



# **Advanced Topics in Software Engineering (CSE 6324)**

# Team 8

Anupama Rani Neerukonda - 1002007833

Pavan Kumar Chaparla - 1001865675

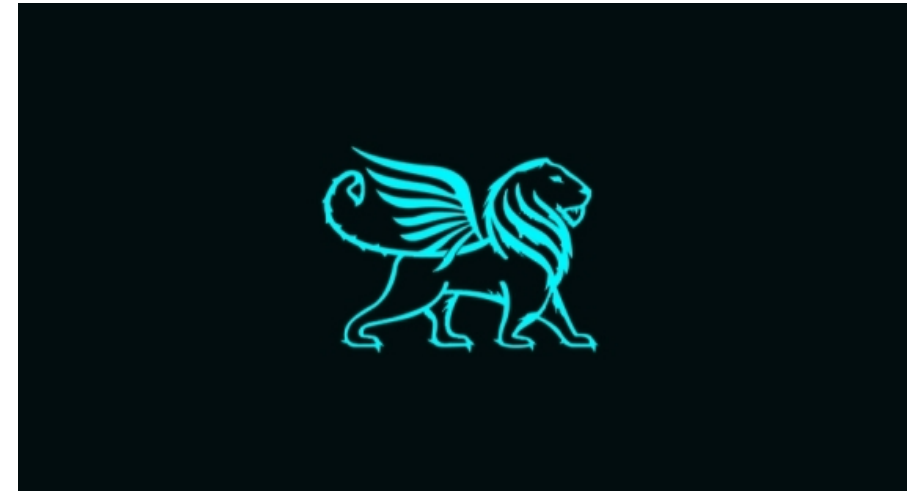
Lokesh Bogam - 1001946861

Jai Venkata Sai Sriram Batchu - 1001949158

Keerthi Chittineni - 1001967726



# Symbolic Analysis for Smart Contracts using Manticore



GitHub: <https://github.com/PavanKumarChaparla/ADV-TOPICS-IN-SE-CSE-6324>

# Contents

- Project Plan
- Use Case Scenarios
- Issue Resolved
- Users and Customers
- References



# Project Plan: Iteration 2

## Introduction: [2]

There are a lot of open issues being reported with the manticore tool. So, our main goal is to improvise the manticore performance on smart contracts by working on these open issues modifying the manticore API's. For the Future Iterations we will be picking an issues related to code coverage and work on them.

## Issues related to Manticore:

- Time Complexity (Current Iteration # 2427)
- Code Coverage
- Compatibility Issues (#2577)

Link: <https://github.com/trailofbits/manticore/issues>

# Project Plan

## Phases:

### 1. Compile and run solidity code

The IDE will allow the user to write the solidity code that user would like to analyze.

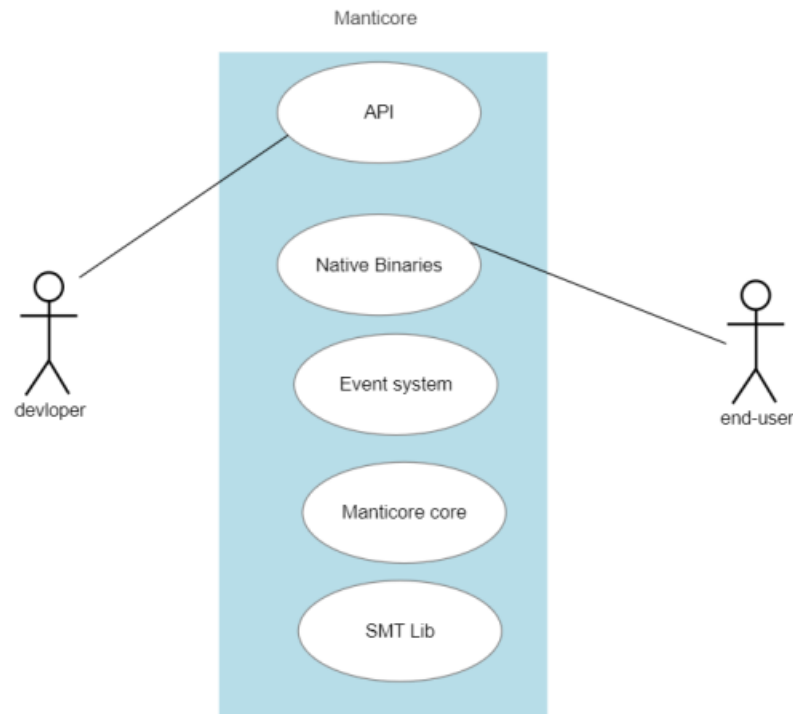
### 2. Analyzing the smart contract with manticore

The user can compile the smart contract code using the manticore API's.

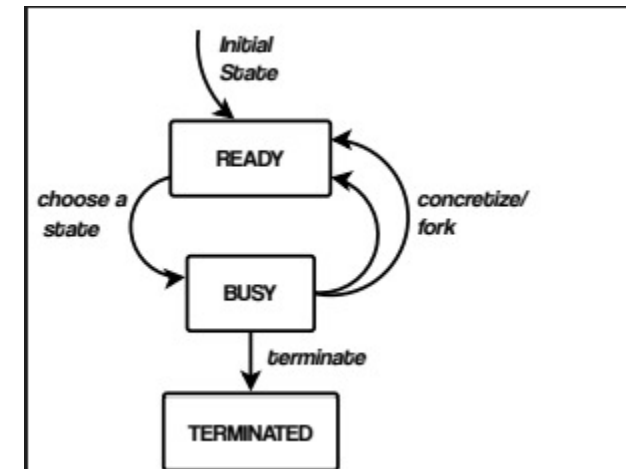
### 3. Generating the results

The tool once finishes the analyzing the code will produce the results into reports and save them in the specified directory.

