Evaluating LLM-Assisted Workflows for Smart Contract Vulnerability Detection Versus Traditional Analysis Tools

Hrvoje Vujcec | MSc in Cybersecurity, Privacy and Trust | South East Technological University

Poster summary of the revised dissertation draft

Repository & evidence pack
github.com/vujcec/sc-vuln-llm-vs-tools
Scan for archived implementation details

Benchmark	Workflows	Per-case budget	Main finding
52 cases	**3 modes**	**12 min**	**Tools win**
50 vulnerable + 2 benign	*Tools-only, LLM-only, hybrid*	*Frozen fairness controls*	*Best dependable full-coverage baseline*

Problem and research aim

Smart contract auditing remains difficult because vulnerabilities range from well-known coding mistakes to broader logic and workflow failures. The study evaluates whether LLM-assisted analysis improves real audit workflows when compared with established tools under the same operating protocol.

Research questions

- RQ1: How does an LLM-only workflow compare with tools-only on the same benchmark?
- RQ2: Does a hybrid workflow improve detection or mainly amplify noise?
- RQ3: Which vulnerability categories remain strong or weak across workflows?
- RQ4: How do runtime, clarity, and usefulness differ in analyst terms?

Method and frozen protocol

Design

- Comparative experiment with shared case set, taxonomy, scoring rules, and execution budget.
- Outputs normalised into a common findings schema before scoring.
- Assisted workflows repeated to observe execution variability.

Frozen settings

- Slither timeout 120s with JSON capture.
- Mythril timeout 240s; tx_count = 3.
- Foundry timeout 240s; 1024 fuzz runs.
- GPT-4.1-mini; temperature 0.2; max_tokens 1200; max_turns 2.

Hybrid policies

Expansive: retains all normalisable LLM findings within the allowed taxonomy.
Conservative: requires tool corroboration for unchecked low-level calls and evidence anchors for LLM-only reentrancy, access-control, arithmetic, and denial-of-service findings.

Benchmark construction

Case composition

Access control	11	Arithmetic	11
Reentrancy	11	Unchecked calls	11
DoS	6	Benign	2

SmartBugs Curated ingestion was deterministic, the selected slice was recorded in a manifest, and explicit vulnerability markers were stripped from source comments before LLM execution to avoid label leakage.

Ground truth

Each case stored Solidity source, compiler information, and one or more taxonomy–SWC labels used for scoring.

Workflow design

All three workflows used the same cases, taxonomy, and scoring pipeline. The main difference was where evidence originated and how findings were filtered before scoring.

Tools-only
Slither + Mythril + Foundry under frozen limits

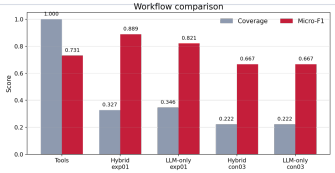
LLM-only
GPT-4.1-mini analyses contract source directly

Hybrid
Tool evidence added to LLM context, then gated

Normalisation
Shared findings schema: case, taxonomy, SWC, source, anchor

Scoring

Key quantitative results

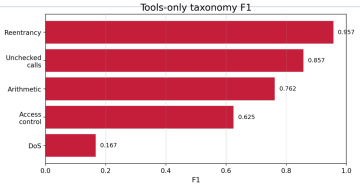


Interpretation

- Tools-only is the only fully completed benchmark result and therefore the most dependable baseline: TP/FP/FN = 34/9/16, micro-F1 = 0.731.
- Hybrid expansive run 01 reached the strongest completed-run micro-F1 (0.889), but coverage was only 32.7%, so the result is promising rather than decisive.
- Conservative run 03 produced small subset evidence only, with 22.2% coverage for both assisted modes.

Best-supported conclusion: traditional tools currently provide the strongest primary audit baseline, while LLM assistance is better treated as grounded triage and explanation support.

Taxonomy-level findings



What the baseline shows

- Reentrancy and unchecked low-level calls were the strongest tool-supported categories.
- Arithmetic issues were detected at a useful but more moderate level.
- Access control and denial-of-service remained the weakest areas, indicating limits where broader context matters more.

RQ4 analyst-oriented findings

The poster does not rely only on detection counts. RQ4 asked whether outputs were also clear and useful to a human analyst.

Hybrid exp01	LLM-only exp01
Mean usefulness 3.18	Mean usefulness 3.11
Mean clarity 1.76	Mean clarity 1.33

Assisted outputs could be helpful on completed cases, but clarity remained low and structure was not reliable enough for unverified use.

Implications, limits, and next steps

Implication: LLMs appear more credible as constrained assistants than as standalone smart-contract auditors.
Limitations: assisted runs showed low and unstable effective coverage; denial-of-service cases were few; interpretation still depended on a single analyst.
Next step: improve execution reliability, expand the benchmark, and test whether evidence-gated hybrid designs can keep their precision gains at full coverage.

The strongest LLM/hybrid figures are exploratory because they were obtained on incomplete runs and cannot yet displace the full tools baseline.

Selected references

[1] Feist, Grieco, and Groce (2019), Slither: a static analysis framework for smart contracts.
[2] Rameder, di Angelo, and Salzer (2022), review of automated vulnerability analysis of Ethereum smart contracts.
[3] Ferreira et al. (2020), SmartBugs benchmarking framework.
[4] Iuliano and Di Nucci (2024), updated SLR of smart contract vulnerabilities, tools, and benchmarks.
[7] Hu et al. (2023), LLM-powered smart contract vulnerability detection.
[14] Vujcec (2026), archived implementation reference and evidence pack.

Practical takeaway

A realistic adoption path is layered rather than substitutive:

- Use Slither, Mythril, and Foundry as the primary reproducible audit baseline.
- Use LLMs to explain, triage, and connect evidence, not to replace tool-grounded review.
- Require evidence-gating before hybrid findings are trusted in security-sensitive settings.

Poster prepared from the revised dissertation draft for MSc showcase use.