THE UNIVERSITY OF TEXAS AT ARLINGTON

Advanced Topics in Software Engineering
CSE-6324-004
Final Iteration(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	Completed
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**	**Completed**
3	a. Testing Dead code detector to identify new vulnerabilities if any. b. Rigorous testing. c. Code optimization. d. Implementing review changes from Iteration 2.	04/04/2023	04/24/2023	In-progress

2. Risk factors and mitigation solution

Risks mitigated in Iteration 2:

To make data analysis and manipulation simpler, the output of Slither Solidity, a static analysis tool used to find security holes in Solidity smart contracts, needed to be converted into a CSV format. Users will be able to design unique reports and visualizations, as well as carry out additional data analysis or manipulation, thanks to the CSV file. However, there are potential drawbacks to watching Slither Solidity output as CSV, including conversion errors or discrepancies as well as possible security risks. In order to convert the Slither Solidity output to CSV file while minimizing these risks and guaranteeing the accuracy and security of the output data, a solution was needed.

#	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. It provides a CSV output format that can be used for further analysis and manipulation of data. The tool is designed to detect various security vulnerabilities, including reentrancy, gas-related vulnerabilities, and integer overflows/underflows. Mythril can also detect issues with the contract's code, such as dead code or unreachable code. [5]

- Securify: Securify is a security analysis tool for smart contracts that provides