the API bill?* We benchmark eight 2026 cloud models and four local models on the two things `vens` actually needs—reading a CVE and using the context—and compare every one against a non-LLM rule engine it has to beat. The headline: accuracy is a solved, cheap problem (a $0.48 model ties a $4.12 one), context use is real but uneven (reachability and accountability move correctly; a denial-of-service cue is mostly keyword-matched), and the weakest models—including every local one we tried—are statistically indistinguishable from a constant guess. We give a concrete model-selection table and release the harness so a practitioner can rank any new model in an afternoon.

1 Introduction

Vulnerability scanners are good at finding CVEs and bad at telling you which to fix first. CVSS base scores, the usual sort key, are a property of the vulnerability, not of the system it sits in: the same 9.8 RCE is a fire drill on an internet-facing payment API and a non-event in an air-gapped batch job. Teams end up hand-triaging, or worse, patching by score and running out of time.

`vens`[1] is an open-source tool that re-prioritizes a scan with an LLM. It takes two inputs only—the scanner report and a hand-written `config.yaml` describing the project—and emits a CycloneDX VEX document with an OWASP Risk Rating per CVE. It has no runtime access; the context file is the operator's declared truth and the CVE description is read to connect the two. Because the scoring is one LLM call per finding, the choice of model is a direct cost and quality knob that operators have to set with no data to go on.

This paper is that data. We ask:

RQ1 How well do current models read a CVE (predict its severity from the text alone), and how cheaply?

RQ2 Do they actually *use* the project context, or just echo the scanner?

RQ3 When a model gets a context case right, is it reasoning or keyword-matching the CVE text?

RQ4 What should a practitioner run, given a budget?

RQ5 Are free local models a viable option?

2 Background: what vens asks the model

`vens` scores four OWASP Risk Rating factors, each 0–9, and multiplies likelihood by impact:

$$\text{Risk} = \underbrace{\frac{\text{ThreatAgent}+\text{Vulnerability}}{2}}_{\text{likelihood}} \times \underbrace{\frac{\text{Technical}+\text{Business}}{2}}_{\text{impact}} \in [0, 81].$$

Three of the four factors are close to arithmetic on the context file: *ThreatAgent* tracks the exposure band, *Technical* tracks data sensitivity, *Business* tracks criticality plus a compliance bump. Only *Vulnerability*—how exploitable this CVE is in this deployment—needs the model to read and reason. This is exactly why a non-LLM baseline is not a strawman (§4): most of the score can be produced by a lookup table, so the model has to earn its cost on the parts that cannot.

3 Related work

CTIBench [1] is a CTI benchmark suite whose CTI-VSP task asks a model to produce a CVSS v3.1 vector from a CVE description; we reuse it directly for RQ1 so our accuracy numbers sit next to its published baselines. Databricks' VulnWatch [2] prioritizes CVEs with an LLM but scores the vulnerability in isolation; EPSS [3] predicts exploitation probability from global signals, again context-free. Reachability tools (callgraph/VEX generators) answer "can this code even run?" but not "does it matter here?". `vens`'s contribution, and what no public dataset labels, is risk *conditioned on a declared deployment context*; RQ2–RQ3 are, to our knowledge, the first benchmark of that specific behavior.

4 Benchmark design

We split "best model for `vens`" into the two skills it composes.

[1] `https://github.com/venslabs/vens`

Half A — CVE understanding (RQ1). The CTI-VSP task: 200 CVEs, predict the CVSS v3.1 vector from the description. We parse the vector with the `cvss` library and score the base value, so the model is never asked to do arithmetic. Metric is mean absolute deviation (MAD) from the NVD base score.

Half B — context use (RQ2–RQ3). 35 scenarios, each a one-CVE Trivy report plus a `config.yaml`, run through the real `vens generate` CLI with attestation on so we capture the four factors and the reasoning. Most scenarios are *contrastive*: hold everything fixed, vary one context axis, and check the score moves the right way (exposure ladder, sensitivity ladder, controls on/off, reachability declared in operator notes). Two are real showcase CVEs. Five are *discriminators*—cases where the naive lookup rule is wrong, so a model only gets them right by reading the CVE: a pure denial-of-service against a critical-data service (availability, not confidentiality), an accountability bug under low data sensitivity, and an irrelevant WAF in front of a local-only exploit.

The construct-validity trap (RQ3). A discriminator can be passed by keyword-matching ("the description says *denial of service*") rather than reasoning. For the DoS, accountability, and irrelevant-control families we add a *non-telegraphed twin*: same context, but the CVE text no longer names the impact class or asserts "no confidentiality impact." If a model's factor survives de-telegraphing, it was reasoning; if it drifts back toward the naive value, it was pattern-matching.

