CSE 6324
Advanced Topics in Software Engineering

Team 8

Team Members

Akshata Raikar - 1002032638

Shovon Pereira - 1002073941

Devyani Singh - 1001959376

Ravi Rajpurohit - 1002079916

Project Vision

- Acknowledging the importance of **GAS** value.
- Separate tools analyzing Gas values
  - Detecting out-of-gas situation
  - Detecting inefficient code wasting Gas
- Merge with another powerful tool Slither
- Make a more robust and diverse tool

Project Features

- **Fuse multiple For-loops :** When multiple For loops are running for the same length, sometimes they can be optimized by achieving the same task in one loop. In these cases, writing multiple loops can cost more Gas than necessary. Hence a warning for such scenarios can be helpful in saving time and other resources.[4]

- **Map instead of Array:** In the case of memory access, searching elements, and dynamic sizing, maps are more efficient as compared to arrays in terms of gas usage and time spent[10]

- **Redundant SSTORE:** Whenever a disk storage operation takes place, disk access costs significant gas value[4]. To optimize this, disk access should be minimized.

- **Calculate Cost of Contract:** It is important to know if a smart contract function exceeds the gas allowed for execution or gas limit. This will allow the developer to know the need for optimization before execution through this detector.

Software Development Plan

- **Iteration 1:**
  - Implemented and tested the 1st detector – **Identify Fusible Loops**.
- **Iteration 2:**
  - Successfully implemented 2nd detector i.e. suggest **to use mapping instead of arrays.**
  - **Test the gas saving** for detector 2 for smart contracts.
  - Planned pseudocode for 3rd feature - **detect redundant storage use.**
  - Document everything for reference.
- **Iteration 3:**
  - Implement the remaining features- **detect redundant storage use and calculate the cost of the contract.**
  - Identify the edge cases.
  - Complete testing of all three features and verify usage for the final use.
  - Address all the corner cases if still left in a document.

UTA

Code Snippets

Array to mapping detector

```python
@staticmethod
def _findArraysAndSuggestMaps(contract: Contract):
    issues = []

    #Iterate through the state variables of the contract:
    for state_variable in contract.variables:
        #Check if the state variable's type is an array (ArrayType):
        if (isinstance(state_variable.type, ArrayType)):
            #If an Array state variable is found, add it to the list of issues:
            issues.append(state_variable)

    return issues
```

Fig 1:Detector 2- ArrayToMapping detector code

```
def _detect(self) -> List[Output]:
    results = []


    for contract in self.contracts:
        #Find array state variables in the contract:
        array_variables = self._findArraysAndSuggestMaps(contract)

        for array_variable in array_variables:
            #Generate a result for the detected issue:
            info = [f"In contract `{contract.name}`, for variable `{array_variable.name}`
                    ({array_variable.type}), consider using a mapping instead of an array."]
            res = self.generate_result(info)
            results.append(res)

    return results
```

Fig 2: entry point for detector 2 – Array to Mapping

Smart Contract to Test

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.18;

contract ArrayUsageExample {
    uint[] public dynamicArray;
    uint[5] public fixedArray;
    mapping(address => uint) public map;

    constructor() {
        dynamicArray.push(1);
        dynamicArray.push(2);
        dynamicArray.push(3);

        fixedArray[0] = 10;
        fixedArray[1] = 20;
        fixedArray[2] = 30;

        map[msg.sender] = 100;
    }

    function addValueToDynamicArray(uint _value) public {
        dynamicArray.push(_value);
    }

    function getValueFromDynamicArray(uint _index) public view returns (uint) {
        require(_index < dynamicArray.length, "Index out of bounds");
        return dynamicArray[_index];
    }
}
```

Fig 3: Smart Contract to test detector 2

Terminal Snippet

```
pereira@PereirasLaptop:~/CSE6324_Team_8/slither$ slither arraymap.sol --detect array-to-mapping
'solc --version' running
'solc arraymap.sol --combined-json abi,ast,bin,bin-runtime,srcmap,srcmap-runtime,userdoc,devdoc,hashes --allo
w-paths .,/home/pereira/CSE6324_Team_8/slither' running
INFO:Detectors:
In contract `ArrayUsageExample`, for variable `dynamicArray` (uint256[]), consider using a mapping instead of
 an array.In contract `ArrayUsageExample`, for variable `fixedArray` (uint256[5]), consider using a mapping i
nstead of an array.Reference: https://github.com/PereiraMavs/CSE6324_Team_8/wiki/Detector-Wiki#fusible-loops
INFO:Slither:arraymap.sol analyzed (1 contracts with 1 detectors), 2 result(s) found
```

Fig 4: Detetctor suggestions logs

Detector 3 - Redundant SSTORE Operation