a JSON output format. The JSON output can be easily converted to CSV format using a third-party tool. Securify uses a combination of static analysis and dynamic analysis to identify potential security vulnerabilities in the contract. It can detect issues such as reentrancy, gas-related vulnerabilities, and other common vulnerabilities. [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.
- A dead code detector locates lines of code in a program that are never run when the program is being performed. Dead code can exist for a number of reasons, including when the code is no longer required following a modification or when a conditional statement is always evaluated as true or false.
- Dead code can make it more difficult to read, maintain, and comprehend code. Additionally, it might lengthen the produced code, which might raise the price of installing and executing the program. In order to ensure code efficiency and maintainability, finding and eliminating dead code is a crucial stage in the software development process.
- In the context of smart contracts, dead code detection is particularly important since any unused code left in a contract can potentially create security vulnerabilities and expose the contract to attacks. A dead code detector can help identify such vulnerabilities and aid in the auditing and testing of smart contracts.

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.

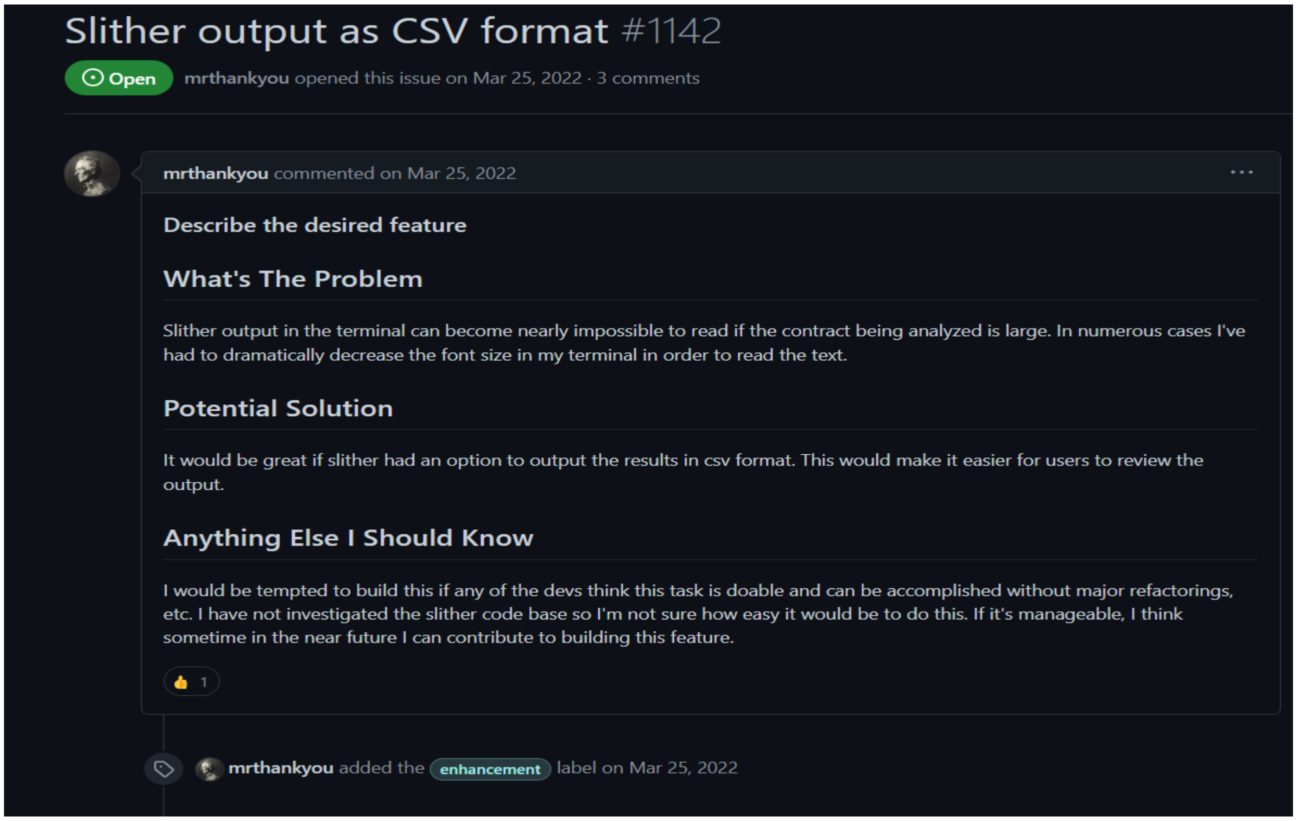
- Checking for the state variable that are being called in the functions
- If found iterate over the state variables that are used in function pointers if there are state variables we are appending it to the function pointers array

