Evaluating the Vulnerability Landscape of LLM-Generated Smart Contracts

A Systematic Security Analysis

Hoang Long Do, Nasrin Sohrabi, Muneeb Ul Hassan (Deakin University)

Introduction: LLMs and Smart Contract Development

- LLMs are increasingly used to automate code generation in software development.
- This trend extends to blockchain, with LLMs generating smart contracts.
- LLMs lower the barrier to entry for smart contract creation.
- Deployed Smart Contracts can't be modified, security is paramount

Background: Smart Contracts - The Foundation of DApps

- Smart contracts as self-executing programs on decentralized platforms.
- Ethereum and the EVM revolutionized smart contract implementation.
- Smart contracts enable various applications (DeFi, governance, etc.).
- Immutability of smart contracts necessitates rigorous security.

The Core Problem: LLMs May Introduce Vulnerabilities

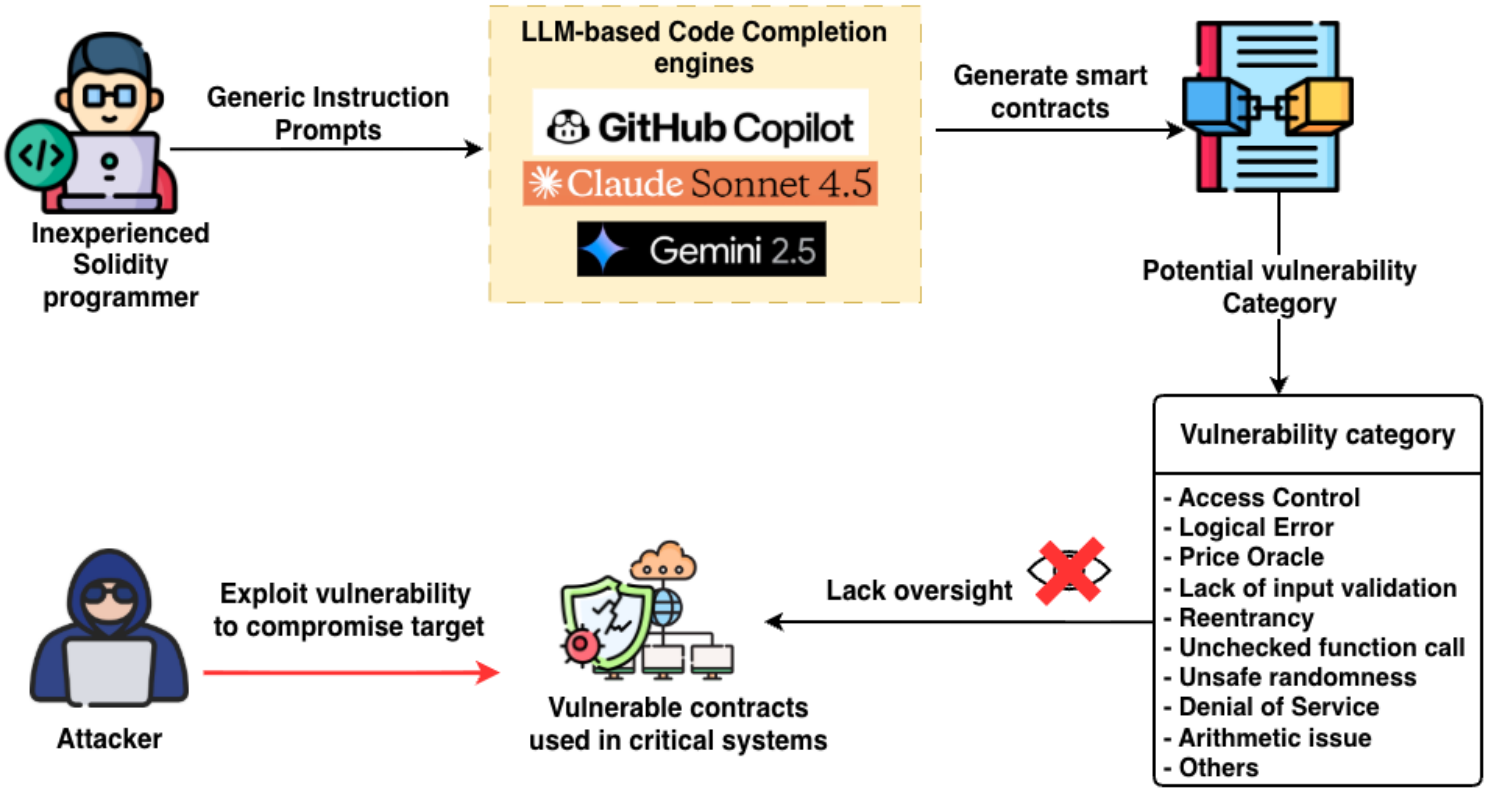
- LLMs can generate syntactically correct but insecure code.
- LLMs lack a deep understanding of adversarial behaviors and blockchain threat models.
- LLM-generated contracts may contain subtle vulnerabilities, hard to detect.
- Question: Are LLM-generated smart contracts secure enough for deployment?

