Identification & Curtailment of Honeypots on Decentralized Blockchain Networks using Machine Learning and Code Analysis

विश्वजीवनामृतं ज्ञानम्

Submitted By:

Ojaswa Sharma (2017IMT-102)

Supervised By:
Prof. Shashikala Tapaswi
Dr. Kiran Kumar Pattanaik

ABV - Indian Institute of Information

Technology and Management

Gwalior - 474015

Background

A blockchain is a decentralized, distributed, and oftentimes public, digital ledger consisting of records called blocks that is used to record transactions across many computers so that any involved block cannot be altered retroactively, without the alteration of all subsequent blocks. This allows the participants to verify and audit transactions independently and relatively inexpensively. A blockchain database is managed autonomously using a peer-to-peer network and a distributed timestamping server. They are authenticated by mass collaboration powered by collective self-interests. Such a design facilitates robust workflow where participants' uncertainty regarding data security is marginal. A blockchain has been described as a value-exchange protocol.

Many modern day blockchains such as Tezos, Ethereum, Hyperledger Fabric etc. enable the concept of "Smart Contracts". Smart contracts are lines of code that are stored on a blockchain and executed automatically when predetermined terms and conditions are met without the participants requiring to trust one another. A smart contract does not necessarily constitute a valid binding agreement at law. Some legal academics claim that smart contracts are not legal agreements, but rather means of performing obligations deriving from other agreements such as technological means for the automation of payment obligations or obligations consisting in the transfer of tokens or cryptocurrencies.

In the cryptocurrency space, smart contracts are digitally signed in the same way a cryptocurrency transaction is signed. The signing keys are held in a crypto wallet. Byzantine fault-tolerant algorithms allow digital security through decentralization to form smart contracts. Additionally, the programming languages with various degrees of Turing-completeness as a built-in feature of some blockchains make the creation of custom sophisticated logic possible.

Motivation

A blockchain-based smart contract is visible to all users of said blockchain. However, this leads to a situation where bugs, including security holes, are visible to all yet may not be quickly fixed. Issues in Ethereum smart contracts, in particular, include ambiguities and easy-but-insecure constructs in its contract language Solidity, compiler bugs, Ethereum Virtual Machine bugs, attacks on the blockchain network, the immutability of bugs and that there is no central source documenting known vulnerabilities, attacks and problematic constructs.

Malicious contracts that look vulnerable but are exploitative are a rising trend. Recently, cyber attackers and criminals have started using these contracts to beguile users by deploying contracts that pretend to give away funds or services, but in fact contain hidden traps. These types of attacks were then named by the community as "***Honeypots***".

These malicious contracts share one trait in common: they almost always try to look like they were designed by a beginner. As such, they are a great place to learn about some of the pitfalls that can befall a new entrant to the space, and serve as an interesting case study into the wild-west world of smart contract security. By analyzing a few of the more interesting cases of not-so-vulnerable contracts, we can gain a deeper understanding of how smart contract security works in practice.

This works as a motivation to create and deploy an efficient and robust system to help users detect malicious HoneyPots and save them from fraudulent transactions. With a rising trend in decentralized networks and a global shift to the usage of Smart Contracts, a curb against these types of attacks is utmost necessary. Unfortunately, the significant lack of research, identification techniques and deployable products to counter Smart Contract vulnerabilities have continued to encourage attackers to follow these kinds of malpractices.

Project Objective

1. To implement a system to detect and classify various types of honeypots on Smart Contracts.
2. To implement machine learning models in order to make the system capable of detecting new and unknown types of honeypots, including honeypots exploiting zero-day vulnerabilities.
3. To develop an approach to generalize existing implementations and extend them to be more robust and efficient
4. To expose this implementation in the form of Service as a Service (SaaS) to allow users/developers to integrate the service into their systems and wallets for real-time protection against honeypots.
5. To allow one or more methods of SaaS consumption in the form of Plugin, Application, or API.

