CoinFabrik

2025-Q4 Stellar Public Good Award

State of the Art: AI-Driven Smart Contract Vulnerability Research

27-Nov-25

Purpose of this document

The purpose of this document is to provide a comprehensive and technical overview of the current state of the art in AI-driven smart contract vulnerability research. It consolidates findings from academic literature, benchmark studies, and commercial solutions to clarify how the field is transitioning from static, rule-based analysis toward dynamic, agentic, and hybrid AI architectures. By examining advanced prompting strategies, multi-agent frameworks, retrieval-augmented systems, and data-driven commercial engines, this document aims to define the factors that currently determine high-performance vulnerability detection. It also highlights the limitations affecting non-EVM ecosystems and outlines the implications for future security tooling. Ultimately, the document serves as a strategic and technical reference for teams designing or adopting next-generation AI audit systems.

Contents

1.0 Executive Summary

This document presents a technical synthesis of the current "State of the Art" (SOTA) regarding the application of Artificial Intelligence (AI) to smart contract vulnerability research. The domain is currently undergoing a paradigm shift from static, rule-based detection to dynamic, agentic reasoning.

Current data indicates that traditional Static Application Security Testing (SAST) tools, such as Slither and Mythril, have reached a performance plateau. These tools fail to detect approximately 50% of vulnerabilities in modern benchmarks and suffer from precision rates below 10%. Conversely, the direct application of Large Language Models (LLMs)—such as zero-shot GPT-4—suffers from low recall (37.8%), resulting in the omission of critical flaws.

The SOTA solution lies in agentic frameworks and commercial hybrid engines. Architectures such as LLM-SmartAudit, SmartLLM, and commercial leaders including Sherlock AI and Zellic V12 have demonstrated that superior performance requires combining specialized prompt engineering with proprietary data moats (vast datasets of human-validated findings from audit contests). This report analyzes these architectures, validates the "Data Moat" hypothesis which currently restricts high-performance AI to Solidity, and explores the alternative strategies required for niche languages such as Rust and Move.

2.0 Prompt Engineering & Fine-Tuning

The efficacy of AI in this domain is defined not by model size alone, but by the structural guidance provided by prompt engineering and domain adaptation.

2.1 Advanced Prompting Strategies

- Chain-of-Thought (CoT): This technique compels the model to externalize its reasoning steps. Recent benchmarks utilizing the ParaVul framework demonstrate that CoT prompting achieves superior performance, reaching F1-scores of 0.9398 for single-label detection, significantly outperforming zero-shot approaches.
- Inception Prompting: This strategy is used to define deep role specialization. In the LLM-SmartAudit framework, this technique initializes agents with specific personas (e.g.,

"Solidity Programming Expert," "Project Manager") to partition the audit task into manageable cognitive loads.

- Step-Back Prompting: Utilized by the SCALM framework, this method prompts the LLM to abstract high-level concepts or principles from retrieved code snippets before attempting to identify specific bugs. This abstraction step is critical for identifying design-level "bad practices" rather than solely syntax errors.

2.2 RAG and Fine-Tuning

- Retrieval-Augmented Generation (RAG): Frameworks such as SmartLLM and SCALM employ RAG to inject external knowledge such as ERC standards and SWC registry definitions at runtime. This knowledge retrieval is essential for reducing false positives. For example, SCALM observed its F1-score for reentrancy detection drop from 94.97% to 75.35% when RAG was removed.
- Parameter-Efficient Fine-Tuning (PEFT): The EVuLLM framework demonstrates the utility of lightweight fine-tuning (using QLoRA) on open-source models like Llama-3. This allows smaller, edge-deployable models to achieve high accuracy by learning domain-specific opcode patterns without the computational cost associated with full model retraining.

3.0 Academic State-of-the-Art Architectures

The academic frontier has advanced beyond single-prompt queries to complex, multi-agent architectures that emulate human audit teams.

3.1 LLM-SmartAudit: Multi-Agent Collaboration

Repository: https://github.com/LLMAudit/LLMSmartAuditTool

This framework emulates a human audit firm by deploying a team of specialized agents:

- **Project Manager**: Orchestrates the workflow and assigns tasks.
- **Auditor**: The primary offensive agent responsible for identifying vulnerabilities.
- **Solidity Programming Expert**: The defensive agent that validates findings against language specifications.

The system uses inception prompting to fix agents within their roles and utilizes a task queue to manage the audit lifecycle.

3.2 SmartLLM: The Sequential Validation Pipeline

Repository: https://github.com/SMARTlab-Purdue/SMART-LLM

SmartLLM introduces a linear validation pipeline designed to eliminate hallucinations:

1. **Detector**: Scans for potential vulnerabilities.
2. **Reasoner**: Generates a justification regarding why the detected pattern constitutes a vulnerability.
3. **Verificator**: Cross-references the finding with a RAG database of ERC standards to validate the Reasoner's logic.

3.3 PromFuzz: AI-Guided Fuzzing

Repository: https://github.com/PROMFUZZ/promfuzz

PromFuzz represents a hybrid approach, combining LLMs with traditional fuzzing:

- **Dual-Agent System**: One agent analyzes the high-level business logic to identify potentially vulnerable functions, while a second agent generates invariant checkers specific to those functions.
- **Bug-Oriented Fuzzing**: These AI-generated invariants guide a traditional fuzzing engine, directing it toward violations of business logic.
- **Performance**: PromFuzz detected 30 zero-day bugs in real-world DeFi projects.

