THE UNIVERSITY OF TEXAS
AT ARLINGTON

Advanced Topics in Software Engineering

CSE 6324 – Section 001

Team 9

INCEPTION

(Written Deliverable)

Abhishek Wadhwani – 10020352719
Mounika Kottapalli– 1002085510
Nitin Raj Thumma– 1002080555
Sai Raghu Rami Reddy Dontireddy – 1002014523

I. Project Proposal:

The main framework that we decided to use in the project is Slither. [1] Slither is an open-source analysis framework developed for smart contracts which is written in solidity. It is used for identifying vulnerabilities, security, weaknesses, and errors in the code. A few key features of this framework are Automated analysis, customizable rules, ease in integration and ease of reporting.

Issue 1: False Positives in Unused-Return Detection [4]

Problem Statement: Slither occasionally emits false positives for the "unused return" issue, specifically when taking only some multiple values from a call to another contract.

The idea is to implement Foundry Expect emit and false-Positive identifier in the naming convention feature in slither. This feature is used to check the events emitted by a smart contract at its execution. In this we modify the analysis tool since this tool works with an analysis tool i.e., **slither** here in this case. We do need to analyze the solidity abstract tree here to locate event emissions statements. There are things to be considered such as rules which are to be followed such as parameter values and conditions that must be met for event emissions to be considered correct.

Issue 2: Error Occurrence on Certain Contracts [5]

Problem Statement: Slither encounters errors when analyzing certain contracts online, resulting in failed analysis attempts.

For this to do we will produce a logic that locates the relevant part of the slither frameworks responsible for the naming conventions detector and checks whether the events emitted in the code match expectations defined by the developer, this does involve comparing event names, parameters, checking the given names to some of the predefined naming conventions and conditions. So, when the tool finds the differences between the actual and expected emissions. Those should be reported to the developer, also picking up some good test cases to test the proposed project would be an ideal approach.

II. Features:

1. Event Emission verification: This feature allows the developer to specify the expected events that are usually considered as the emitted event under any specific conditions.
2. Customizable prefixes: Any user using it can customize the prefix they would like to use to filter out the false positive outcome.
3. Event Trace Analysis: This feature would analyze the control flow and the change of state in any contract to ensure the expected events are emitted in order with the expected parameters.
4. Various configuration options: This provides diverse options in the command line interface or configuration files.
5. Integration with testing frameworks: This is another interesting feature that makes the work easier as we can integrate into various testing frameworks like Truffle or Hardhat.

III. Planning

1. We will start by looking into the tool's issues and finding ways to solve them, like fixing false alarms in Slither's naming-convention detector and creating a new one for expectEmit function arguments.
2. Then, we will set up the development environment for the Slither detector and make it ignore certain naming prefixes.
3. After that, we will create the detector for expectEmit function arguments and test it on example contracts to make sure it works well.
4. We will also check event signatures in Solidity contracts to validate expectEmit arguments and provide clear warnings if needed.
5. Finally, we will put the expectEmit argument checker into our automated testing process to ensure it works reliably across different contracts and situations.

IV. Delivery Schedule

1. First Iteration
    1.1. Implementing review changes and recommendations from the inception.
    1.2. Code Study and Analysis, detectors development, and Initial Testing
    1.3. Implement the detector for the Foundry expectEmit feature.
    1.4. Study the naming-convention detector codebase in Slither.
2. Second Iteration
    2.1. Implementing review changes from the first iteration
    2.2. Integration and Compatibility Testing
    2.3. Ensure seamless integration with Foundry and compatibility across versions.
3. Final Iteration
    3.1. Implementing review changes from the second iteration.
    3.2. Final Testing and additional compliance for edge cases (if any) to get precise analysis.
    3.3. Comprehensive testing and user documentation.
    3.4. Optimizing code and providing user documentation.

V. Competitors:

1. Mythril: It is a security platform that does smart contract auditing for solidity developed by ConsenSys
2. Echidna: It uses Fuzzing techniques to find vulnerabilities which have the highest false-positive rates
3. Remix IDE: It is used to develop Smart Contracts which has issues with compliance checker.

VI. Risks

1. Technical Complexity: The detectors that ensure the solutions to the issues are compatible and interoperable with your existing systems and processes may not be compatible with those of your suppliers and partners.
2. Security Threats: Blockchain and Smart Contracts are not immune to cyber-attacks and malicious actors. They need to be frequently monitored and security measures such as encryption, authentication, authorization, and backup should be implemented.
3. Delayed Transactions: Congestion in a blockchain network can result in delayed transactions and eventually push transactional costs above the cost incurred in traditional contracts. [6]
4. Slither Limitations: Slither may be unable to detect some of the known vulnerabilities that may impact solidity contracts. [1]

VII. Users

1. Developers of Slither Community
2. Academic Researchers
3. Developers of Solidity Community
4. Security Experts
5. Ethereum Researchers

VIII. References

1. Slither: https://github.com/crytic/slither
2. Detector Documentation - https://github.com/crytic/slither/wiki/Detector-Documentation
3. Adding new detector - https://github.com/crytic/slither/wiki/Adding-a-new-detector
4. Issue1: https://github.com/crytic/slither/issues/2036
5. Issue2: https://github.com/crytic/slither/issues/2083
6. What are smart contracts? https://whatfix.com/blog/smart-contracts-changing-legal-landscape/

GitHub Account: https://github.com/abhismoothie

GitHub Repository: https://github.com/Abhismoothie/Slither-Enhancement-project-team-9-CSE6324-001