Iteration Report 1

Enhancement of slither for analyzing smart contracts

ADV TOPS SOFTWARE ENGINEERING

2232-CSE-6324-004

Github link: https://github.com/Sampath2901/6324-Project#6324-project

TEAM - 8

SAI NIKHIL KANCHUKATLA – 1002034488
SAMPATH KUMAR MEDIPUDI - 1002032901
SAI KRISHNAM RAJU BHUPATHIRAJU - 1002019782
FARAZ SHAIK AHMAR - 1002035224

Project Plan:

Features:

Inheritance hierarchy: This is the feature that we are going to add for missing zero check detector.

Iteration Plan

Iterations	Goals	Status
1 (Current)	• To gain a deeper understanding of the slither tool, we perform analysis on example smart contracts, providing a practical experience and familiarization with the tool. • Set up and configure the slither tool in order to reproduce an identical error message. • Evaluate the issue more comprehensively and devise a plan to address and resolve it.	completed
2	• Examine the "missing-zero-check" detector, comprehend all its attributes, and grasp its operational flow. • Develop pseudo code that will facilitate the implementation of	In-progress

	the improvements we aim to introduce to the slither tool.	
3	• Translate the pseudo code into real code and evaluate the tool to determine if any new vulnerabilities arise after modifying the existing tool. • Verify the effectiveness of the improvements made to the tool by testing it on smart contracts.	Yet to start

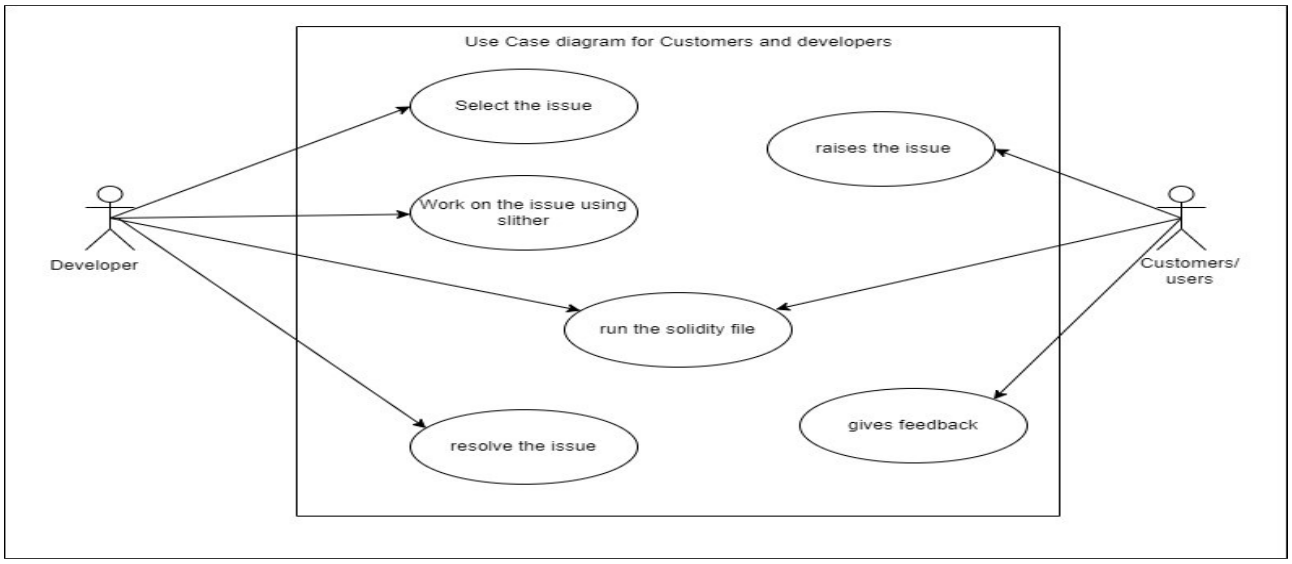
Specification and design:

