THE UNIVERSITY OF TEXAS
AT ARLINGTON

Advanced Topics in Software Engineering

CSE 6324 – Section 004

Team 10

INCEPTION

(Written Deliverable)

Vanjari, Vaishali Sunil - 1001956614

Solanki, Siddhrajsinh Pradumansinh - 1001957988

Parimi, Taraka Naga Nikhil - 1001985955

Komireddy, Sannihith Reddy – 1001982437

I. Project Proposal

The idea is to add a compliance checker as a new detector in Slither analysis framework. The Slither is an open-source static analysis framework used to detect vulnerabilities in Solidity programs[1]. It consists of built-in 'printers' which help to identify crucial contract information[1]. It provides a detector API[3] to write custom analyses in Python 3.6v and above. We will be using detector API and Python API[4] provided by Slither to build a customized compliance checker on the top of Slither framework.

The proposed idea can provide real-world relevance in solidity contracts related to solidity contracts and security aspects related to that. There are compliance standards named ERC Conformance[5] available such as ERC1155 for multi-token standards, ERC777 for token standards, and so on. ERC Conformance is used to check the preciseness and correctness of functions and events. The compliance checker will help developers to ensure that these ERC Conformance are compliant with solidity contracts.

II. Features

i) Increased reliability - The main feature is to provide assurance that the compliance standards are compliant with the Solidity contracts which will help to increase the security and reliability of the solidity contracts.
ii) Learning Opportunity – Deploying the compliance checker on the top of Slither framework will provide opportunities to learn more about security aspects for developers.
iii) Scalability – The application can handle large-scale solidity contracts as well as small-scale solidity contracts.
iv) User-friendly UI – It will be helpful to quickly upload the solidity contract and get the generated reports.

III. Planning

i) Set up compliance standards.

There are many ERC Conformance standards available in Slither to ensure standards of functions and events. The solidity contract must be compliant with the ERC conformance standard. Hence, the first step is to set up compliance standards concerning solidity contracts.

ii) Initialize and define the compliance rule.

The next step is to initialize and define the compliance rules in the form of Slither analysis scripts after selecting the compliance standard. These Slither analysis scripts ensure the correctness of tokens, functions, and data types.
For example- defining a function such as totalSupply() can be checked using ERC20 compliance[5].

iii) Load the solidity contract.

The next step is to add a solidity contract in Slither. Either a solidity contract can be provided or the corresponding ABI file or complied bytecode can be uploaded using SolidityContract class.

iv) Analyze the solidity contract.

The next step is to analyze the solidity contract based on the compliance rules defined. The run_script() method[3] can be used to analyze the solidity contract against the compliance standard and rules defined earlier.

v) Generate the compliance report.

The next step is to generate the compliance report once the analysis has been carried out. The generated report consists of a summary of the compliance status concerning the solidity contract. In addition to that, Slither provides output formatting options while generating compliance reports [1]. The report includes a list of compliant and non-compliant functions, events, data types, and other requirements.

vi) Complete the analysis using several iterations.

Once the compliance checker has been deployed, there is a need to check for additional edge cases, and compliance rules. Iterate all over again to check these additional edge rules and to get a precise analysis and coverage of compliance checks.

IV. Delivery Schedule

i) First Iteration
   (a) Set up compliance standards.
   (b) Develop UI to upload the solidity contract.

ii) Second Iteration
   (a) Implementing review changes from the first iteration
   (b) Defining the compliance rules concerning compliance standards.
   (c) Uploading the solidity contract.

iii) Third Iteration
   (a) Implementing review changes from the second iteration.
   (b) Analyzing the uploaded solidity contract concerning compliance rules.
   (c) Generating compliance reports and providing required metrics.

iv) Final Deliverable
   (a) Implementing review changes from the third iteration.
   (b) Implementing re-iteration for additional compliance and edge cases to get precise analysis on solidity contracts.
   (c) Optimizing codes.

V. Competitors

There are several competitors[8] along with Slither which can be used for compliance checking in solidity contracts.

i) Mythrill – It is a security analysis platform for Ethereum Smart Contracts. It detects vulnerabilities such as unchecked calls, bad coding patterns, and many more.

However, Mythrill is unable to discover some business logic vulnerabilities causing financial loss.

ii) ContractFuzzer – It uses fuzzing techniques to find vulnerabilities in Ethereum Smart contracts based on ABI specifications. However, it has the highest false-positive rate which demotivates the developer community.

iii) Remix IDE – It is used to develop and deploy smart contracts in blockchains such as Ethereum. However, it has issues with the compliance checker and does not support linting.

VI. Risks

i) Evolution of Solidity Contract

There is a need to adapt the compliance standard concerning the evolving solidity contract [6].

ii) Accurate and precise compliance rule definition.

To get a precise analysis of a given smart contract, it is important to define a precise compliance rule concerning the compliance standard.

iii) Technical Complexity.

The Compliance checker developed with Slither may not be compatible with other tools considering some technical complexities.

iv) Limitations in Slither.

Slither may be unable to detect the some of known vulnerabilities which may impact the solidity contract[8].

VII. Customers and Users

i) Developers of the Solidity community

Deploying a Compliance checker with Slither will give real-world relevance to the developers of the solidity community.

ii) Developers of Slither community

Implementing the compliance checker with Slither needs a better understanding of all aspects of Slither. This is a challenging but useful project to carry out different scenarios in Slither for developers of Slither community.

iii) Ethereum Researchers

VIII. Collaborative Work & Constant Feedback

i) Shovon Niverd Pereira

ii) Bhupesh Avinash Patil

IX. References

1. Slither, A Solidity source analyzer
2. Detector Documentation - https://github.com/crytic/slither/wiki/Detector-Documentation
3. Adding a new detector - https://github.com/crytic/slither/wiki/Adding-a-new-detector
4. Python API - https://github.com/crytic/slither/wiki/Python-API
5. ERC Conformance - https://github.com/crytic/slither/wiki/ERC-Conformance
6. Slither: A Static Analysis Framework for Smart Contracts
7. GitHub repository
8. Top 10 Solidity Smart Contract Audit Tools