Streamlined Interface for Detection of Vulnerabilites and Deployment of Smart Contracts

1st Lohith J J
Dept. of Computer Science and Engineering
National Institue of Technology
Tiruchirapalli, Tamilnadu, India
lohithjj@gmail.com

2nd Shouri Muralidharan
Dept. of Computer Science and Engineering
B.M.S College of Engineering
Bengaluru, Karnataka, India
shouri.muralidharan248@gmail.com

3rd Shreehari Kulkarni
Dept. of Computer Science and Engineering
B.M.S College of Engineering
Bengaluru, Karnataka, India
shreeharim7@gmail.com

4th Bindhu J S
Dept. of Computer Science and Engineering
B.M.S College of Engineering
Bengaluru, Karnataka, India
bindhujs19@gmail.com

5th Nisarga S
Dept. of Computer Science and Engineering
B.M.S College of Engineering
Bengaluru, Karnataka, India
snisarga8@gmail.com

***Abstract*—Blockchain is a revolutionary technology that enables users to communicate in a trust-less manner. It revolutionizes the modes of business between organizations without the need for a trusted third party. It is a distributed ledger technology based on a decentralized peer-to-peer (P2P) network. It enables users to store data globally on thousands of computers in an immutable format and empowers users to deploy small pieces of programs known as smart contracts. The blockchain-based smart contract enables auto enforcement of the agreed terms between two untrusted parties. There are several security vulnerabilities in Ethereum blockchain-based smart contracts, due to which sometimes it does not behave as intended. Because a smart contract can hold millions of dollars as cryptocurrency, so these security vulnerabilities can lead to disastrous losses.**

Ongoing smart contract attack events have seriously impeded the practical application of blockchain. Although lots of researches have been conducted, they mostly focus on off-chain vulnerability detection. However, smart contracts cannot be modified once they have been deployed on-chain, thus existing techniques cannot protect those deployed contracts from being attacked. To mitigate this problem, we propose a general smart contract interface name Streamlined Interface for Detection of Vulnerabilities and Deployment of Smart Contract, which can automatically patch vulnerable deployed contracts without changing the contract codes. The core insight of is Streamlined Interface for Detection of Vulnerabilities and Deployment of Smart Contract to generate patch contracts to abort malicious transactions in advance. Taking the three most serious bug types (i.e., reentrancy, arithmetic bugs, and unchecked low-level checks) as examples, we present how Streamlined Interface for Detection of Vulnerabilities and Deployment of Smart Contract automatically repairs them on-chain.

***Index Terms*—Ethereum Virtual Machine (EVM), GETH, Smart Contracts, Vulnerabilities, Machine Learning, Blockchain**

I. Introduction

Several strategies have been put forth to safeguard deployed contracts. With the help of examples like Sereum, which detects whether transaction executions are consistent with known vulnerability patterns and dynamically avoids reentrancy attacks by watching them. Sereum, however, can only deal with re-entrancy problems.

To do this, we put out a streamlined interface for vulnerability identification and smart contract deployment that can provide patches to close security gaps in already-deployed vulnerable smart contracts. We must fix contracts without changing their codes because nothing stored in blockchain can be tampered with, which is incredibly difficult and hard. The main concept of this study is to use independent smart contracts that have security rules (patches) incorporated in them to prevent fraudulent transactions from happening in the first place.

We specifically improve Ethereum Virtual Machine (EVM) to assist transactions verification and contracts binding. When presented with a weak smart contract, the streamlined interface for vulnerability identification and smart contract deployment first automatically creates a patch contract with secure rules based on the repair template, then deploys the patch on blockchain as a regular

contract. Second, in order to repair the contract, the owner of the vulnerable contract must issue a unique transaction through our improved EVM. Finally, the fix that has been submitted will be used to verify any transaction that uses the vulnerable contract. Afterward, transactions that may expose vulnerabilities will be stopped. Therefore, it is possible to defend against attacks on the original susceptible contract.

II. Problem Statement

Given that smart contracts are a relatively new technology, there are a number of vulnerabilities that are yet to be discovered. Hence a need to discover more vulnerabilities.