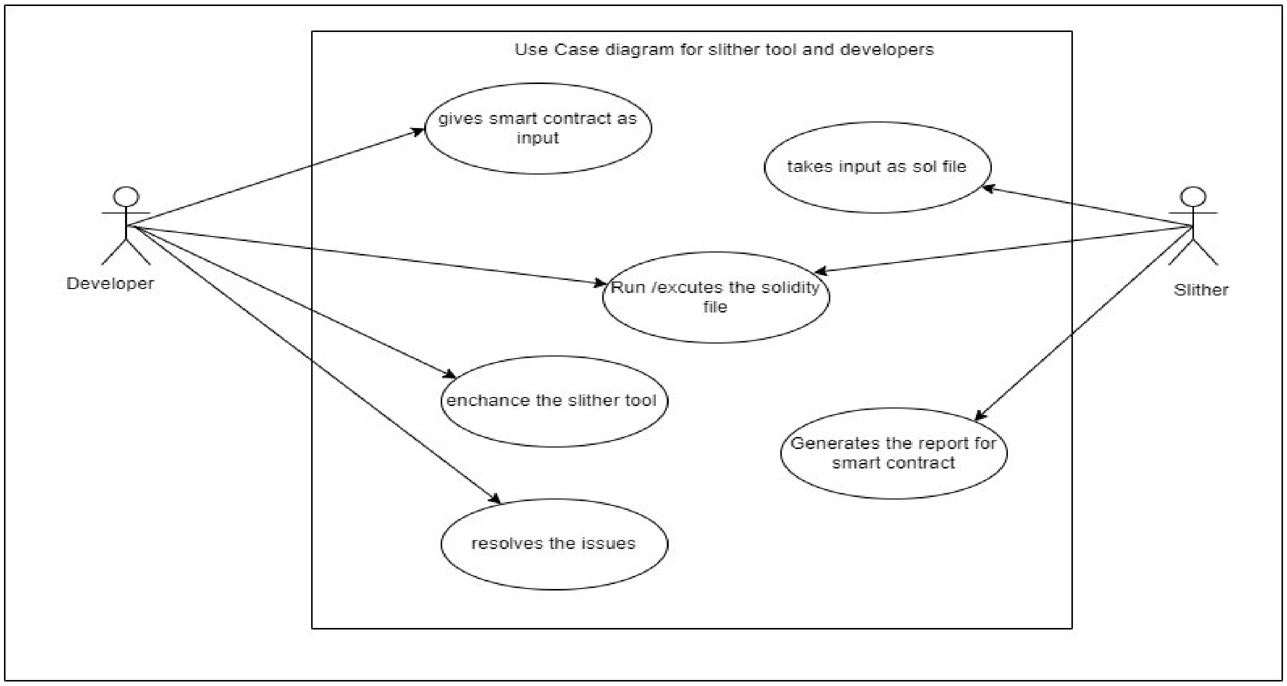
• Inputs and Outputs:

- ➢ Input: Smart contracts
- ➢ Output: The analysis findings from conducting test cases.

• Use-Case (User) [1]:

With the enhancements made in the Slither tool, the user can analyze smart contracts thoroughly.




Use Case diagram for slither tool and developers
gives smart contract as input
takes input as sol file
Developer
Slither
Run /excutes the solidity file
enchance the slither tool
Generates the report for smart contract
resolves the issues

• Mock Designs:

The main idea behind the feature: The zero address is commonly used as a default or provisional value for an address variable when the address parameter or property is not yet initialized. Checking for zero addresses is vital in smart contract security to prevent potential vulnerabilities like reentrancy attacks, where a malicious contract can repeatedly call a vulnerable contract's functions and exhaust its funds.

However, the 'missing-zero-check detector in Slither only examines zero-access checks within the same contract and does not take into account the inheritance hierarchy. To enhance the detector's effectiveness and boost the overall security of analyzed contracts, we plan to include the inheritance hierarchy in the 'missing-zero-check' detector, which is expected to enhance its efficiency.

Code and Tests:
Configuration and setup:[4]

Slither requires Python 3.8+ and solc, the Solidity compiler. We recommend using solc-select to conveniently switch between solc versions, but it is not required. For additional configuration, see the usage documentation.[4]

First, we utilized the Digital Ocean platform to generate a Linux server and installed Visual Studio on it. Next, we included remote SSH access and created a root account to connect to the server. Finally, we followed the below set of instructions to install the necessary dependencies.

- Installed python and checked for the version using the following command :
  python3 –version

