Different Metrics to see if our LLM-powered Model out / under performs auditors and other LLM-powered models:

The Top Five Metrics For Our Project

1. Vulnerability Detection Rate (VDR)

Why it matters: This is the simplest and clearest signal of whether our generated dataset actually contains the vulnerability that Slither found.

What it shows: "How often is the LLM correct about the existence of the intended vulnerability?"

Why this is especially important for us: Our entire dataset generator is built around each contract having exactly one intended vulnerability. Perfect setup for VDR.

2. False Positive Rate (FPR)

Why it matters: LLMs hallucinate and security auditors hate noise.

What it shows: "How often does the LLM invent vulnerabilities that Slither does not detect?"

Why important for us: Our LLM prompt forces 1–3 vulnerabilities, but Slither might detect fewer. FPR will expose when the model "over-creates" issues.

3. Category-Level F1 Score

Why it matters: Not all attack types are equally detectable or describable for an LLM.

What it shows: "Is the model equally good at identifying reentrancy, business logic bugs, arithmetic bugs, front-running, etc.?"

Why important for us: We generate contracts across 10 distinct categories in ATTACK_TYPES. Our data is balanced ⟶ perfect for per-category metrics.

4. Line-Level Localization Accuracy

Why it matters: A model that says "there is a reentrancy bug somewhere" is much worse than one that can point to:

lines: [24, 28]

What it shows: "How precise is the LLM at pointing to the actual vulnerable lines?"

Why important for us: Slither gives exact line numbers. LLM metadata includes line numbers by design. Our dataset is perfectly aligned for this.

5. Agreement Rate with Slither (Consensus)

Why it matters: Slither is deterministic and widely trusted. Agreement shows you're capturing baseline, ground-truth patterns.

What it shows: "How often does the LLM identify the same vulnerability Slither reports?"

Why important for us: Your project uses Slither as the "truth" to validate the dataset integrity. Agreement rate naturally emerges from our architecture.

1. Vulnerability Detection Rate (VDR)

Definition: Percentage of true vulnerabilities correctly identified. Why it matters: Directly comparable to Slither, human auditors, other LLMs. How to compute: # correctly detected vulnerabilities / # total labeled vulnerabilities.

Note:

You take a contract → run your LLM auditor → it outputs a list of vulnerabilities Then you take another LLM auditor (or Slither, or human audit) → they output their vulnerabilities Then you compare the two to see:

which vulnerabilities match

which ones only your model found

which ones the other auditor found but yours missed

This is precisely how we measure whether our LLM is doing better or worse

Feasibility:

Very easy to measure because we already have LLM-predicted vulnerabilities and Slither-generated ground truth to compare directly.

2. False Positive Rate (FPR)

Definition: % of findings the model flags that are incorrect or nonexistent. Why it matters: Real auditors care deeply about noise. Too many FP = unusable tool. Comparison: Many LLM and static tools generate huge FP noise—this measures how your system stacks up.

Note:

You take a contract → run your LLM auditor → it outputs a list of vulnerabilities. Then you take the ground truth (your metadata.json, Slither summary, or human audit) → these are the vulnerabilities that actually exist in the contract. Then you compare your model's findings to the truth to identify:

which vulnerabilities your LLM flagged but aren't real

which issues your model invented or hallucinated

whether competitor LLMs or Slither also hallucinate the same issues or avoid them

This is exactly how we measure how noisy or trustworthy our auditor is.

Feasibility:

Very easy to measure because anything the LLM reports that Slither does not detect can be counted automatically as a false positive.

3. Per-Category F1 Score (10 Exploit Classes)

Definition: F1 score for each of your 10 attack categories. Why it matters: Some attacks are harder (e.g., oracle manipulation) than others (arithmetic errors). This lets you evaluate:

Do certain attacks confuse the model?

Does fine-tuning improve category-level performance?

Note:

You take a contract → run your LLM auditor → it outputs vulnerabilities with categories. Then you compare those predicted categories to the true attack types from your metadata.json or Slither summary. This lets us measure how well our model performs for each individual vulnerability type, instead of only looking at overall accuracy.

Feasibility:
Easy to measure because both the LLM and Slither outputs map to the same 10 attack categories using our existing code.

4. Line-Level Localization Accuracy

Definition: Percentage of predicted line numbers that match the true vulnerable lines in metadata.json. This is a major differentiator between a toy model and a real auditor. Levels you can measure:

Exact match

±2 lines

±5 lines

This allows fair comparison to humans (who often approximate) and to Slither (which is exact).