With the existing Vulnerability detection tool like Mythril,there is a need to develop a tool that provides both detection of vulnerability and generating patches of the code to overcome them.

III. Literature Survey

A. Key Terminologies

1. **Ethereum Virtual Machine** : The Ethereum Virtual Machine (EVM) is the component of the Ethereum network that manages the deployment and execution of smart contracts. It's sole purpose is to keep the continuous, uninterrupted, and immutable operation of this special state machine.

2. **Smart Contract** : It is a simple piece of code that runs on the Ethereum blockchain. They can be interacted with by other wallet addresses according to the program written. The pragraming language used to write such codes is calledd Solidity.

3. **Vulnerabilities** : These are loop holes in smart contracts that can be exploited in a number of ways that lead to loss of funds or destruction of the network or enabling the execution of some malicious code intended to do harm to the network.

4. **Blockchain Network** : A shared ledger of data like transactions or code(smart contracts) are batched into units called blocks. These have to be verified by the network consisting of users(compute nodes) that validate the authenticity of the block through a consensus mechanism.

5. **Layer 1** : Layer 1 is nothing but the blockchain itself. Consisting of data, network and consensus protocol stacks, it is upon this layer that all application or layers are built.

6. **Layer 2** : Any solution built to improve the scaling of Layer 1, without compromising security or decentralization.

7. **Hardhat** : An environment that mimics Ethereum network and facilitates testing of code in a test environment.

B. Existing System

Figure 1 Shows Structure of the blockchain The most of the detection tools that are currently in use , detect Vulnerability in the smart contract Before deploying them on chain, problem being is that once deployed on chain block chain smart contracts become immutable.

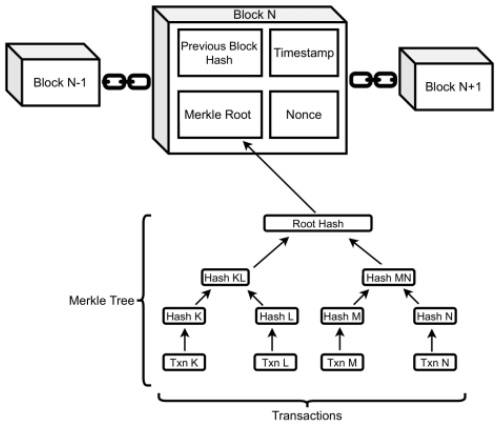

Fig. 1. Structure of blocks and blockchain

Smart contract vulnerabilities can be grouped into 3 broad categories :

1: Solidity Programming Language
2: Features of Ethereum Virtual Machine
3: Design Features of Ethereum Blockchain

1) **Vulnerability Detection using CNN architecture** :

A code targeted CNN architecture that aims at extracting semantics from written code and then analysing them in two steps.

I Data Pre-Process:Figure 2: shows the first stage of the CodeNet, data preprocessing step, Which Majorly Involves Compiling the Code to generate the byte code and finally mapping these bytecode

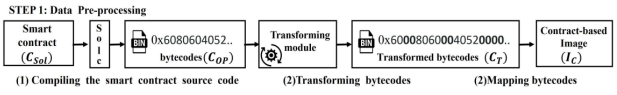

Fig. 2. Data Pre Processing phase of CNN based system

II Vulnerability Detection: Figure 3 Shows the Vulnerablity Detection Phase Key Point to be noted here is that,This CNN based architecture does not use Stride so that important instructions and feature of the smart contracts are not lost

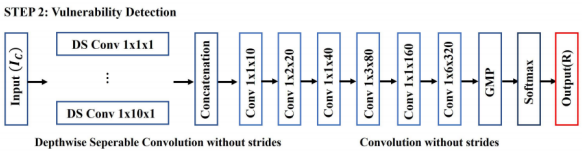

Fig. 3. Vulnerability Detection phase of CNN based system

2) **Automatic Repair FrameWork For On Chain Smart Contracts**

Using the three most dangerous varieties of bug types, a novel technique described in suggests how to create patch contracts to stop harmful transactions before they happen (Re-Entrancy,arithmetic bugs,unchecked-low level checks).The goal is to build a Counter for Automatic On-Chain Contract Protection without changing the original contracts, take precautions against numerous vulnerabilities. Based on backward slicing and data flow analysis, an effective method for autonomously synthesising patch contracts, using a framework called AROC that comprises of an on-chain contract protection system and an autonomous patch synthesis mechanism.It has three part:

