Smart Contract Vulnerability Detection Using Machine Learning

Lucas Kim
CMSI 5350 / EECE 5150
Loyola Marymount University
Los Angeles, CA, USA
lkim60@lion.lmu.edu

***Abstract*—Smart contracts on blockchain platforms like Ethereum are prone to critical vulnerabilities such as reentrancy and integer overflow, resulting in billions of dollars in financial losses. Manual auditing is time-consuming and error-prone. This project applies machine learning to automate vulnerability detection by analyzing Solidity source code features, including opcode frequency and Abstract Syntax Tree (AST) node patterns. We fine-tune and compare three classification models—Logistic Regression, Random Forest, and XGBoost—on a labeled dataset of over 12,000 Ethereum smart contracts. The expected outcome is that XGBoost achieves over 92% F1-score with less than 5% false positives, enabling faster and more scalable security audits.**

***Index Terms*—Smart Contracts, Vulnerability Detection, Machine Learning, Blockchain Security, Opcode Analysis, AST Features**

I. Overview of the Project

A. Motivation and Impact

Smart contracts power decentralized applications (dApps) in DeFi, NFTs, and DAOs, managing over $100 billion in assets as of 2025. However, 80% of hacks originate from code-level flaws [1]. Manual audits cost $10,000–$100,000 per contract and take weeks. An ML-based detector can reduce audit time by 90%, prevent exploits, and increase trust in blockchain ecosystems.

B. Problem Statement

Given a Solidity smart contract, predict whether it contains vulnerabilities (binary classification: vulnerable or safe).
Features: Opcode frequency (e.g., CALL, REVERT), AST node counts (e.g., function calls, modifiers), control flow patterns.
Target: Binary label (1 = vulnerable, 0 = safe).
Learning Objective: Maximize F1-score on imbalanced data (minimize false negatives).

C. Dataset Description

We use the *Smart-Contract-Dataset* [2], containing 12,000+ real Ethereum contracts collected from Etherscan (2020–2025). It includes Solidity source code and labels for 8 vulnerability types (reentrancy, overflow, etc.). Dataset size: 50 MB; 20% vulnerable. We focus on binary classification and subsample to 10,000 contracts for efficiency.

D. Proposed Approach and Expected Outcomes

1) **Preprocessing**: Parse Solidity with `slither-analyzer` and `py-solc-x` to extract opcode and AST features.
2) **Models**: Train Logistic Regression (baseline), Random Forest, and XGBoost with 5-fold cross-validation.
3) **Tuning**: Grid search on tree depth and estimators.
4) **Evaluation**: F1-score, precision, recall, confusion matrix.

Expected Outcomes: We expect XGBoost to achieve an F1-score of at least 85% with a false positive rate below 10%, outperforming logistic regression and random forest baselines by 5–10 percentage points, consistent with state-of-the-art results on similar vulnerability datasets [3].

E. External Resources

- Dataset: https://github.com/Messi-Q/Smart-Contract-Dataset
- Tools: `slither-analyzer`, `py-solc-x`, Scikit-learn, XGBoost

No code cloning—implementation from scratch.

II. Timeline

Week	Milestones
Week 11 (11/9)	Proposal submission (this document)
Week 12	Data download, cleaning, feature extraction
Week 13	Train 3 models, hyperparameter tuning
Week 14 (11/25)	Progress report (2–3 pages)
Week 15 (12/1)	Presentation slides
Final Week (12/9)	Final report (4 pages) + code

TABLE I
Project Timeline

References

[1] Chainalysis, "Crypto crime report 2025," https://www.chainalysis.com/blog/crypto-crime-2025/, 2025.
[2] Messi-Q, "Smart-contract-dataset," https://github.com/Messi-Q/Smart-Contract-Dataset, 2025.
[3] S. Team, "Smartbugs: A framework to analyze ethereum smart contracts," https://github.com/smartbugs/smartbugs, 2021.