3. Use Cases:

Use Case 1	Improving Accuracy of Dead code detection in programs that uses function pointers
Actor	User/Customer
Basic flow	Adding support for detecting false positives on function pointer will provide users with more accuracy for analyzing programs that use function pointers.
Use Case 2	Identify unused or unnecessary functions
Actor	User/Customer
Basic flow	Helping user in reducing the size of the contract and improve its efficiency
Use Case 3	Improved program performance and resource consumption
Actor	User/Customer
Basic flow	Removing dead code can help user to improve performance and reduce its resource consumption

4. Screenshots:

- Issue #1:



Slither output as CSV format #1142

Open mrthankyou opened this issue on Mar 25, 2022 · 3 comments

mrthankyou commented on Mar 25, 2022

Describe the desired feature

What's The Problem

Slither output in the terminal can become nearly impossible to read if the contract being analyzed is large. In numerous cases I've had to dramatically decrease the font size in my terminal in order to read the text.

Potential Solution

It would be great if slither had an option to output the results in csv format. This would make it easier for users to review the output.

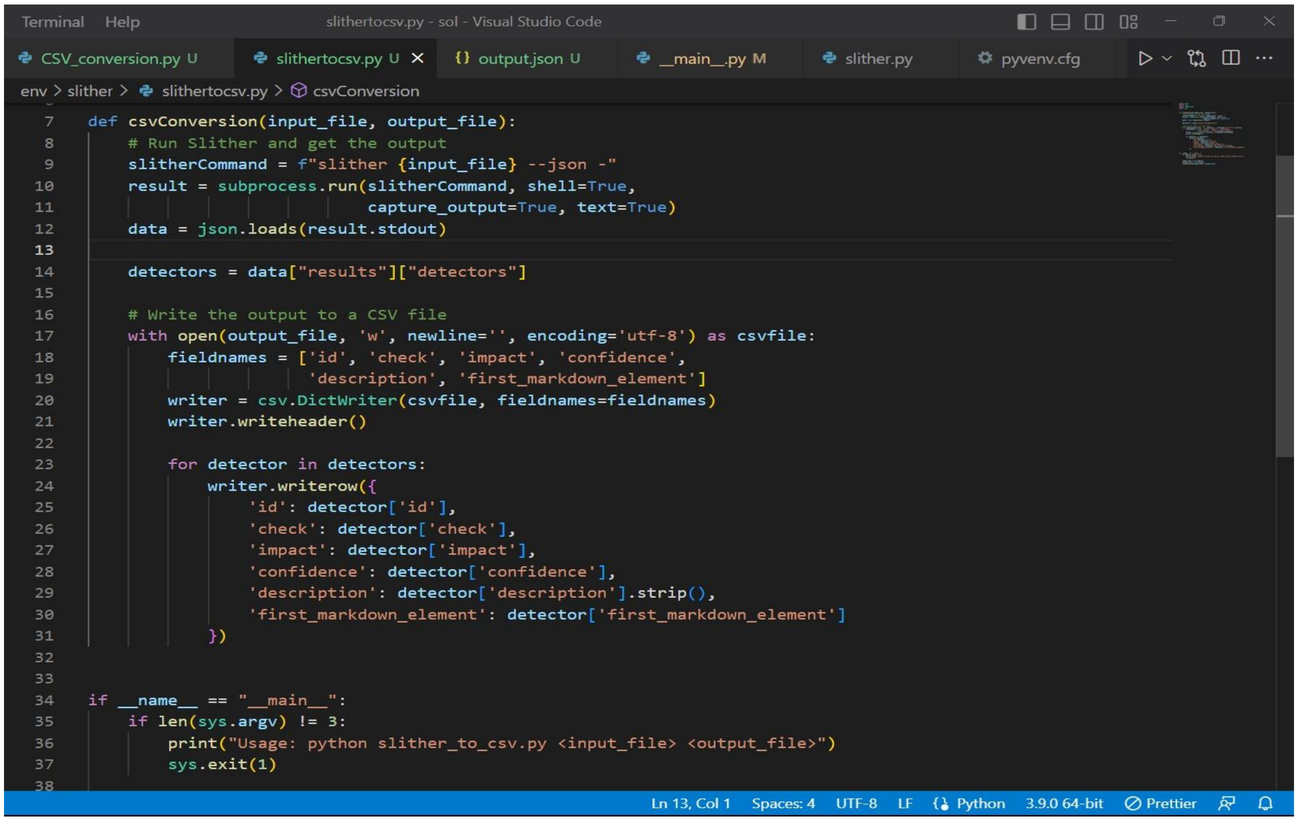
Anything Else I Should Know

I would be tempted to build this if any of the devs think this task is doable and can be accomplished without major refactorings, etc. I have not investigated the slither code base so I'm not sure how easy it would be to do this. If it's manageable, I think sometime in the near future I can contribute to building this feature.

👍 1

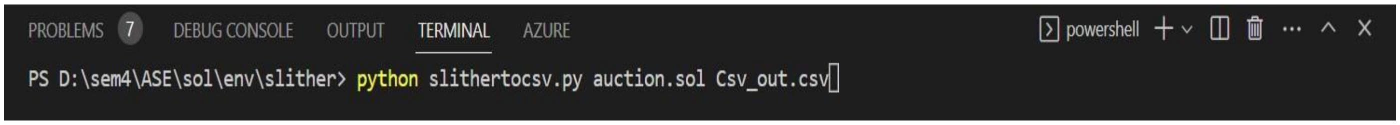
mrthankyou added the enhancement label on Mar 25, 2022

I. Pseudo Code:


Terminal Help
slithertocsv.py - sol - Visual Studio Code
CSV_conversion.py U
slithertocsv.py U
output.json U
__main__.py M
slither.py
pyvenv.cfg
env > slither > slithertocsv.py > csvConversion
Ln 13, Col 1 Spaces: 4 UTF-8 LF Python 3.9.0 64-bit Prettier