1.Info Extractor Module
2.Patch Synthesis Module
3.Patch Binding

3) **Tools to detect Vulnerabilities**

One of the availablle toold that suits our problem statemen is SMARTEMBED, which uses code embedding and similarity checking to target clone identification and bug discovery as a single effort. SMARTEMBED can recognise similar smart contracts for clone identification. With the help of a bug database, bugs that are comparable to any known bug in the database in any smart contract provided by solidity developers as well as any existing contracts on the Ethereum network may be cross referred and have a patch code written. To achieve this, the entire system flow is broken into two parts :
a.**Training phase** : **Step1:** For each smart contract available in the dataset, the parser creates an abstract syntax tree (AST), and based on the kinds of the tree nodes, serialises the tree into a stream of tokens.
Step2: The normalize reassembles the token streams to remove inconsequential variations between smart contracts (such as the stop word, values of constants, or literals).
Step3: Following the input of the output token streams, a code embedding learning module embeds each code fragment into a fixed-dimension numerical vector.
Step4: The source code embedding matrix contains all of the source code, and the bug embedding matrix contains all of the bug statements that we gathered.
b.**Prediction phase** : **Step5:**In the prediction phase, any given new smart contract is turned into embedding vectors by going through the steps 1,2,3 and utilizing the learned embedding matrices
Step6: similarity thresholds are used to govern whether a code fragment in the given contract will be considered as code clones or clone-related bugs

IV. Proposed System

The proposed system is aimed at solving the absence of a singular interface/platform where all the features of multi-layer deployment, simple IDE for writing solidity code and vulnerability detection.The system-architecture of the proposed system is as follows :

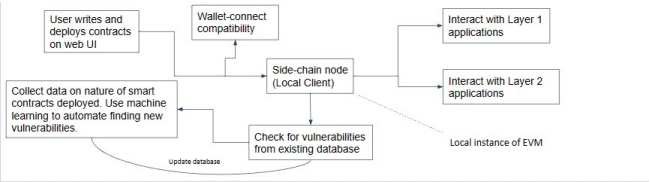

Fig. 4. Proposed System Architecture

1) Abstract Specification of sub-system:

- **Web UI** : A simple interface for users to write and see code. It is through this interface that users can connect their wallets, deployment to different layers and test their code.

- **Database** : This contains the current historical data of different vulnerabilities and their respective patches of code. With new contracts being deployed, the system look for patterns and keeps testing them in a local test-net environment. New found vulnerabilities are then updated and fed back into the system.

- **Side-chain Node** : This node will operate off the mainnet, while acting as a gateway for users to interact with either layer 1 or layer 2.

- **Contracts** : Smart contracts are pieces of code that are used to interact with the blockchain. It is this code that has a lot of vulnerabilities and is susceptible to attacks.

The user only interacts with the WebUI and wallet connect interface while writing and compile their smart contract code, as shown in Figure 4. The side chain node acts as a simple broker that sits between user and network. The intention of the system is to act as a mediator before a smart contract is deployed.

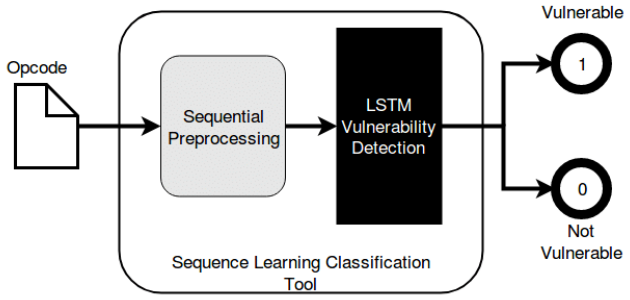

Fig. 5. Existing system to detect vulnerability

Figure 5 Shows the existing system to detect vulnerabilities in smart contract,From the diagram it clear that Before Smart Contract is deployed on the chain,difference detection techniques are applied,The proposed technique here is to use technology which identifies the vulnerability after the Smart Contract is deployed on the chain [**?**].

