THE UNIVERSITY OF TEXAS
AT ARLINGTON

Advanced Topics in Software Engineering
CSE-6324-004
Iteration 1 (Written Deliverable)

Sharma, Arnav -1002070507
Ashraf, Abdul Rafay - 1001918598
Kamble, Nikhil - 1001837272
Mohammed, Ishraq - 1002082521

I. **Project Plan**

1. Delivery & Schedule

#	Task/Milestone Description	Anticipated Start Date	Anticipated End Date	Status
1	a. Installing Slither on our local machine. b. Analyzing smart contracts to gain practical experience with Slither and gain a deeper knowledge on the working of Slither. c. Understanding Existing algorithms used for analysis of solidity contracts. d. Developing pseudo-code for CSV conversion.	02/14/2023	03/05/2023	In-progress
2	a. Testing the code to identify new vulnerabilities if any. b. Developing a new method to add Dead code detector for function pointers. c. Implementing review changes from Iteration 1.	03/07/2023	04/02/2023	To be completed
3	a. Testing Dead code detector to identify new vulnerabilities if any. a. Rigorous testing. b. Code optimization. c. Implementing review changes from Iteration 2.	04/04/2023	04/24/2023	To be completed

2. Risk factors and mitigation solution

#	Risk	Description	Mitigation plan	Risk exposure
1	Lack of Experience with solidity	The lack of knowledge the team members have with the solidity programming language or Ethereum smart contracts could affect how the features are implemented.	Spending few/Initial weeks in understanding the flow and gain deeper knowledge on working of program analysis tools	Risk impact: 4 weeks Probability that risk will materialize: 94% Risk exposure: 4 weeks (approx.)
2	Lack of Data validation	Inaccurate analysis or conclusions may result from flaws or omissions in CSV files.[1]	User will have an option to include only the data which is required for their analysis	Risk impact: 4 weeks Probability that risk will materialize: 94% Risk exposure: 4 weeks (approx.)
3	Data Accuracy	Slither Solidity output to a CSV file conversion could result in errors or data loss, particularly if improper conversion procedures are used. This can result in inaccurate analysis or wrong interpretation of the findings.[2]	Checking the CSV file carefully for mistakes and anomalies and comparing the data to the original Slither output to guarantee data accuracy. To make sure the data is formatted correctly.	Risk impact: 4 weeks Probability that risk will materialize: 94% Risk exposure: 4 weeks (approx.)

4	False Positive on Function pointers	The detector ignores the dynamic behavior of function pointers and simply examines the control flow of the contract.[4]	Our approach is to modify the dead-code detector to incorporate function pointers' dynamic behavior into account.	Risk impact: 6 weeks Probability that risk will materialize: 20% Risk exposure: 6 weeks (approx.)
5	Analyzer method	By reducing false positives on function pointers, we might eliminate all the false positives [4]	Retrieving the list of function calls from function pointers after initializing the collection of function pointers. The set of live code is then determined using the set of function calls.	Risk impact: 2-week Probability that risk will materialize: 94% Risk exposure: 2 weeks (approx.)

3. Competitors

a. Overview:

- Mythril: It is a publicly accessible open-source analytical tool that was created in 2017 by ConsenSys, a pioneer in blockchain software engineering. This tool accepts byte code from the Ethereum Virtual Machine as input for analysis. Mythril works for various blockchain platforms in addition to analyzing smart contracts built on the Ethereum network. Mythril analyzes smart contracts using three different methods: symbolic execution, SMT solution, and taint analysis.[5]

- Solhint: It is a Java-based static analysis command-line tool that was created in 2017 and is freely accessible to the public. For effective parsing and validation efficiency, Solhint uses an antlr4-based version of the Solidity parser. The tool offers a variety of setup choices, including managing configuration rules at the code level, utilizing a predetermined set of rules, and customizing the default rule set. The three main instructions used by Solent 1) **/*. Sol: This command sends a list of file patterns for it to examine. 2) Stdin: It offers source code for validating standard input. 3. init config: This function generates a default configuration file that can be altered as necessary. [5]

b. What makes our tool unique?

The output of Slither Solidity is in JSON format by default, and includes information such as:

- General information about the analyzed contract
- A list of all the issues detected by the tool
- Detailed information about each detected issue
- Other metrics related to the code complexity and structure.

- Ease of analysis: Several tools, including spreadsheet programs like Excel and Google Sheets, can read CSV files with ease. It is simpler to carry out additional analysis, sort, filter, and data visualization when Slither output is converted to CSV.
- Standardization: For storing and transferring tabular data, CSV is a popular file format. Slither output can be converted to CSV, making it simpler to share and work on the analysis with people who might not have access to the original Slither output or who might not be familiar with the software.
- Customization: Instead of having to comb through the full JSON file, Slither output to CSV enables you to pick and choose only the data that is pertinent to your research. This makes it simpler to concentrate on weaknesses or interesting concerns.
- Integration: Tools for automated analysis and reporting can simply incorporate CSV data. It is simpler to integrate Slither output with other tools and systems when it is converted to CSV.
- reducing time and effort while reviewing and maintaining manual code, the detector is less likely to report code as dead when it is truly live.
- Smart contracts frequently use function pointers, which make the contracts more adaptable and modular. However, because the destination of a function call can only be known at runtime, they also make it more challenging to examine the control flow of a contract. The analysis becomes more accurate and can generate more helpful results by updating the dead-code detector to handle function pointers more properly.