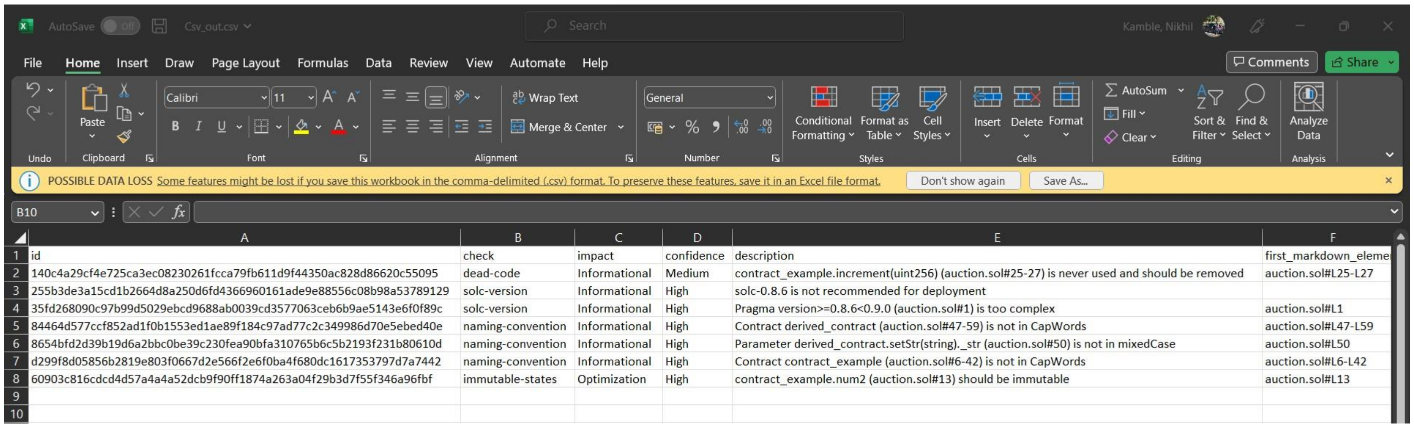
```python
def csvConversion(input_file, output_file):
    # Run Slither and get the output
    slitherCommand = f"slither {input_file} --json -"
    result = subprocess.run(slitherCommand, shell=True,
                            capture_output=True, text=True)
    data = json.loads(result.stdout)

    detectors = data["results"]["detectors"]

    # Write the output to a CSV file
    with open(output_file, 'w', newline='', encoding='utf-8') as csvfile:
        fieldnames = ['id', 'check', 'impact', 'confidence',
                      'description', 'first_markdown_element']
        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
        writer.writeheader()

        for detector in detectors:
            writer.writerow({
                'id': detector['id'],
                'check': detector['check'],
                'impact': detector['impact'],
                'confidence': detector['confidence'],
                'description': detector['description'].strip(),
                'first_markdown_element': detector['first_markdown_element']
            })


if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Usage: python slither_to_csv.py <input_file> <output_file>")
        sys.exit(1)
```

II. CLI Command:



```
PS D:\sem4\ASE\sol\env\slither> python slithertocsv.py auction.sol Csv_out.csv
```

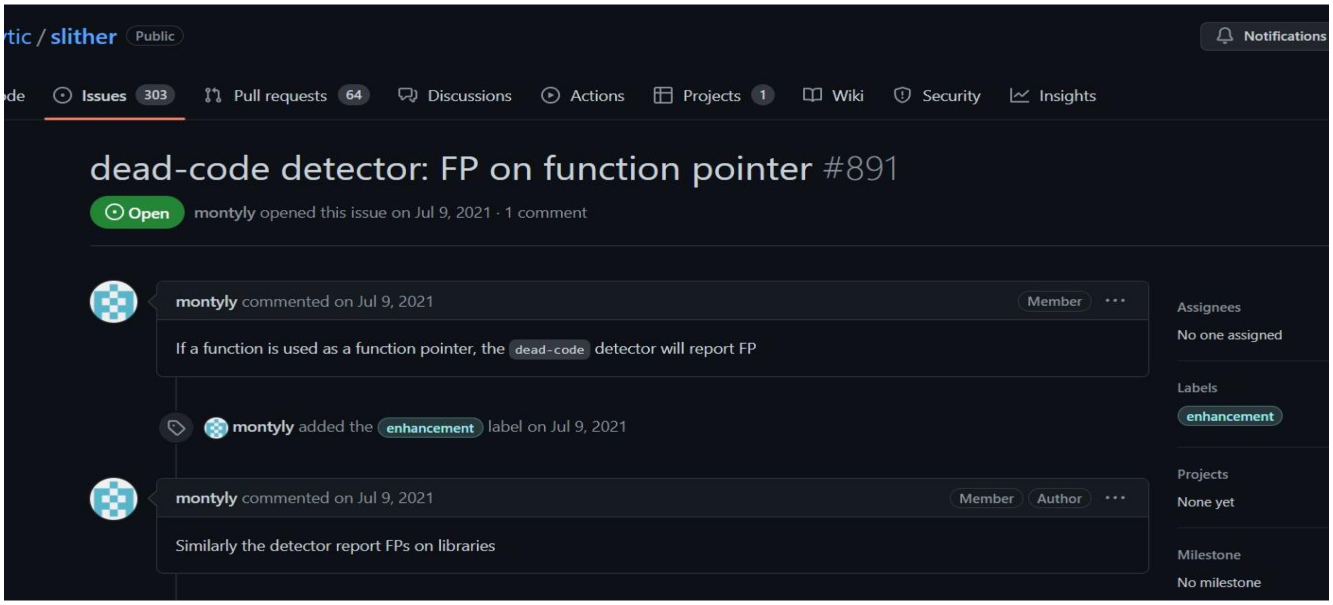
III. Output:



POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.

	A	B	C	D	E	F
1	id	check	impact	confidence	description	first_markdown_eleme
2	140c4a29cf4e725ca3ec08230261fcca79fb611d9f44350ac828d86620c55095	dead-code	Informational	Medium	contract_example.increment(uint256) (auction.sol#25-27) is never used and should be removed	auction.sol#L25-L27
3	255b3de3a15cd1b2664d8a250d6fd4366960161ade9e88556c08b98a53789129	solc-version	Informational	High	solc-0.8.6 is not recommended for deployment	
4	35fd268090c97b99d5029ebcd9688ab0039cd3577063ceb6b9ae5143e6f0f89c	solc-version	Informational	High	Pragma version>=0.8.6<0.9.0 (auction.sol#1) is too complex	auction.sol#L1
5	84464d577ccf852ad1f0b1553ed1ae89f184c97ad77c2c349986d70e5ebed40e	naming-convention	Informational	High	Contract derived_contract (auction.sol#47-59) is not in CapWords	auction.sol#L47-L59
6	8654bfd2d39b19d6a2bbc0be39c230fea90bfa310765b6c5b2193f231b80610d	naming-convention	Informational	High	Parameter derived_contract.setStr(string)._str (auction.sol#50) is not in mixedCase	auction.sol#L50
7	d299f8d05856b2819e803f0667d2e566f2e6f0ba4f680dc1617353797d7a7442	naming-convention	Informational	High	Contract contract_example (auction.sol#6-42) is not in CapWords	auction.sol#L6-L42
8	60903c816cdcd4d57a4a4a52dcb9f90ff1874a263a04f29b3d7f55f346a96fbf	immutable-states	Optimization	High	contract_example.num2 (auction.sol#13) should be immutable	auction.sol#L13
9						
10						

- Issue #2:


tic / slither Public
Notifications
de Issues 303 Pull requests 64 Discussions Actions Projects 1 Wiki Security Insights

dead-code detector: FP on function pointer #891

Open montyly opened this issue on Jul 9, 2021 · 1 comment

montyly commented on Jul 9, 2021 Member

If a function is used as a function pointer, the `dead-code` detector will report FP

