Inception Report

Symbolic Execution Of Smart Contracts Using Manticore

Advanced Topics in Software Engineering

2228-CSE-6324-001

Github link:https://github.com/PavanKumarChaparla/ADV-TOPICS-IN-SE-CSE-6324

Team 8:

Pavan Kumar Chaparla-1001865675

Keerthi Chittineni-1001967726

Anupama Rani Neerukonda-1002007833

Lokesh Bogam-1001946861

Jai Venkata Sai Sriram Batchu-1001949158

Competitors Of Manticore:

- Studying 47,587 Ethereum Smart Contracts from the Ethereum network discovered 69 susceptible contracts with the use of nine automated analysis tools [1]

Properties/ Tools	**Manticore**	**Osiris**	**Oyente**	**Securify**	**Slither**	**Smartcheck**	**Maian**	**Honey-badger**
Avg Execution Time	24:28 min:sec	34 secs	30 secs	6:37 min:sec	5secs	10secs	1:38 secs	5:16 mins:sec

- Total number of detected vulnerabilities by each tool, including vulnerabilities not tagged in the dataset used for testing:

Category Of Vulnerability/ Tools	**Manticore**	**Osiris**	**Oyente**	**Securify**	**Slither**	**Smartcheck**	**Maian**	**Honey-badger**
Access Control	28	0	0	6	20	3	10	0
Arithmetic	11	62	69	0	0	23	0	0
Denial of Service	0	27	11	0	2	19	0	0
Reentrancy	4	5	5	32	15	7	0	0
Unchecked Low Level Calls	4	0	0	21	13	14	0	0
Unknown Unknowns	25	0	0	0	28	8	2	5

Features [2]:

- **Program Exploration:** It can execute a program with symbolic inputs and explore all possible states.
- **Input Generation:** As a result of concrete inputs, Manticore can automatically produce a particularly program state based on the inputs.
- **Error Discovery:** The Manticore platform detects crashes and other failure cases in binaries and smart contracts.
- **Instrumentation:** Manticore uses event callbacks and instructions hooks give users fine grained control over state exploration.
- **Programmatic Interface:** Through a Python API, Manticore makes its analysis engine accessible to programmers.

Risks:

Risks	Major/Minor	Feasible solution	Current status
Sample smart contract testing	Major	We are working on the implementation of manticore tool	In Progress
Installation of dependences	Minor	As we need Z3, sol C as dependencies of manticore are have to be installed locally	Completed
Installation of manticore	Medium	We are currently facing some issues In OS and virtual environment	In Progress

Customers and Users [3]:

What kind of customers/users use Manticore and how is it useful for them? [4]

Usually, Symbolic Execution for Smart Contracts are used by the Smart Contract Engineers and Smart Contract Developers. However general customers involve those who would like to perform symbolic testing for scripts written in various languages like python, solidity, etc as the Manticore tool supports those languages mentioned. This is an open-source dynamic framework and the end user, or the customer could be the general audience who need the convenience of having the automated analysis for binaries and Ethereum smart contracts.

The Symbolic Execution for Analysis of Smart Contracts using Manticore can be helpful in the following ways:

- Create inputs automatically for a variety of special code paths.
- Trace the inputs that caused the program to crash.
- Keep track of the execution of each instruction.
- Make its analytical engine accessible via the Python API.

The end users can use the **feedback** returned by the manticore symbolic analysis results in finding the unique code paths and inputs that crashes the programs.

References:

- [1] https://arxiv.org/pdf/1910.10601.pdf
- [2] https://github.com/trailofbits/manticore
- [3] https://medium.com/haloblock/introduction-to-manticore-a-symbolic-analysis-tool-for-smart-contract-9de08dae4e1e