The baseline it must beat. A non-LLM rule engine reproduces `vens`'s own prompt arithmetic (Technical=sensitivity, Business=criticality +compliance, ThreatAgent=exposure band) and *copies the vendor severity* for the Vulnerability factor—because a rule engine cannot judge exploitability. This is a null model, not an oracle: divergence from it is exactly where the LLM adds value. We also report a constant "always 7.5" predictor as an accuracy floor.

Protocol. Temperature 0, seed 7, three repeats per cell. Models: OpenAI `gpt-5.4-nano/mini`, `gpt-5.5`; Anthropic `claude-haiku-4-5`, `claude-sonnet-4-6`; Google `gemini-2.5-flash-lite/flash/pro`; and four local models via Ollama[2] (`llama3.2`, `qwen2.5:7b`, `gemma2:9b`, `deepseek-r1:8b`). 95% CIs are bootstrap over the 200 CVEs where per-CVE predictions were saved.

[2] `https://ollama.com`

Table 1: Half A: CVSS prediction on 200 CVEs. MAD lower is better; a constant 7.5 guess scores 1.573.

Model	MAD	95% CI	$/200
claude-sonnet-4-6	**0.746**	[0.62, 0.88]	1.90
gpt-5.5	0.751	—	4.12
gpt-5.4-mini	0.836	[0.69, 1.00]	**0.48**
gemini-2.5-pro	0.879	[0.73, 1.03]	1.16
gemini-2.5-flash-lite	1.149	[0.98, 1.32]	0.07
gemini-2.5-flash	1.168	[1.00, 1.33]	0.34
gpt-5.4-nano	1.195	—	0.21
claude-haiku-4-5	1.288	—	0.52
local (Ollama, free):			
deepseek-r1:8b	1.479	[1.26, 1.72]	0.00
qwen2.5:7b	1.536	[1.35, 1.74]	0.00
gemma2:9b	1.643	[1.45, 1.84]	0.00
llama3.2	1.686	[1.53, 1.85]	0.00
constant 7.5 (floor)	1.573	—	0.00

5 Results

5.1 RQ1 — reading a CVE is a solved, cheap problem

Table 1 ranks all twelve models. The four strongest cloud models (`sonnet`, `gpt-5.5`, `gpt-5.4-mini`, `gemini-2.5-pro`) sit within overlapping confidence intervals—this is a *tier*, not a ranking. Two facts matter for operators. First, `gpt-5.4-mini` ($0.48/200) ties `gpt-5.5` ($4.12/200): on the Pareto front of accuracy × cost, **`gpt-5.5` is dominated**—`sonnet` is both more accurate and cheaper. Second, every 2026 model beats 2024's GPT-4 (MAD 1.31), but the cheap tier still trails 2024's Gemini-1.5 (1.09); progress is real but not uniform. Accurate models slightly under-rate (bias −0.15); the cheap Gemini models over-rate (up to +0.82), which for a triage tool is the safer direction but inflates the queue.

5.2 RQ2 — context use is real but uneven

On the two showcases the good models do the hard thing. A Log4Shell finding (CVE-2021-44228, CVSS 10.0) in a low-value, internal, message-lookup-disabled context drops to LOW on 7 of 8 cloud models; a medium axios token leak (CVE-2023-45857) on an internet PII portal under GDPR rises to HIGH on all 8. The one failure is telling: `gemini-2.5-flash-lite` *re-inflates* Log4Shell to 48.8/Critical—worse than the rule baseline, which de-escalates it correctly to LOW. Cheapness has a price.

The contrastive ladders (exposure, sensitivity, criticality, compliance) are monotone for nearly every model. The discriminators separate the field: *reachability* declared in operator notes moves the score for 6/8 models (`flash-lite` and `nano` ignore the notes entirely, scoring reachable and unreachable identically); *accountability* lifts the Technical factor on 8/8; the *irrelevant WAF* is correctly *not* credited against a local-only exploit by all. Per-model, a Wilcoxon test of (LLM − baseline) risk is

Table 2: RQ3: mean Technical factor, telegraphed vs. non-telegraphed CVE (naive rule in parentheses). Small drift = reasoning; upward drift = keyword-matching.

Model	Accountability (2) tel. → non-tel.	DoS (9) tel. → non-tel.
claude-sonnet-4-6	9.0 → 9.0	8.0 → 8.0
gpt-5.5	8.0 → 8.0	8.0 → 8.0
gpt-5.4-mini	8.3 → 8.0	7.7 → 8.0
gemini-2.5-pro	8.0 → 8.0	8.0 → 8.0
claude-haiku-4-5	8.3 → 8.0	7.0 → 8.5
verdict	**robust**	**fragile**

Table 3: RQ4: recommendation. SD = risk stdev over 3 repeats (lower = more auditable).