Research Questions & Key Contributions

- Urgent to understand if LLM-generated Smart Contracts can be deployed for production?
- Key Contributions:
  - Comprehensive security evaluation of LLM-generated contracts.
  - Categorization of vulnerability patterns specific to LLM-generated contracts.
  - Detailed threat model and experimental analysis.
  - Practical insights and mitigation strategies.

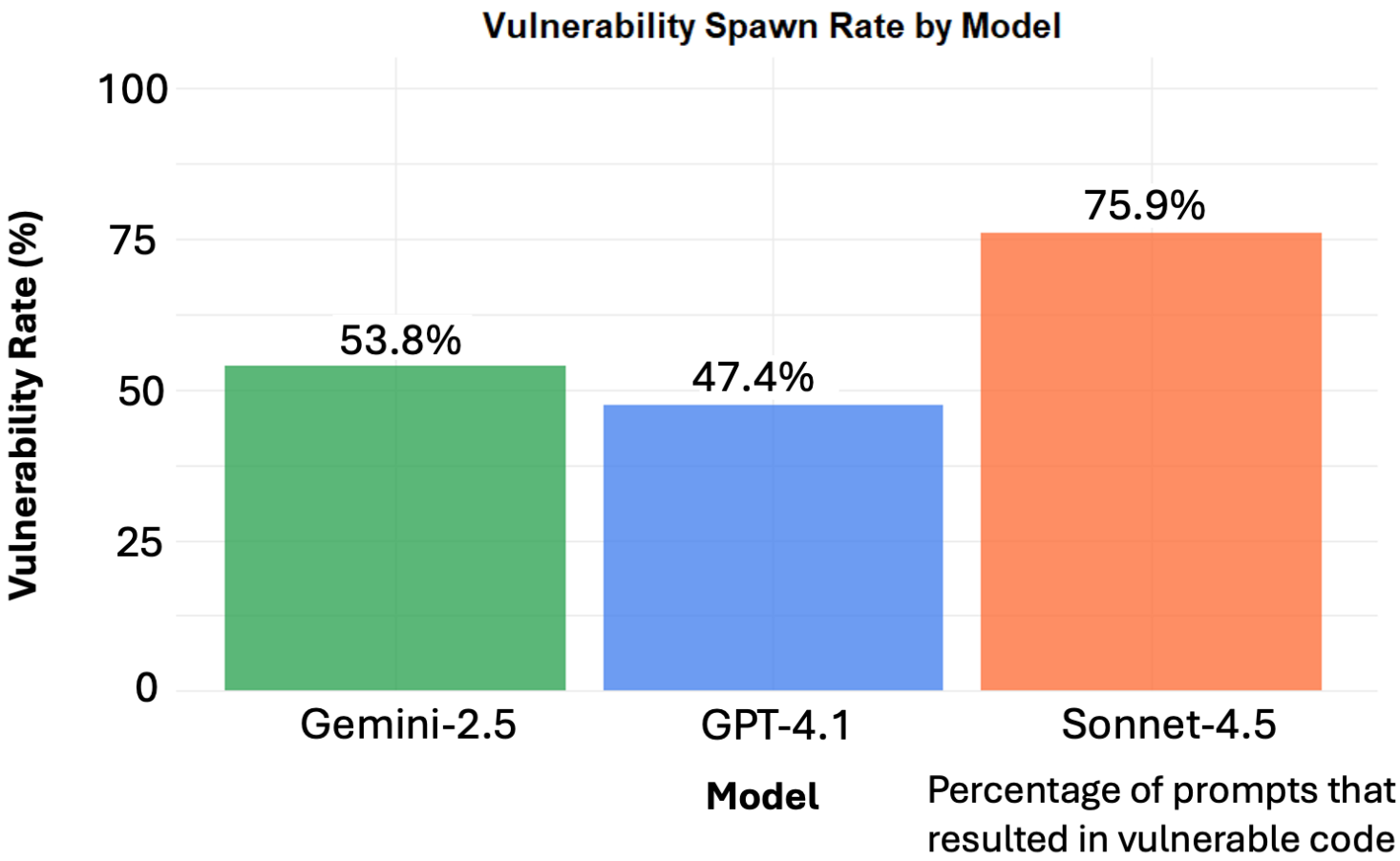
Threat Model

- Adversarial capabilities
- Attack Vectors
- Vulnerability Landscape



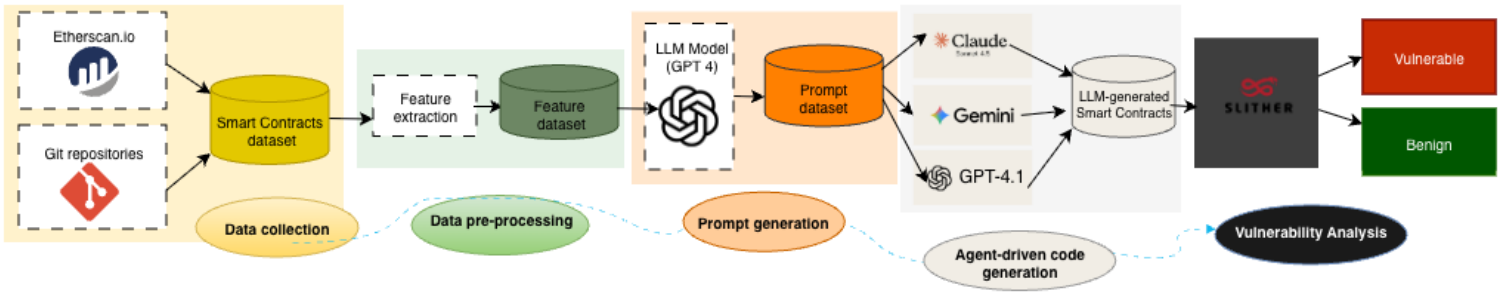
Vulnerability Spawn Rate Model

- Vulnerability Spawn Rate Model: Equation here
- Explanation of the parameters



Reference Architecture

- Diagram of the System Architecture
- Integration points between LLMs, Smart Contracts, and Scanners