II. Specification and Design

1. Inputs and Outputs:

   We load a contract into slither analysis tool for static analysis and obtain an output as a “.json” file. We intend to convert this file format into a “.csv” format which enables us to pick and choose only the data that is pertinent to user's research.

2. Data Structures:

Depending on the precise analysis being done, the data structure for Slither Solidity output as CSV will vary. The CSV output, however, will typically include a list of variables, values, and other characteristics that are pertinent to the analysis.

The output may include the following fields for each contract:

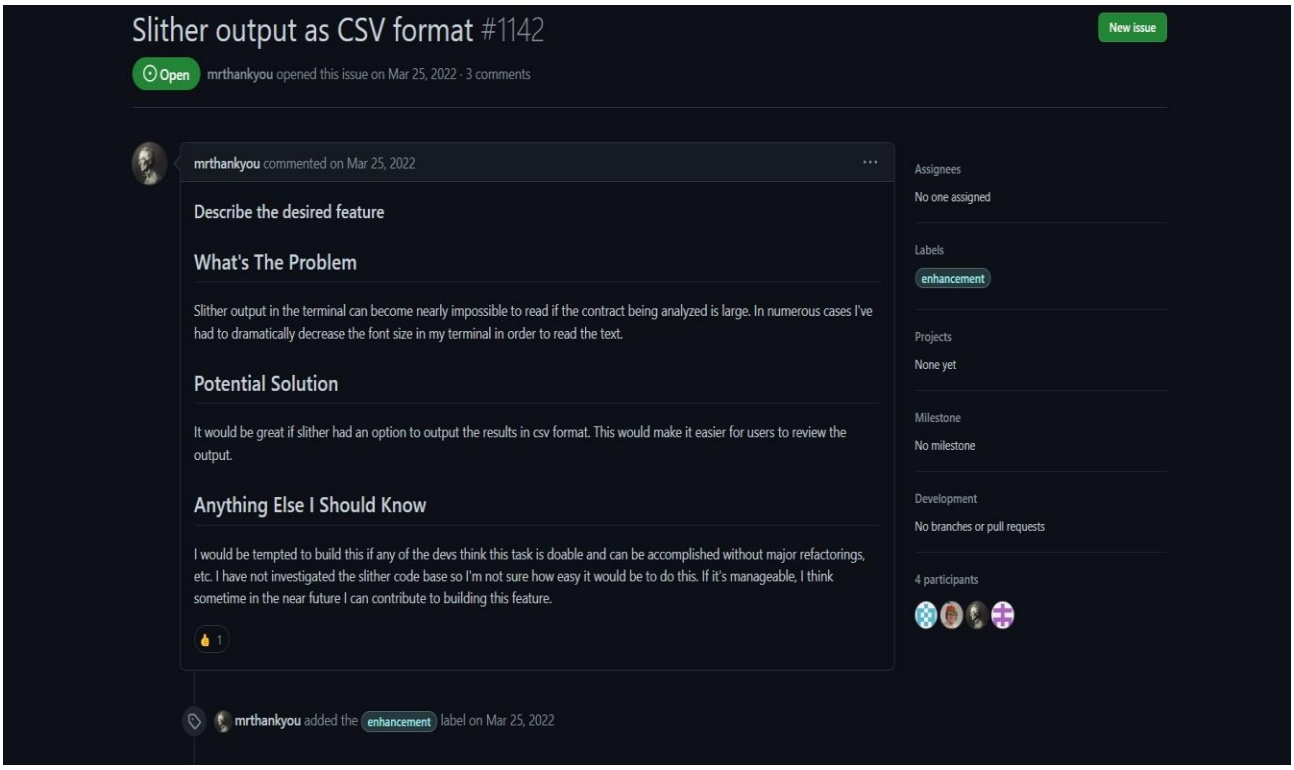
a. Contract Name: The title of the agreement under investigation.
b. Contract Address: The blockchain address of the contract.
c. Name of the Function: The name of the function being examined.
d. Function Signature: The function signature, which also specifies the return type and the kinds of parameters.
e. And many more.