Note:

You take a contract → run your LLM auditor → it outputs vulnerabilities and the specific line numbers where they appear. Then you compare those predicted line numbers to the true vulnerable lines from metadata.json or Slither. This lets us measure how precisely the model can pinpoint the location of the bug, not just whether it knows the bug exists.

Feasibility:
Easy to measure because both our LLM metadata and your Slither summaries already include exact line numbers.

5. Structured JSON Accuracy Rate

Definition: Fraction of model outputs that conform perfectly to your JSON schema (correct fields, no hallucinated keys, valid paths, valid severity enums). Why it matters: Fine-tuning should dramatically reduce malformed JSON. Human auditors do not output structured JSON consistently. Many LLMs are brittle with structure.

Note:

You take a contract → run your LLM auditor → it outputs a JSON vulnerability report. Then you validate that JSON against your official schema (fields like id, attacks, lines, severity, etc.).

This lets us measure whether the model can consistently produce valid, machine-readable output instead of broken or hallucinated structures.

Feasibility:
Structured JSON Accuracy is very easy to measure because our pipeline already checks whether the LLM's JSON parses successfully, so we just count how often it works versus fails.

6. Severity Classification Accuracy

Definition: Match between predicted severity (low/medium/high) and metadata/Slither labels. Why it matters: Severity misclassification is a real-world failure mode—a model that correctly finds bugs but mislabels severity is still dangerous.

Note:

You take a contract → run your LLM auditor → it outputs a vulnerability and assigns it a severity (low, medium, high). Then you compare that severity label to the true severity from your metadata.json or Slither summary. This lets us measure how well our model understands the risk level, not just the presence of a bug.

Feasibility:
Moderately easy to measure because both the LLM and Slither include severity labels, but severity can sometimes be subjective.

7. Explanation Quality Score (Human Evaluated)

Definition: Human auditors rate the clarity and correctness of explanations on a 1–5 scale. Why it matters: This is where LLMs shine and where humans often underperform (humans write vague notes). Rubric you can use:

correctness

clarity

reference quality

actionability

You can evaluate 20–30 samples to keep cost/time low.

Note:

You take a contract → run your LLM auditor → it outputs not only the vulnerability but also an explanation describing why it is dangerous. Then a human reviewer scores that explanation on a simple 1–5 rubric (clarity, correctness, depth, and usefulness). This lets us measure how effectively each auditor communicates the problem in a way developers and auditors can understand and act on.

Feasibility:
Hard to measure because it requires human reviewers to manually rate explanation clarity and correctness.

8. Agreement Rate with Slither (Static Tool Consensus)

Definition: Percentage of Slither-detected vulnerabilities the model also flags. Why it matters: Static analysis → extremely precise but narrow LLMs → broad but pattern-based High agreement shows your model is covering baseline deterministic issues.

You can also measure:

model detects vulnerabilities Slither missed

Slither detects vulnerabilities model missed

This gives a full coverage map.

Note:

You take a contract → run Slither, which produces a deterministic list of vulnerabilities. Then you run your LLM auditor on the same contract and compare which vulnerabilities match Slither's findings based on category and location. This lets us measure how often our model reaches the same conclusions as the most widely used static analysis tool in the Ethereum ecosystem.

Feasibility:
Very easy to measure because we already have normalized Slither summaries that can be directly compared to LLM outputs.

9. End-to-End Audit Time / Latency

Definition: Time from input → complete JSON output. Why it matters: Real auditors take days. LLMs take seconds. This is a key differentiator between models and across tiers:

baseline LLM

RAG enhanced LLM

fine-tuned LLM

You can report:

average time

95th percentile time

Note:

You take a contract → run your LLM auditor → measure how long it takes from input to final structured JSON output. You repeat this for competitor LLM auditors and measure their total response times under the same conditions. This lets you quantify how fast each auditing system performs, which directly affects usability, deployment feasibility, and developer adoption.

Feasibility:
Moderately easy to measure because we just need to add timing code around the LLM call and Slither run.

10. Consistency on Repeated Audits (Stability Score)

Definition: Run the same contract 5 times. Measure the similarity of results. Why it matters: LLMs are not deterministic unless using temperature = 0. But even at temperature 0, many models change line numbers or severity wording.

Measure: stability = # identical outputs / # total runs.

Fine-tuned models should significantly increase stability.

Note:

You take a contract ⟶ run your LLM auditor multiple times (usually 5–10 runs) ⟶ and compare whether the outputs are identical. You repeat this for competitor LLMs and track how often their outputs vary (different vulnerabilities, line numbers, severities, or JSON structures). This measures how predictable, deterministic, and reliable each auditor is.

Feasibility:

Moderately easy to measure because we need to run the same prompt multiple times and compare outputs, which requires minor script changes.