Implementation Feasibility

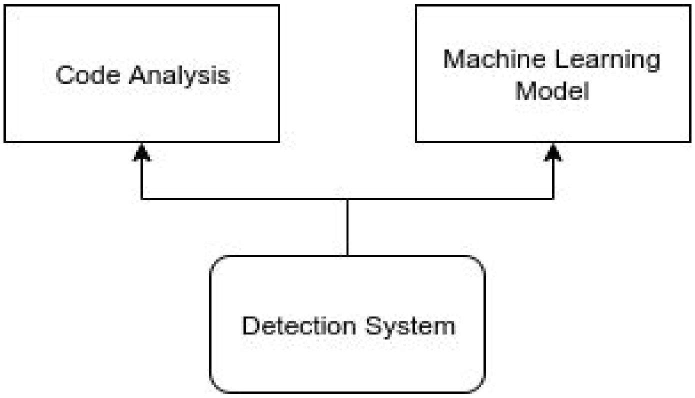
Smart Contracts are implemented in various languages such as Solidity, Vyper, Michelson, Ergo etc. . With Solidity being the most popular, the project primarily aims to implement code analysis and Machine Learning classifications to analyze & classify Smart Contracts written in Solidity on the Ethereum blockchain. Furthermore, a far-fetched goal would be to generalize the bits wherever possible, if permitted by time and resource constraints.

Detection System: There exists some research and proof of concepts for systems which identify code vulnerabilities and classify potential honeypots over the Ethereum blockchain. These miniature implementations (see references) can be refined and extended to develop the proposed solution.

SaaS: The detection system constitutes the internal working of the overall project. The system can be exposed via a Client-Server Model through an API (Application Programming Interface) which can then be consumed via multiple services and architectures such as an application, plugin or extension.