```
55   uint baseprice;
     ...
108  function setBasePrice(uint new_baseprice){
109     if((msg.sender != owner) && (msg.sender != cbAddress)) throw;
110     baseprice = new_baseprice;
111     for(uint i=0; i < dsources.length; i++) price[dsources[i]] = new_baseprice *
                price_multiplier[dsources[i]];
112  }
113
114  function setBasePrice(uint new_baseprice, bytes proofID){
115     if((msg.sender != owner) && (msg.sender != cbAddress)) throw;
116     baseprice = new_baseprice;
117     for (uint i=0; i < dsources.length; i++) price[dsources[i]] = new_baseprice *
                price_multiplier[dsources[i]];
118  }
```

Fig 5: Redundant SSTORE scenario

EVM Playground SHANGHAI **Fig 6 : Gas Consumption for arrays [8]** Solidity

Current: 3 Total: 73698

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.18;

contract ArrayUsageExample {
    uint[] public dynamicArray;
    uint[5] public fixedArray;
    mapping(address => uint) public map;

    constructor() {
        dynamicArray.push(1);
        dynamicArray.push(2);
        dynamicArray.push(3);

        fixedArray[0] = 10;
        fixedArray[1] = 20;
        fixedArray[2] = 30;

        map[msg.sender] = 100;
    }

    function addValueToDynamicArray(uint _value) public {
        dynamicArray.push(_value);
```

[00]	PUSH1	80
[02]	PUSH1	40
[04]	MSTORE	
[05]	CALLVALUE	
[06]	DUP1	
[07]	ISZERO	
[08]	PUSH2	0010
[0b]	JUMPI	
[0c]	PUSH1	00
[0e]	DUP1	
[0f]	REVERT	
[10]	JUMPDEST	
[11]	POP	
[12]	PUSH1	00

Value to send Wei Advance Mode Run

EVM Playground SHANGHAI

Fig 7 : Gas Consumption for Mapping [8]

Solidity

Current: 3 Total: 66086

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.18;
3
4  contract ArrayUsageExample {
5      mapping(uint => uint) dynamicArray;
6      mapping(uint => uint) fixedArray;
7      mapping(address => uint) public map;
8
9      constructor() {
10
11
12         dynamicArray[0] = 0;
13         dynamicArray[1] = 0;
14         dynamicArray[2] = 0;
15
16         fixedArray[0] = 10;
17         fixedArray[1] = 20;
18         fixedArray[2] = 30;
19
20         map[msg.sender] = 100;
21     }
```

[00]	PUSH1	80
[02]	PUSH1	40
[04]	MSTORE	
[05]	CALLVALUE	
[06]	DUP1	
[07]	ISZERO	
[08]	PUSH2	0010
[0b]	JUMPI	
[0c]	PUSH1	00
[0e]	DUP1	
[0f]	REVERT	
[10]	JUMPDEST	
[11]	POP	

Risks

Problems faced in Iteration 2

- Finding out and identifying SSTORE patterns to minimize that may pose a challenge as Slither actually receives solidity files, not bytecode files.

- Encountered problems during the testing of the newly added detector on WSL. Failed to activate despite new build. Resolved on another computer when using Visual Studio.

- Identifying edge cases for the proposed detectors. We are working on this by brainstorming, comparing with competitors, reading research papers, and recognizing the use cases in the coding community.

Future Risks

Problems anticipated for future iterations –

- The tool's performance after adding the new features also brings a concern. If the performance is compromised in terms of time, it will require additional time to fix it. We plan to keep the last week of iteration 3 for this.

- We can't get the length of the mapping or parse all its elements, so depending on the use case you may be forced to use an array even if you need more gas. So, in that case detector suggestion would not be useful to use mapping. Finding & handling such scenarios.

- Finding out and identifying SSTORE patterns to minimize that may pose a challenge as Slither actually receives solidity files, not bytecode files.

Repository

- https://github.com/PereiraMavs/CSE6324_Team_8
- Version: 2.0

References

- [1] S. S. Kushwaha, S. Joshi, D. Singh, M. Kaur and H. -N. Lee, "Ethereum Smart Contract Analysis Tools: A Systematic Review," in IEEE Access, vol. 10, pp. 57037-57062, 2022, doi: 10.1109/ACCESS.2022.3169902.
- [2] https://ethereum.org/en/developers/docs/smart-contracts/ , accessed on 09/23/2023
- [3] https://ethereum.org/en/developers/docs/gas/ , accessed on 09/23/2023
- [4] T. Chen et al., "GasChecker: Scalable Analysis for Discovering Gas-Inefficient Smart Contracts," in IEEE Transactions on Emerging Topics in Computing, vol. 9, no. 3, pp. 1433-1448, 1 July-Sept. 2021, doi: 10.1109/TETC.2020.2979019.
- [5] I. Ashraf, X. Ma, B. Jiang and W. K. Chan, "GasFuzzer: Fuzzing Ethereum Smart Contract Binaries to Expose Gas-Oriented Exception Security Vulnerabilities," in IEEE Access, vol. 8, pp. 99552-99564, 2020, doi: 10.1109/ACCESS.2020.2995183.
- [6] https://github.com/crytic/slither, accessed on 09/23/2023
- [7] https://arxiv.org/abs/2112.14771, accessed on 09/23/2023
- [8] https://www.evm.codes/
- [9] https://www.linkedin.com/pulse/optimizing-smart-contract-gas-cost-harold-achiando
- [10] https://www.linkedin.com/pulse/how-reduce-smart-contract-gas-usage-arslan-maqbool

Thank You!

Any Questions?