3. Use Cases

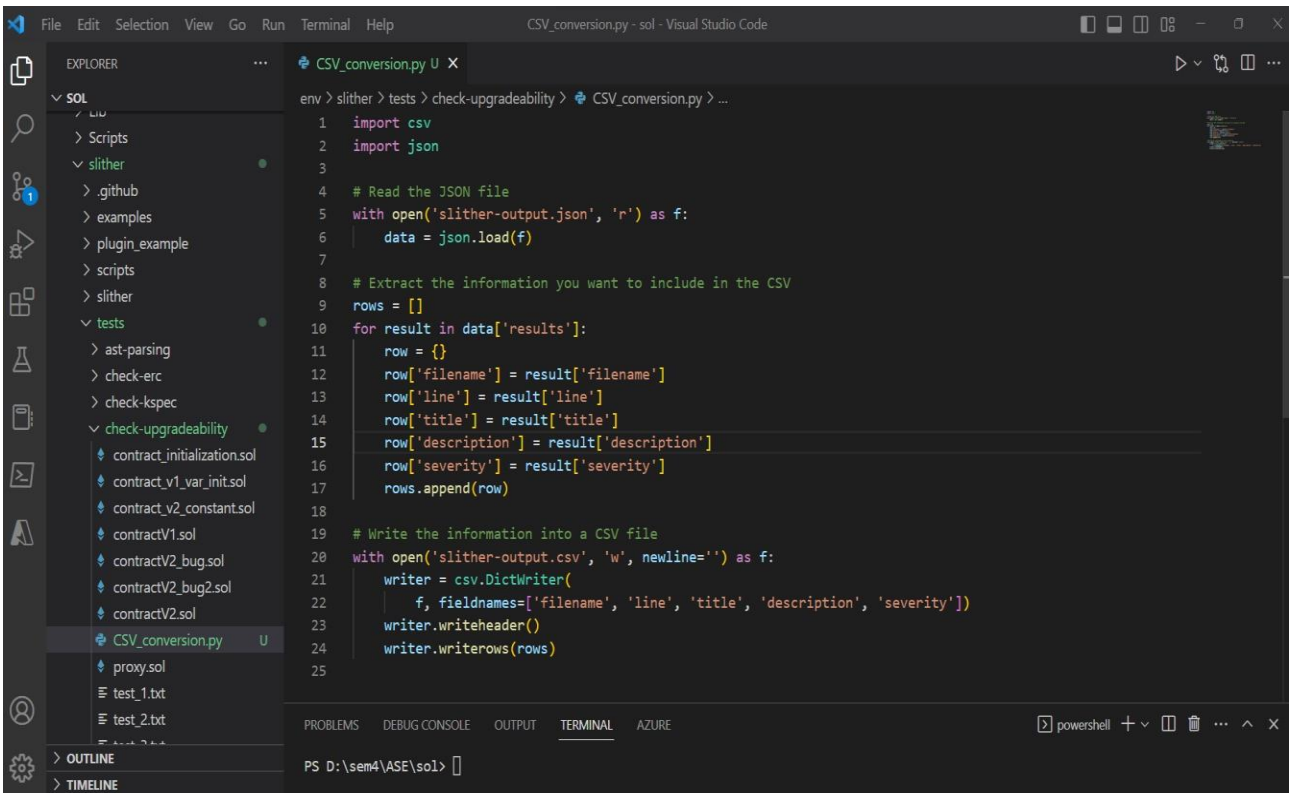
Use Case 1	Precise Data Analysis
Actor	User/Customer
Basic flow	Users can use various tools and software to perform further analysis and gain deeper insights into the code being analyzed.
Use Case 2	Report Generation
Actor	User/Customer
Basic flow	Converting the output to CSV makes it easier to generate reports and visualizations. The data can be easily imported into various reporting tools and software, such as Excel, Google Sheets, or Tableau

4. Screenshots:

- The issue:



- The Pseudo-Code:



III. Code and tests

1. Test Cases

#	Test Scenario	Expected Output
1	Verify output format	User should be displayed with correct format and all necessary data fields
2	Verify Data Accuracy	Users should get data which is accurate and match the data generated by slither.

2. Getting Started with:

- Install python 3.8+
- Install solc
- Pip3 install slither-analyzer

3. GitHub Link: https://github.com/Nikhil19-hub/Solify

IV. Customers and users

1. Users:

- Developer community of Solidity and Ethereum smart contracts – Provide developers an option to view the output in a ".csv" file wherein it is easier to carry out additional analysis, sort, filter, and data visualization. It can also help with picking and choosing only the data that is relevant to the user.

2. Feedback:

#	**Customer/User**	**Feedback**	**Suggestions/Comments**
1	Reena Ughade – Smart Contract Enthusiast	This would make it easier for users to review the output.	having the option to get a csv (or txt) output directly from the CLI would be a very good idea
2	Sagar Jadhav – solidity Beginner	Good concept	I think sometime soon I can contribute to building this feature.
3	Girish Pagare – Solidity developer	Slight, But good addition overall. Great job...!!	Converting Slither output to CSV format will provide a more accessible and flexible way to analyze and work with the results of Slither analysis.

VI. References

1. Smart Contracts in Blockchain Technology: A Critical Review
2. Step by Step Towards Creating a Safe Smart Contract
3. Finding The Greedy, Prodigal, and Suicidal Contracts at Scale
4. Dead code Elimination based pointer analysis
5. Ethereum Smart Contract Analysis Tools: A Systematic Review