# Design and Specifications: Pictorial representation :

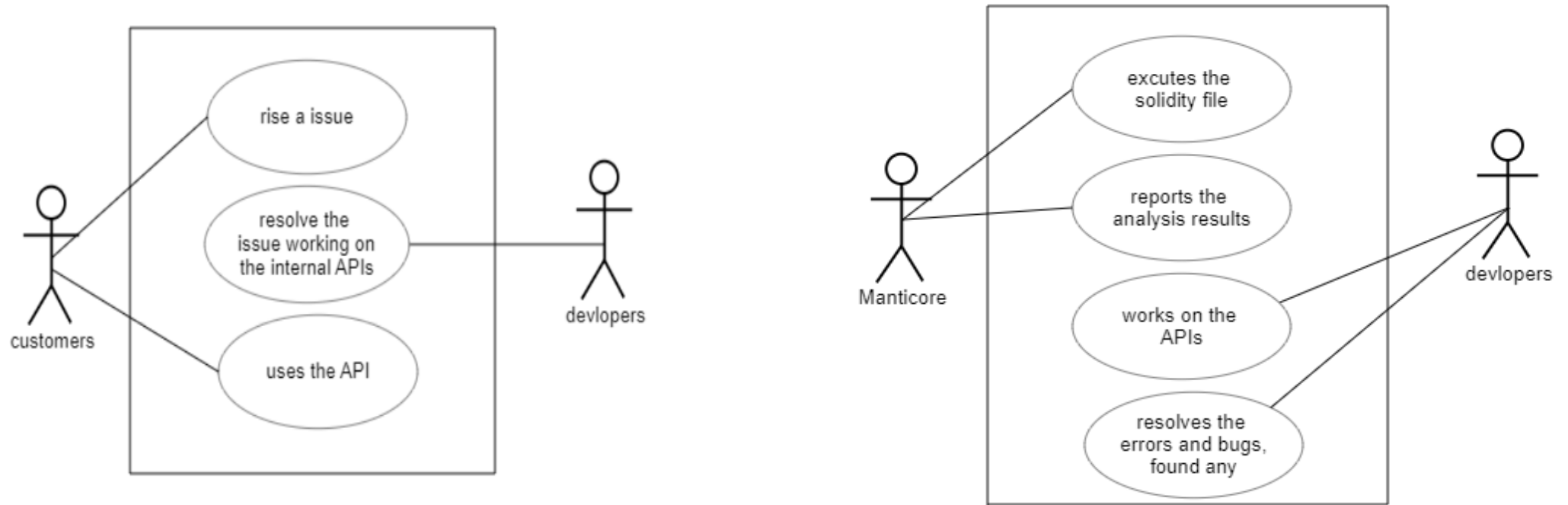


Manticore High Level Architectural View Representation



Workers States Designed Initially in Manticore Core Engine.

# Design and Specifications: Pictorial representation :



Interaction of End Users and Developers with Manticore API



# Code And Tests: Issue Resolved #2427

- When a state goes from busy to killed, the worker handling that state won't take another. In larger targets, it is likely that some workers will stop running, degrading the overall performance of our tool. Also, in the extreme case of a single worker configuration, Manticore will hang for ever (until the user presses ctrl+c or the process is terminated). When the workers are done working with their targets, It was designed in such away that workers will be terminated after the process.

# RISKS

- Modification to the manticore API file in local turned out to be very challenging.
- Picking the proper solidity program for testing the against the time complexity was challenging.
- The latest libraries, which are added to the latest version are not supported by the Z3.
- Manticore Analysis Reports, which are generated by manticore were not user friendly and clear.
- OS centric and testing on various machines needed to be done on certain bugs raised by the developers.
- For the faster analysis, we required a powerful local machine setup.
- As a tool, the Manticore CLI has additional dependencies on the Manticore API's, which will have long-term effects.

# Users and Customers

What kind of customers/users use Manticore and how is it useful for them? [3]

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 Manticore which after the API's being modified performs better with regards to the results and also has the bugs resolved which are identified as the open issues by the developers.



# Resources Used


- Programming Language: Solidity v0.8.17
- Tools: Visual Studio Code v1.69.1
- Repository: GitHub
- Compiler: SolC v0.8.17
- Python v3.10
- Z3 v4.5.0

**Pull request raised for the issue #2577:**

<https://github.com/trailofbits/manticore/pull/2597>

# References

- [1] <https://dl-acm-org.ezproxy.uta.edu/doi/epdf/10.1109/ASE.2019.00133>
- [2] <https://github.com/trailofbits/manticore/issues>
- [3] <https://medium.com/haloblock/introduction-to-manticore-a-symbolic-analysis-tool-for-smart-contract-9de08dae4e1e>
- [4] <https://raw.githubusercontent.com/trailofbits/manticore/master/docs/images/manticore.png>
- [5] <https://www.cam.tv/cryptodimo/blog/what-s-a-smart-contract/PID081737>
- [6] <https://github.com/trailofbits/manticore/issues/2577>
- [7] <https://tse1.mm.bing.net/th?id=OIP.dC4Ky7FPXRshwjCjLHpfgHaEo&pid=Api&rs=1&c=1&qlt=95&w=159&h=99>
- [8] <https://arxiv.org/pdf/1907.03890.pdf>



**THANK  
YOU!**