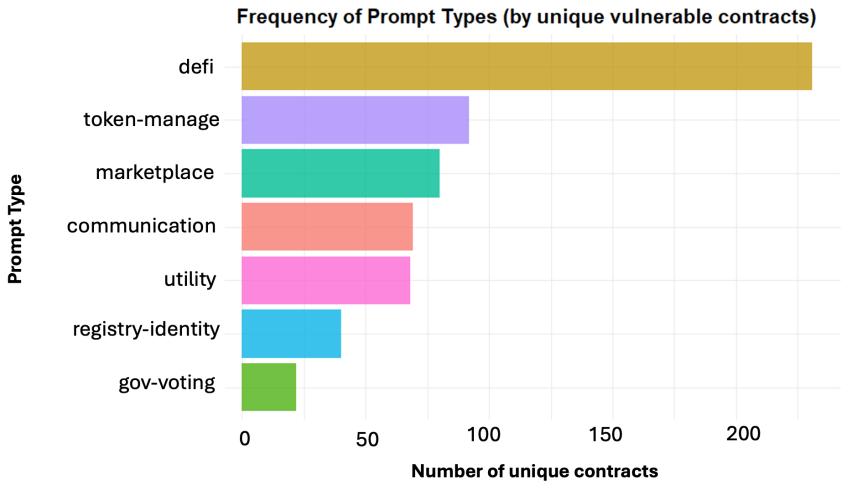


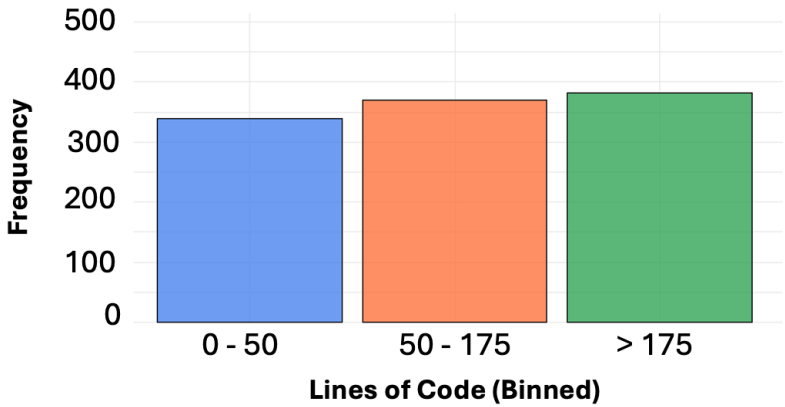
Experimental Evaluation

- LLMs Used: ChatGPT, Gemini, Sonnet.
- Application Domains: DeFi, Token Management, Governance, Marketplaces, Social Networks, etc.
- Vulnerability Detection Tools/Methods: (Mention the tools you used)
- Metrics: Vulnerability prevalence, types of vulnerabilities, Lines of Code (LOC), etc.

Key Findings: Vulnerability Prevalence

- Overall percentage of vulnerable contracts generated by each LLM.
- Comparison of vulnerability types across different LLMs (e.g., reentrancy, overflow).
- Vulnerable Contracts Comparison for each contract type/application domain
- Discuss any surprising or significant differences between LLMs.





Discussion: Root Causes and Implications

- Lack of adversarial awareness in LLMs.
- Incomplete understanding of blockchain-specific nuances (gas costs, EVM behavior).
- Potential for subtle vulnerabilities that static analysis might miss.
- Risks of deploying LLM-generated contracts directly to production.

```
contract Refunder {

  address[] private refundAddresses;
  mapping (address => uint) public refunds;

  constructor() {
    refundAddresses.push(0x79B483371E87d664 cd39491b5F06250165e4b184);
    refundAddresses.push(0x79B483371E87d664cd39491b5F06250165e4b185);
  }

  function refundAll() public {
    for (uint x; x < refundAddresses.length; x++) {
      require(refundAddresses[x].send(
        refunds[refundAddresses[x]]));
    }
  }
}
```

Conclusion & Future Directions: Towards Secure LLM-Assisted Smart Contract Development

- LLM-generated smart contracts are frequently vulnerable and unsuitable for direct deployment.
- Summarize the identified vulnerability patterns and their implications.
- Future Directions:
  - Research into safer prompting strategies for LLMs in smart contract generation.
  - Development of automated auditing tools tailored for LLM-generated code.
  - Formal verification techniques to guarantee the security of LLM-generated contracts.
  - Integration of secure coding principles into LLM training data.
  - Investigating hybrid approaches: combining LLM generation with human expert review processes.
- Increased awareness and responsible integration of LLMs into blockchain development workflows.