3.4 AIxCC 2025: The Validation of Hybrid Architectures

Repository: https://github.com/Team-Atlanta/aixcc-afc-atlantis

The theoretical shift toward hybrid architectures was empirically validated in August 2025 at the DARPA AI Cyber Challenge (AIxCC), where Team Atlanta secured the grand prize with their system, Atlantis. Rejecting monolithic AI models, Atlantis implements N-version programming, a fault-tolerant strategy that orchestrates multiple independent analysis engines with orthogonal strengths to maximize detection coverage.

Managed by a central CP-MANAGER, the system coordinates three specialized bug-finding modules: Atlantis-C, which employs BULLSEYE (a directed graybox fuzzer leveraging static analysis) for C/C++ targets; Atlantis-Java, which utilizes LibAFL and Jazzer for Java; and Atlantis-Multilang, an LLM-centric

engine that uses agents like MARTIAN and PRISM to synthesize inputs and refine patches. This heterogeneous architecture, which combines high-instrumentation fuzzing with agentic reasoning, successfully identified 18 zero-day vulnerabilities in critical infrastructure and generated valid patches in an average of 45 minutes, demonstrating the necessity of diverse neuro-symbolic tools for autonomous auditing.

4.0 Commercial Landscape

While academic tools rely on public datasets, commercial leaders have established dominance through "Data Moats"—proprietary repositories of high-quality vulnerability data that enable superior fine-tuning.

4.1 The "Data Moat" Hypothesis: Sherlock AI & Zellic V12

The efficacy of top-tier commercial tools is predicated on their access to non-public, labeled data from competitive audit platforms.

- Sherlock AI: This tool leverages a machine learning model trained on thousands of real vulnerabilities derived from Sherlock's own audit contests. By training on both the omissions and findings of thousands of human auditors, the model learns from a "Reinforcement Learning from Human Feedback" (RLHF) loop that is impossible to replicate with public data.
- Zellic V12: Zellic's acquisition of Code4rena (the largest competitive audit platform) secured a massive dataset of findings. Zellic V12 functions as an "autonomous auditor" that combines this offensive dataset with static analysis.

Crucially, these are not pure LLMs. They utilize a hybrid architecture, combining the reasoning of an LLM (fine-tuned on contest data) with the precision of traditional static analysis (SAST).

4.2 The Non-EVM Gap

Fine-tuning requires thousands of labeled examples distinguishing vulnerable code from secure code. The Ethereum ecosystem, being older, possesses a dataset containing millions of contracts and thousands of documented hacks.

Conversely, ecosystems such as Solana (Rust), Aptos/Sui (Move), and Starknet (Cairo) are younger and lack a historical dataset large enough to train a robust LLM. For non-EVM languages, the big data approach of fine-tuning is currently unviable. Security in these domains relies on hybrid architectures where the AI's role is reduced to interpreting the output of rigorous static analyzers or formal provers.

5.0 Conclusion

The state of the art has moved definitively beyond simple "bug finding" prompts. The most effective systems today are hybrid architectures. In the Solidity ecosystem, these hybrids (Sherlock, Zellic) leverage "Data Moats" of proprietary contest findings to fine-tune powerful models. In non-EVM ecosystems (Rust, Move), where data is scarce, hybrids rely on formal verification and static analysis to guide the AI. The future lies in the convergence of these technologies into "self-healing" contracts that can proactively generate their own security invariants and mitigate attacks in real-time.

References

[1] Static Application Security Testing (SAST) Tools for Smart Contracts: How Far Are We?

[2] Evaluation of ChatGPT's Smart Contract Auditing Capabilities Based on Chain of Thought - arXiv

[3] ParaVul: A Parallel Large Language Model and Retrieval-Augmented Framework for Smart Contract Vulnerability Detection - arXiv

[4] LLM-SmartAudit: Advanced Smart Contract Vulnerability Detection - arXiv

[5] SCALM: Detecting Bad Practices in Smart ... - AAAI Publications

[6] EVuLLM: Ethereum Smart Contract Vulnerability Detection Using Large Language Models

[7] SmartLLM: Smart Contract Auditing using Custom Generative AI - arXiv

[8] Detecting Functional Bugs in Smart Contracts through LLM-Powered and Bug-Oriented Composite Analysis - arXiv

[9] Flames: Fine-tuning LLMs to Synthesize Invariants for Smart Contract Security - arXiv

[10] Decompiling Smart Contracts with a Large Language Model - arXiv

[11] Prompt to Pwn: Automated Exploit Generation for Smart Contracts - arXiv

[12] FaultLine: Automated Proof-of-Vulnerability Generation using LLM Agents - arXiv

[14] What Can Generative AI Red-Teaming Learn from Cyber Red-Teaming? - Software Engineering Institute - Carnegie Mellon University

[15] LLM01:2025 Prompt Injection - OWASP Gen AI Security Project

[16] LLM Red Teaming: The Complete Step-By-Step Guide To LLM Safety - Confident AI

[17] Introducing V12 | Zellic — Research

[18] ATLANTIS: AI-driven Threat Localization, Analysis, and Triage Intelligence System