1)Info Extractor Module, Which is responsible for extracting the Variable Dependence and Path Constraints.
2) Path Synthesis Module,Given all such data it is the responsibility of the path synthesis module to generate the patch template for the given Vulnerability.
3)Enhanced EVM Miner Binds the Malicious Contract with Patch Contract, Thus preventing any instructions from accessing the malicious Smart Contract Figure 6 Shows

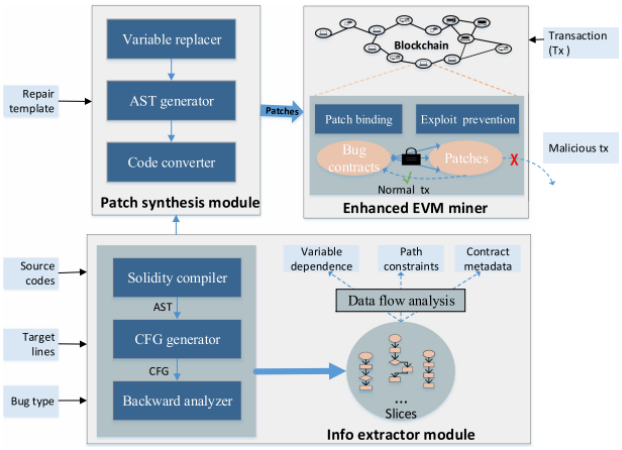

Fig. 6. Structure of Our Proposed Technique

Three Modules Which are essential part of the technique proposed

V. Conclusion

In conclusion, a consolidated and streamlined approach to writing smart contracts, detecting vulnerabilities and deploying then can be built using the proposed system. It will reduce the complexes arising from having to go to multiple different service providers for a single use case.

Summary of future development :

- Integration of new chains, with their respective methodologies.
- Making decentralised server/nodes available to deploy and maintain applications.

References

[1] Huashan Chen, Marcus Pendleton, Laurent Njilla, and Shouhuai Xu. A survey on ethereum systems security: Vulnerabilities, attacks, and defenses. ACM Computing Surveys (CSUR), 53(3):1–43, 2020.

[2] eepak Prashar et al. Analysis on blockchain vulnerabilities attacks on wallet. In 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), pages 1515–1521. IEEE, 2021

[3] Xiao Yi, Yuzhou Fang, Daoyuan Wu, and Lingxiao Jiang. Blockscope: Detecting and investigating propagated vulnerabilities in forked blockchain projects. arXiv preprint arXiv:2208.00205, 2022

[4] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. Sok: Layer-two blockchain protocols. In Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24, pages 201–226. Springer, 2020

[5] Yuntao Wang, Zhou Su, Ning Zhang, Rui Xing, Dongxiao Liu, Tom H Luan, and Xuemin Shen. A survey on metaverse: Fundamentals, security, and privacy. IEEE Communications Surveys Tutorials, 2022.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] A Averin and O Averina. Review of blockchain technology vulnerabilities and blockchain-system attacks. In 2019 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon), pages 1–6. IEEE, 2019.

[8] Ananda Badari and Archie Chaudhury. An overview of bitcoin and ethereum whitepapers, forks, and prices. Forks, and Prices (April 26, 2021), 2021.

[9] eon-Jin Hwang, Seok-Hwan Choi, Jinmyeong Shin, and Yoon-Ho Choi. Codenet:Code-targeted convolutional neural network architecture for smart contract vulnerability detection. IEEE Access, 10:32595–32607, 2022

[10] Hai Jin, Zeli Wang, Ming Wen, Weiqi Dai, Yu Zhu, and Deqing Zou. Aroc: An automatic repair framework for on-chain smart contracts. IEEE Transactions on Software Engineering, 48(11):4611–4629, 2021

[11] Zhipeng Gao, Vinoj Jayasundara, Lingxiao Jiang, Xin Xia, David Lo, and John Grundy. Smartembed: A tool for clone and bug detection in smart contracts through structural code embedding. In 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), pages 394–397. IEEE, 2019. Department of Computer Science and Engineering, BMS College of Engineering 19