Milestone 3
Open-Source Vulnerability Scanner for Cardano Smart Contracts

UnboundedMarket Team

April 13, 2025

Contents

1 Motivation

Smart contract vulnerabilities present a serious risk to Blockchain ecosystems, such as Cardano, and can lead to significant financial losses, undermining trust in the platform. Existing auditing processes mitigate this risk but are time-consuming and costly and may overlook critical bugs, which can hinder Cardano's adoption and growth. Recent studies have demonstrated the potential of large language models (LLMs) to identify coding errors and vulnerabilities in smart contract languages such as Solidity, which are used on Ethereum. Building on these findings, we aim to bring these capabilities to the Cardano ecosystems by fine-tuning a large language model to identify bugs in Cardano smart contracts.

While our tool is not intended to be a standalone solution for ensuring bug-free smart contracts, it provides an additional layer of security by finding potential bugs and vulnerabilities early in the development process. Smart contract audits typically involve multiple iterations; after identifying and addressing a bug or vulnerability, an additional audit is required to confirm the fix. By enabling early vulnerability detection, we hope that our tool reduces the number of audit iterations, reducing the overall cost and time of the process. Furthermore, it may identify subtle bugs that could be overlooked in manual audits. We thus hope that our tool will improve the accuracy and efficiency of smart contract audits, reducing vulnerabilities and increasing the security of smart contracts within the Cardano ecosystem.

This report represents the completion of the third milestone of our project. We start by detailing the motivation for our selected pre-trained model in Section 2 and proceed by describing the process of creating the dataset in Section 3. We then describe the fine-tuning steps in Section 4 and describe the evaluation procedure in Section 5.

2 Model Selection

For the base model, we selected the OpenLLaMA model[1]. This choice was motivated by the model's fully open-source nature, public accessibility, and demonstrated performance that aligns closely with other state-of-the-art language models, including LLaMA.

3 Dataset

Our dataset consists of a diverse collection of Cardano smart contracts developed using three languages: Aiken, Plutus, and OpShin. We sourced contracts from 42 publicly

[1] openlm-research/open_llama_3b_v2

available repositories, listed in our GitHub repository.[2]

To generate examples of vulnerabilities in smart contracts, we systematically introduced commonly encountered smart contract bugs using GPT-4o. Both the prompts used to generate these vulnerabilities and the corresponding code scripts for their introduction are publicly accessible.[3]

Initially, this process produced a dataset of 1213 modified smart contracts. Following a quality control step, we filtered out contracts where the intended vulnerability was not successfully introduced, resulting in a refined dataset with 947 contracts. Subsequently, we randomly partitioned this dataset into two subsets: a training set containing 870 contracts and a test set of 97 contracts for evaluation.

4 Fine-Tuning

We perform instruction tuning on the OpenLLaMA base model using the prepared training dataset of smart contracts. We train the model to detect whether a bug exists in the contract and generate a description of the bug. The fine-tuning process spans 20 epochs with a batch size of 10 contracts per iteration. Table 1 summarizes the key hyperparameters used during the training process. Our fine-tuned model is publically available on Hugginface. [4]

5 Evaluation

Here, we detail the performance metrics of our fine-tuned model and benchmark it against the performance of our base model. We evaluate the performance of our model using a held-out test dataset of smart contracts. Our evaluation focuses on the following metrics: Weighted precision, recall, and F1-score, as well as accuracy and perplexity. We select the weighted average precision, recall, and F1-score to deal with the imbalance of classes within our dataset.

Table 2 summarizes the performance metrics of both the baseline and fine-tuned models.

[2] https://github.com/unboundedmarket/cardano-smart-contracts/blob/main/repositories.csv

[3] Prompts and scripts are available at https://github.com/unboundedmarket/cardano-smart-contracts/blob/main/repositories.csv and https://github.com/unboundedmarket/smart-contract-vulnerabilities/blob/main/scripts/create_vulnerabilities_data.py, respectively.

[4] https://huggingface.co/unboundedmarket/vulnerabilities-openllama-3b

Hyperparameter	Value
Model Type	LlamaForCausalLM
Tokenizer Type	LlamaTokenizer
Load in 8-bit	True
Load in 4-bit	False
Sequence Length	2048
Micro Batch Size	10
Gradient Accumulation Steps	1
Epochs	20
Optimizer	adamw_bnb_8bit
Learning Rate	2e-4
LR Scheduler	Cosine
Warmup Steps	20
LoRA Adapter	True
LoRA Rank (r)	8
LoRA Alpha	16
LoRA Dropout	0.0
Precision	fp16
Weight Decay	0.1
Gradient Checkpointing	True
Flash Attention	True
Validation Set Size	7%

Table 1: Hyperparameters for Fine-Tuning the OpenLLaMA Model

Model	Precision	Recall	F1-score	Accuracy	Perplexity
Baseline	0.861	0.928	0.893	0.928	63.286
Fine-Tuned	0.945	0.948	0.947	0.948	7.866

Table 2: Performance of Baseline and Fine-Tuned Models

6 Discussion

Our fine-tuned model outperforms the baseline model across all metrics. Precision, recall, and F1-score all show improvements, indicating enhanced model effectiveness in correctly identifying vulnerabilities in smart contracts. Furthermore, the substantial reduction in perplexity from 63.286 in the baseline model to 7.866 in the fine-tuned model demonstrates improved predictive certainty and confidence in generating explanations of the bugs. Given the weighted averaging approach due to dataset imbalance, the weighted precision matches the accuracy value.

Overall, these improvements highlight the potential of our vulnerability scanner to detect and explain bugs in Cardano smart contracts effectively. We believe this capability can significantly reduce the need for manual auditing, lower associated costs, and enhance the overall security and reliability of the Cardano blockchain ecosystem. That said, as with our initial report, we emphasize that this tool is intended to assist developers and auditors, not to replace extensive, manual audits.

Looking ahead to our next milestone, we aim to further enhance the model's performance and conduct more extensive testing to gain deeper insights into the specific scenarios where our vulnerability scanner excels and where it falls short.