Use case	Model	SD
Signed / audited VEX	claude-sonnet-4-6	0.85
Best value	gpt-5.4-mini	3.19
Throwaway triage	gemini-2.5-flash-lite	0.00
Avoid	gpt-5.5, gpt-5.4-nano	—

significant for `gemini-2.5-flash`, `-pro`, and `gpt-5.4-mini`; crucially, `flash-lite` and `nano` have a median divergence of *zero*—on context, they add nothing over the lookup table.

5.3 RQ3 — some "context use" is keyword-matching

Table 2 is the honest part. When we strip the giveaway phrase, the **accountability** signal holds—Technical stays at 8–9 (naive rule: 2) with near-zero drift, so models genuinely infer that tampering with an audit trail is an integrity loss. The **DoS** signal does not: Technical was only marginally below the naive 9 to begin with (∼8), and it drifts back up when "denial of service" is removed. Under three different baseline threshold sets the accountability conclusion survives and the DoS one does not. So "the model understands availability-only impact" is largely an artifact of the CVE saying so; report it as weak.

5.4 RQ4 — what to run

Table 3 is the practitioner takeaway. Consistency is the sample standard deviation of the risk over three repeats; it is a reliability floor, not a quality measure (a model can be reproducibly wrong), but for auditable VEX you want it low. `claude-sonnet-4-6` is the default: best accuracy, second-most stable (SD 0.85), on the Pareto front. `gpt-5.4-mini` is the value pick but jittery (SD 3.19)—run $n \geq 3$ and take the median. `gemini-2.5-flash-lite` is the throwaway-triage option at $0.07/200, with the caveat that it re-inflates the hard case and adds little on context. `gpt-5.5` and `gpt-5.4-nano` have no niche: one is dominated, the other is jittery *and* adds nothing.

5.5 RQ5 — local models are not there yet

All four local models land below the cheapest cloud model on accuracy, and their CIs *overlap the constant-guess floor*: statistically they are indistinguishable from predicting 7.5 for everything, and they over-rate by +0.8 to +1.4. On context they are worse than the numbers suggest. The three non-reasoning models are extremely stable (SD 0–0.15), but by rigidity, not skill: `llama3.2` keeps Log4Shell at Critical regardless of context and returns an identical maxed score across the whole exposure ladder. Only the reasoning model, `deepseek-r1:8b`, actually engages the context—it is the one local model that de-escalates the showcase (7.5/LOW) and reads reachability—but it is jittery (SD 2.25) and still poor on Half A. For context-conditioned scoring today, "run a small local model" is close to "do not use an LLM."

6 Discussion

The split result is the useful one. Picking a model for CVE *accuracy* is nearly free: the cheap tier is good enough and the expensive tier buys nothing, so spend the budget elsewhere. Picking a model for *context* is where the money goes—and where the cheap models quietly fail, not by erroring but by collapsing onto the lookup table. An operator who benchmarks only accuracy (the easy, public number) will over-value `flash-lite` and `nano` and get a tool that ignores the very context that justified using an LLM. The two-half design exists to catch exactly that.

7 Threats to validity

Construct. Half B scenarios are synthetic (fictional CVE IDs) to avoid recall; the two showcases use real CVEs, where a model recalling the true severity makes the de-escalation harder, not easier. RQ3's non-telegraphed twins address the keyword-matching threat directly for two families but not reachability (which is legitimately operator-declared) or the ladders. **Statistics.** Pooled sign tests over model×CVE cells are pseudoreplicated; we report per-model counts instead, where N=8 makes 8/8 and 7/8 meaningful. CIs come from a single run's per-CVE errors and capture sampling over CVEs, not API nondeterminism. **Scope.** MAD against NVD treats the NVD vector as ground truth, which is itself context-free; that is the point of Half B. The baseline is a faithful mirror of `vens`'s prompt, so "beats baseline" means "the LLM adds something over `vens`'s own hard-coded arithmetic," not "beats a strawman." **Data.** CTIBench is CC BY-NC-SA 4.0 and is fetched at runtime, not redistributed.

8 Conclusion

For `vens`, and for context-aware CVE scoring generally: buy accuracy cheap, pay for context, and measure context directly—because the models that look fine on the public accuracy number are the ones that ignore your config. Use

`claude-sonnet-4-6` if you sign the output, `gpt-5.4-mini` if you do not, and skip the flagship. The harness, scenarios, and a one-line hook to benchmark any new model are released so the next model can be placed the day it ships.

References

[1] CTIBench: A Benchmark for Evaluating LLMs in Cyber Threat Intelligence. NeurIPS 2024. `https://github.com/xashru/cti-bench`

[2] Databricks. VulnWatch: AI-Enhanced Prioritization of Vulnerabilities. 2024.

[3] J. Jacobs et al. Exploit Prediction Scoring System (EPSS). FIRST. `https://www.first.org/epss/`