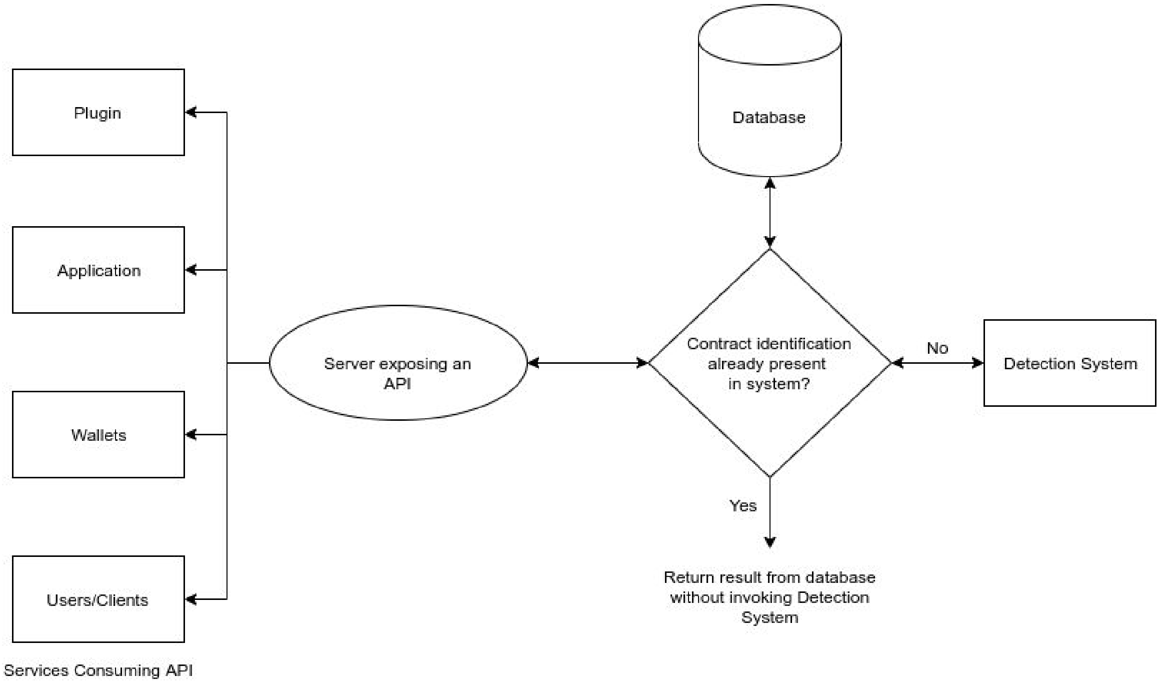
Solution Overview

The project can be viewed as 2 independent components: The Detection System & The SaaS Utility System, which leverages the first. The detection system contains two units, as represented in the following diagram:



Block diagram for the Detection System

The architecture for Service as a Service can be represented in the following block diagram:


Plugin
Application
Wallets
Users/Clients
Services Consuming API
Server exposing an API
Database
Contract identification already present in system?
No
Detection System
Yes
Return result from database without invoking Detection System

Block diagram representing Service Architecture

The above depictions explain the overall architecture of the detection system as well as the service it will provide.

Timeline

This project is divided into 5 phases to accomplish the proposed objectives. An overview of the same is given below:

1. Phase 1

a. Literature Review
b. Studying the core concepts of Smart Contracts & Honeypots.
c. Selecting and defining the problem to be solved.

2. Phase 2

a. Understanding the issues related to Honeypot implementations and their study.
b. Understanding the current solution implementations, finding their limitations and discovering the research gap.
c. Studying types of honeypots and their implementations

3. Phase 3

a. Dataset Preprocessing
b. Implement Code Analysis for Detection System
c. Implement Machine Learning Algorithms for Detection System

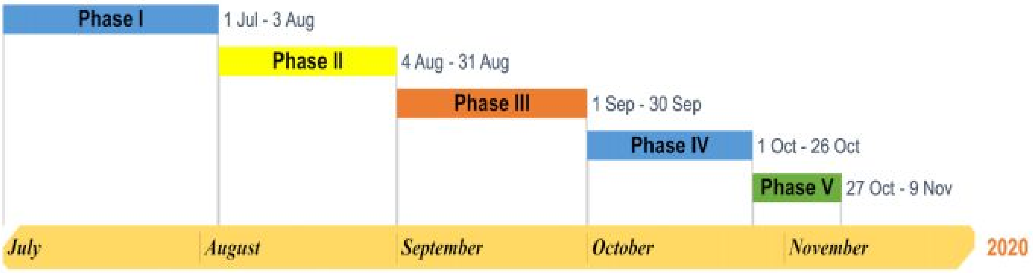
4. Phase 4

a. Design service architecture
b. Develop SaaS solution
c. Deploy overall infrastructure

5. Phase 5

a. Cross Validation
b. Testing
c. Documentation and reports

Phase Wise distribution of timeline with deadline of each phase :



References

[1] Shunfan Zhou, Zhemin Yang, Jie Xiang, Yinzhi Cao†, Min Yang, et al. "An Ever-evolving Game: Evaluation of Real-world Attacks and Defenses in Ethereum Ecosystem" *29th USENIX Security Symposium* (2020).

[2] Nicola Atzei, Massimo Bartoletti, Tiziana Cimoli, Stefano Lande, Roberto Zunino, et al. "SoK: Unraveling bitcoin smart contracts" International Conference on Principles of Security and Trust (2018).

[3] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli, et al. "A Multi-Modal Hierarchical Recurrent Neural Network for Depression Detection." Principles of Security and Trust: 6th International Conference (2017)

[4] Christof Ferreira Torres, Mathis Steichen, Radu State, et al. "The Art of The Scam: Demystifying Honeypots in Ethereum Smart Contracts" *28th USENIX Security Symposium* (2018).

[5] Christof Ferreira Torres, Mathis Steichen, Radu State, et al. "A Data Science Approach for Detecting Honeypots in Ethereum" IEEE International Conference on Blockchain and Cryptocurrency (2020)