montyly added the enhancement label on Jul 9, 2021

montyly commented on Jul 9, 2021 Member Author

Similarly the detector report FPs on libraries

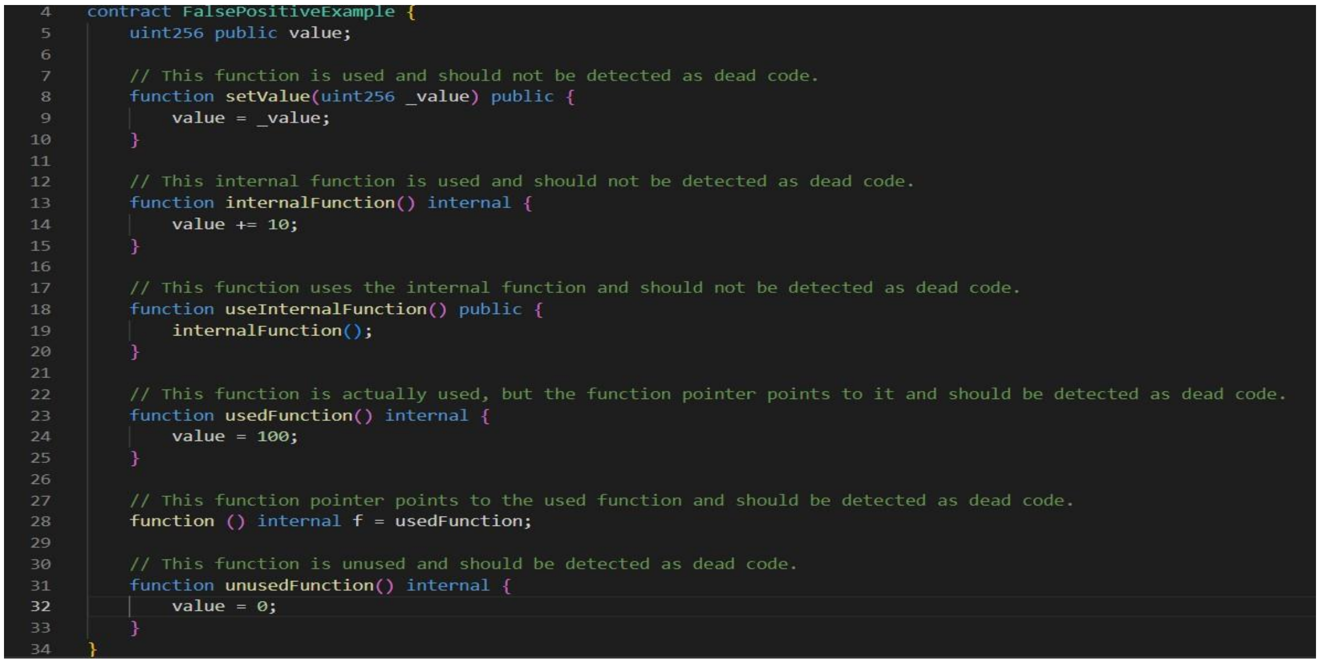
Assignees
No one assigned

Labels
enhancement

Projects
None yet

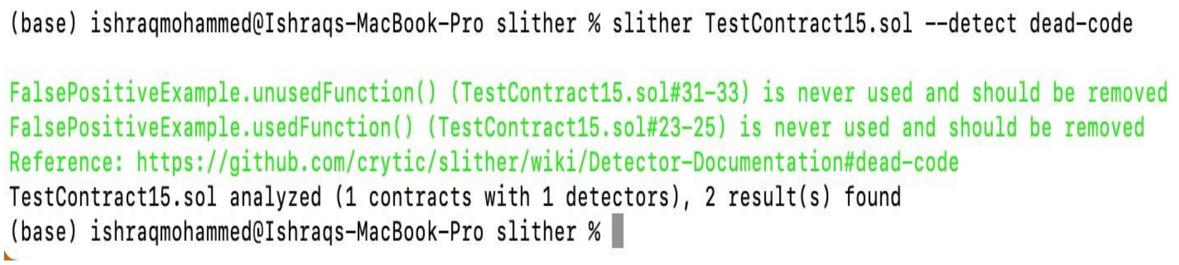
Milestone
No milestone

- Sample Contract:



