PROJECT INCEPTION

CSE 6324

Christoph Csallner

University of Texas at Arlington

Project Title: Conkas 2.0 – Static Analysis Tool

Team - 6

Apoorva Siri Mattewada - 1002026609

Harshith Kannalli Sachidananda0 Murthy -1001949874

Manisha Sohanlal Jain - 1001869817

Vinayak Neemkar - 1002022438

Vignaan Erram - 1002032121

Introduction

A smart contract (self-executable code) is deployed on the blockchain and auto executes due to a triggering condition [4]. These smart contracts are usually written on tools that automate the implementation of agreements between two intermediaries. Their main goal is to deal with extremely valuable financial assets. Due to the sensitive nature of the transactions, even little errors can cause substantial financial losses and permanent damage to consumers. Comprehensive testing is essential for identifying flaws in smart contract programming and minimizing any potential weaknesses. Due to the immutability of smart contracts deployed within the Ethereum Virtual Machine (EVM), runtime testing is necessary to patch vulnerabilities rather than the conventional methods of correcting issues after release. Our tool orients around solving this problem.

GitHub Repository

- Link : Github - Team 6

Project Vision

When working with smart contracts, several static analysis tools have low coverage to work with. Our objective is to increase the number of modules in such tools, to help them find new vulnerabilities and additionally, increase their said functionality.

Conkas is one such tool that is in use, that has a total of 5 known vulnerabilities. Our objective is to expand on the tool's current modules, thereby improving it.

Features and Workflow

Features

There are 5 categories of vulnerabilities that Conkas deals with:

- Reentrancy - A Solidity reentrancy attack continually removes money from a smart contract and transfers it to an unauthorized contract until no more money is available [1].
- Arithmetic - Arithmetic vulnerabilities are integer overflow/underflow and occur when a variable cannot hold a specific value and store another [1].
- Unchecked Low-Level Calls – In Ethereum, there exists several ways for a contract to invoke another contract. These instructions are generally low-level. Typical users use high-level functions to call for transfers between contracts, but this does not propagate exceptions and only returns false when an exception occurs [1].
- Front-Running – Also called as transaction order dependence, it occurs when two transactions invoke the same contract [1].
- Time Manipulation – Contracts may need timely information which can be obtained by a “block.timestamp” variable. This information is controlled by miners, but developers tend to use it and is known to show deviation in time [1].

To address the following vulnerabilities, we would aim to expand the Conkas tool.

- Short Address attack - A short address attack occurs when a contract receives less data than it is anticipated, and Solidity inserts zeros in the spaces left by the missing bytes. The deployed smart contract will not be able to stop this and will consider the extra zeros to be a part of the correct number, leading to significant problems [2].
- Bad Randomness – Randomness is hard to get in Ethereum. This means, it is hard to achieve hard-to-predict values, that are generally either more public than they seem or subject to miner's influence. As the source of randomness are predictable to an extent, malicious users can replicate it and attack a function that uses this unpredictability [2].

Workflow

- Iteration 1 Objective -To generate a vulnerability function for the existing code. Add a functionality to capture additional vulnerabilities in Conkas. Additionally, each team member comes up with vulnerabilities to work on, while deploying contracts.
- Iteration 2 Objective – Further working on adding new vulnerabilities to the same modules and testing it. Additionally, we work on new vulnerabilities on the same module, and implement them on Conkas.
- Iteration 3 Objective - Work on existing vulnerabilities which are not found by the existing modules. Finally, test all the modules and run it on the tool successfully. As there are four vulnerabilities that show up while working on arithmetic category (this prevents Conkas to reach 100% precision), we will aim to implement and test it on several contracts.

Competitors

- Honey Badger – It is a publicly available open-source static analysis tool that was developed in Python. It performs a systematic analysis of honeypot smart contracts [4].
- Maian – It is a publicly available open-source dynamic analysis tool implemented in python that employs symbolic analysis and concrete validation approach. This helps discover a violation of specific properties in smart contracts [4].
- Manticore - It is a publicly available open-source dynamic analysis tool implemented in python by Trell of Bits. User customization is allowed in Manticore for analysis purposes [4].
- Mythril – It is a publicly available open-source analysis tool implemented in Python by CosensSys, a software engineering leader in blockchain space. Ethereum Virtual Machine bytecode is used for analysis of this tool [4].

Smartbugs is a publicly available open-source static analysis framework implemented in python. This framework was used to check vulnerabilities on few static analysis tools. In comparison to other tools, this is where Conkas stands.

No.	Tool	Average Execution Time	Total Execution Time
1	Conkas	0:00:32	1:14:37
2	HoneyBadger	0:01:12	2:49:03
3	Maian	0:03:47	8:52:25
4	Manticore	0:12:53	1 day, 6:15:28
5	Mythril	0:00:58	2:16:21

Table: Average execution time of each tool. [1]

Risks Involved

- Some contracts contain vulnerabilities on their own, however, when these contracts are compiled, those vulnerabilities are gone because the vulnerability is in dead code.
- Updating the tool regularly to counter changes in the contract as smart contracts evolve through time is tough.
- People from different development backgrounds come together to build the tool, this leads to language barriers, and working with such contracts without experience might pose financial risks as well.
- Dealing with false positive vulnerabilities can be tough as they come as a part of any statistical analysis tool.

Customers and Users

- All smart contract developers in the business who need their contracts assessed for a variety of vulnerabilities can use this tool.
- Local entrepreneurs who deal in Ether and manage their own smart contracts assessed for a variety of vulnerabilities can use this tool.
- All professors and students who desire to do study on this type of tool are welcome to utilize it.
- People that want to modify it and add new features can adopt this tool.

References

1. https://www.semanticscholar.org/paper/Conkas%3A-A-Modular-and-Static-Analysis-Tool-for-Veloso/425e474177885f9ac9e57d44e8e2386d13f9c87d

2. https://dasp.co/#item-7

3. https://github.com/nveloso/conkas

4. https://www.researchgate.net/publication/360128366_Ethereum_Smart_Contract_Analysis_Tools_A_Systematic_Review