Task Report

Prepared by: Anushka Sharma
Date: 23 August 2025
Project: RAG layer + LLM Smart Contract Vulnerability Analyzer
GitHub Repository Link:
https://github.com/a-nushkasharma/Smart-Contract-Analyser-with-RAG-layer.git

---

Problems Addressed

- Single-pass LLM analysis can miss subtle vulnerabilities or produce conflicting results.
- Repeatedly resending the entire user smart contract to LLMs during multi-step which blows up token usage.

Objectives

- Implement a RAG layer to provide LLMs with relevant snippets for cross-verification.
- Reduce the token size.
- Deliver user-friendly visualization and downloadable reports via a Streamlit frontend.
- Scaffold a pipeline that allows future cycles of mutual refinement between LLMs.

RAG Layer

The RAG layer provides relevant code snippets to the LLMs during the cross-verification phase:

- **Purpose:** Helps LLMs verify and confirm vulnerabilities by consulting contract-specific evidence.
- **Functionality:** Retrieves indexed contract snippets related to a given vulnerability or evidence query.
- **Benefit:** Reduces hallucinations and improves consistency between LLM1 and LLM2 results

Completed Work

LLM Integration and Architecture

- Designed a modular multi-stage pipeline:
  LLM1 (GPT-4): Initial vulnerability identification
  LLM2 (Gemini): Verification and refinement
- Integrated prompt templating for different phases: analysis, cross-verification, and confirmation

Smart Contract Analysis Engine

- analyzer.py extracts Solidity functions for isolated analysis.
- Output annotated with vulnerabilities, metadata, and evidence references.

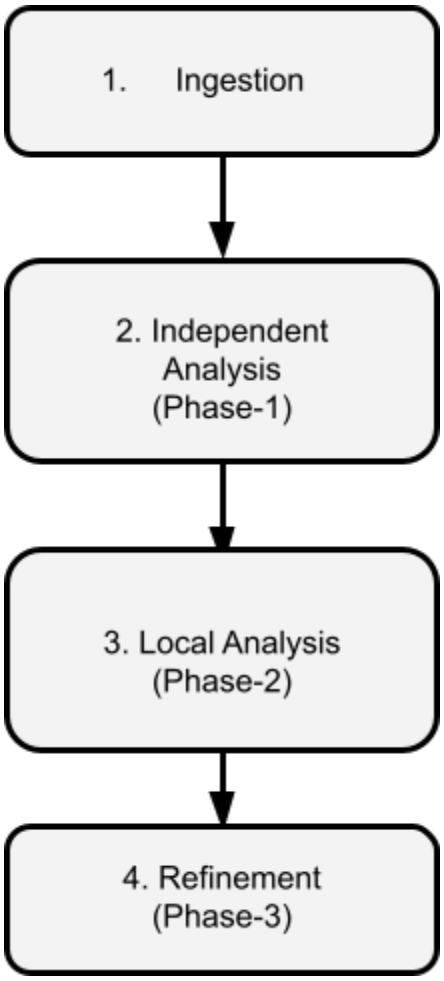
Report Generation

- JSON report includes:
  - Function name
  - Detected vulnerability
  - Confidence scores
  - Violated security properties

Frontend: Streamlit UI

- Contract upload and code preview
- Phase-wise analysis feedback
- Final report download as JSON

Flow-Chart



Phase 1: Ingest Contract

- Input: Solidity Smart Contract
- Action: Chunk/Index the contract content (
- Output: Contract ID

Phase 2: Full Contract Analysis

- Input: Contract Code
- Action: Run LLM1 and LLM2 on the full contract for a global scan
- Output: Initial Findings from both LLMs

Phase 3: Cross-Verification

- Input: LLM1 and LLM2 findings
- Action: LLM1 verifies LLM2 findings and vice versa using contract snippets
- Output: Verified/Disputed items

Phase 4: Confirm/Dispute

- Input: Phase 3 verification results
- Action: Each LLM confirms or disputes the verification results
- Output: Consensus decisions (confirm, dispute, needs more evidence)

Phase 5: Final Report

- Input: Phase 3 + Phase 4 results
- Action: Consolidate all findings into structured report
- Output: Final JSON report with statistics and vulnerability details