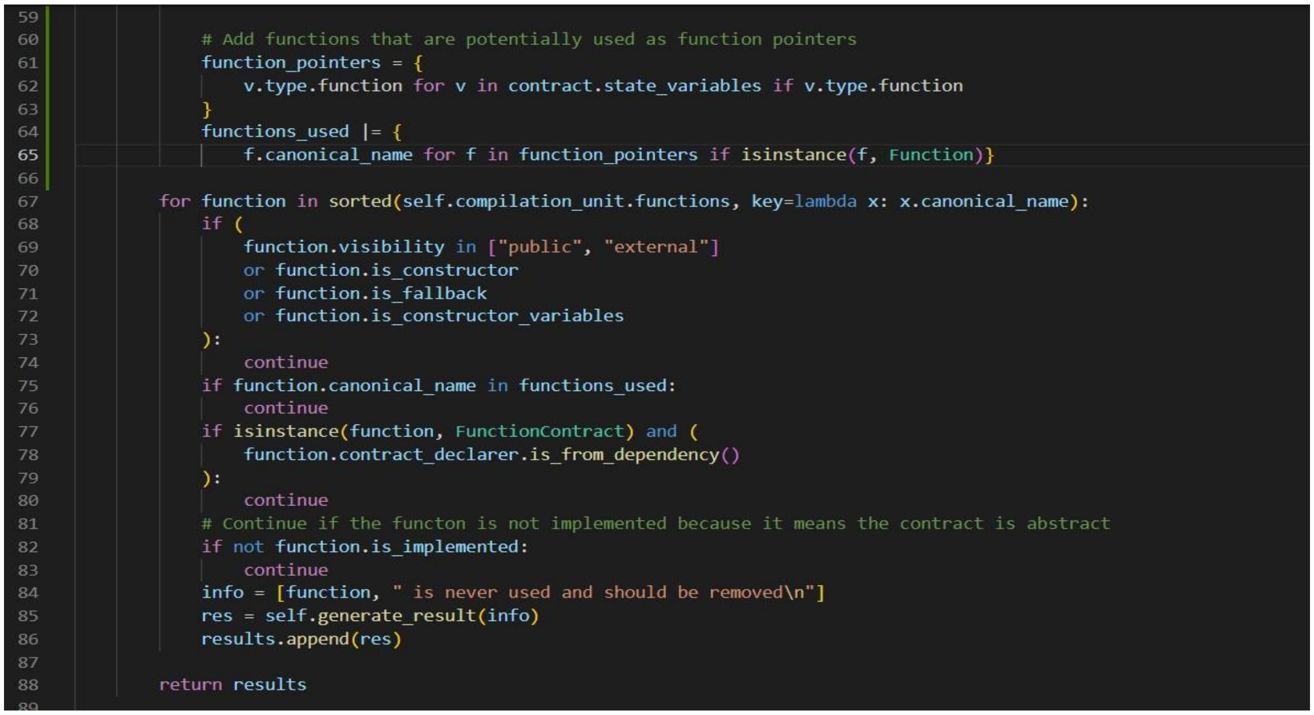
```
contract FalsePositiveExample {
    uint256 public value;

    // This function is used and should not be detected as dead code.
    function setValue(uint256 _value) public {
        value = _value;
    }

    // This internal function is used and should not be detected as dead code.
    function internalFunction() internal {
        value += 10;
    }

    // This function uses the internal function and should not be detected as dead code.
    function useInternalFunction() public {
        internalFunction();
    }

    // This function is actually used, but the function pointer points to it and should be detected as dead code.
    function usedFunction() internal {
        value = 100;
    }

    // This function pointer points to the used function and should be detected as dead code.
    function () internal f = usedFunction;

    // This function is unused and should be detected as dead code.
    function unusedFunction() internal {
        value = 0;
    }
}
```

- Problem with Current Implementation:



```
(base) ishraqmohammed@Ishraqs-MacBook-Pro slither % slither TestContract15.sol --detect dead-code

FalsePositiveExample.unusedFunction() (TestContract15.sol#31-33) is never used and should be removed
FalsePositiveExample.usedFunction() (TestContract15.sol#23-25) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
TestContract15.sol analyzed (1 contracts with 1 detectors), 2 result(s) found
(base) ishraqmohammed@Ishraqs-MacBook-Pro slither %
```

I. Pseudo Code:



```
            # Add functions that are potentially used as function pointers
            function_pointers = {
                v.type.function for v in contract.state_variables if v.type.function
            }
            functions_used |= {
                f.canonical_name for f in function_pointers if isinstance(f, Function)}

        for function in sorted(self.compilation_unit.functions, key=lambda x: x.canonical_name):
            if (
                function.visibility in ["public", "external"]
                or function.is_constructor
                or function.is_fallback
                or function.is_constructor_variables
            ):
                continue
            if function.canonical_name in functions_used:
                continue
            if isinstance(function, FunctionContract) and (
                function.contract_declarer.is_from_dependency()
            ):
                continue
            # Continue if the functon is not implemented because it means the contract is abstract
            if not function.is_implemented:
                continue
            info = [function, " is never used and should be removed\n"]
            res = self.generate_result(info)
            results.append(res)

        return results
```

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
- Solc-select use 0.5.0(or as required)
- Python slithertocsv.py <Contract_Name.sol> <Output.csv>

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