- Install npm using the following command :
  apt install npm

- Install nodejs using the following command :
  sudo apt install nodejs

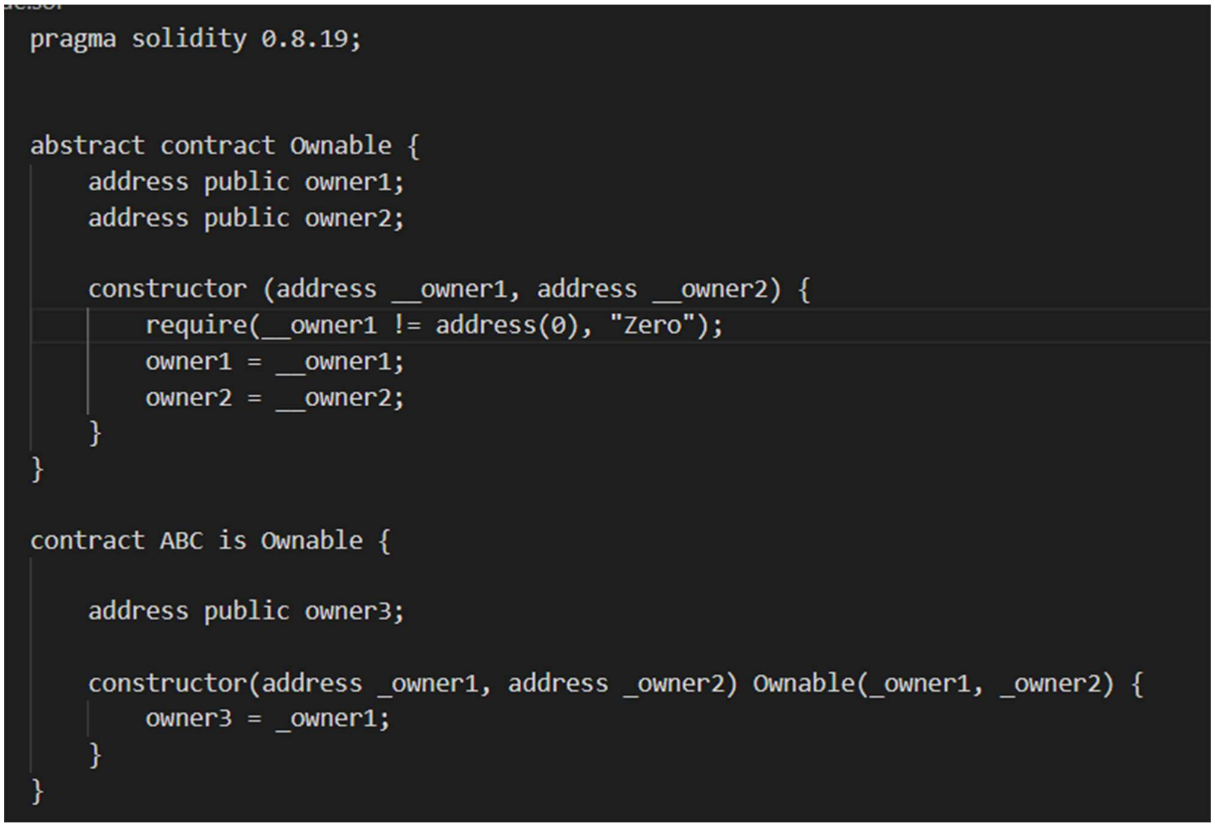
- Install solc using the following command :
  apt install solc

- Install slither using the following command :
  git clone https://github.com/crytic/slither.git && cd slither
  python3 setup.py install

Issue Description :

Issue #981: Copy-paste from github[8]

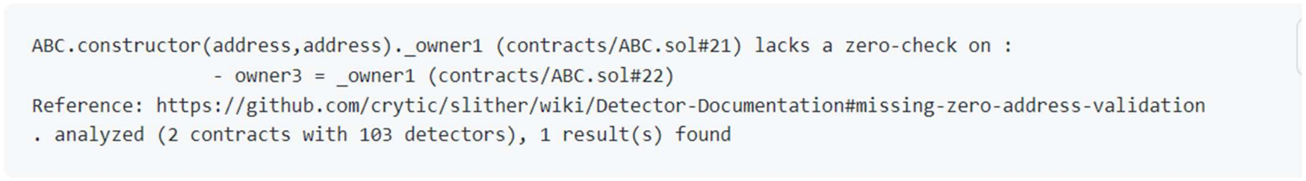
"missing-zero-check" detector doesn't seem to check the arguments for constructor of parent contracts. Even if the parent contract's constructor has zero-access checks & the child contract is using the same variable, it's gets flagged. [8]



```
pragma solidity 0.8.19;


abstract contract Ownable {
    address public owner1;
    address public owner2;

    constructor (address __owner1, address __owner2) {
        require(__owner1 != address(0), "Zero");
        owner1 = __owner1;
        owner2 = __owner2;
    }
}

contract ABC is Ownable {

    address public owner3;

    constructor(address _owner1, address _owner2) Ownable(_owner1, _owner2) {
        owner3 = _owner1;
    }
}
```

[Figure,[8] https://github.com/crytic/slither/issues/981]

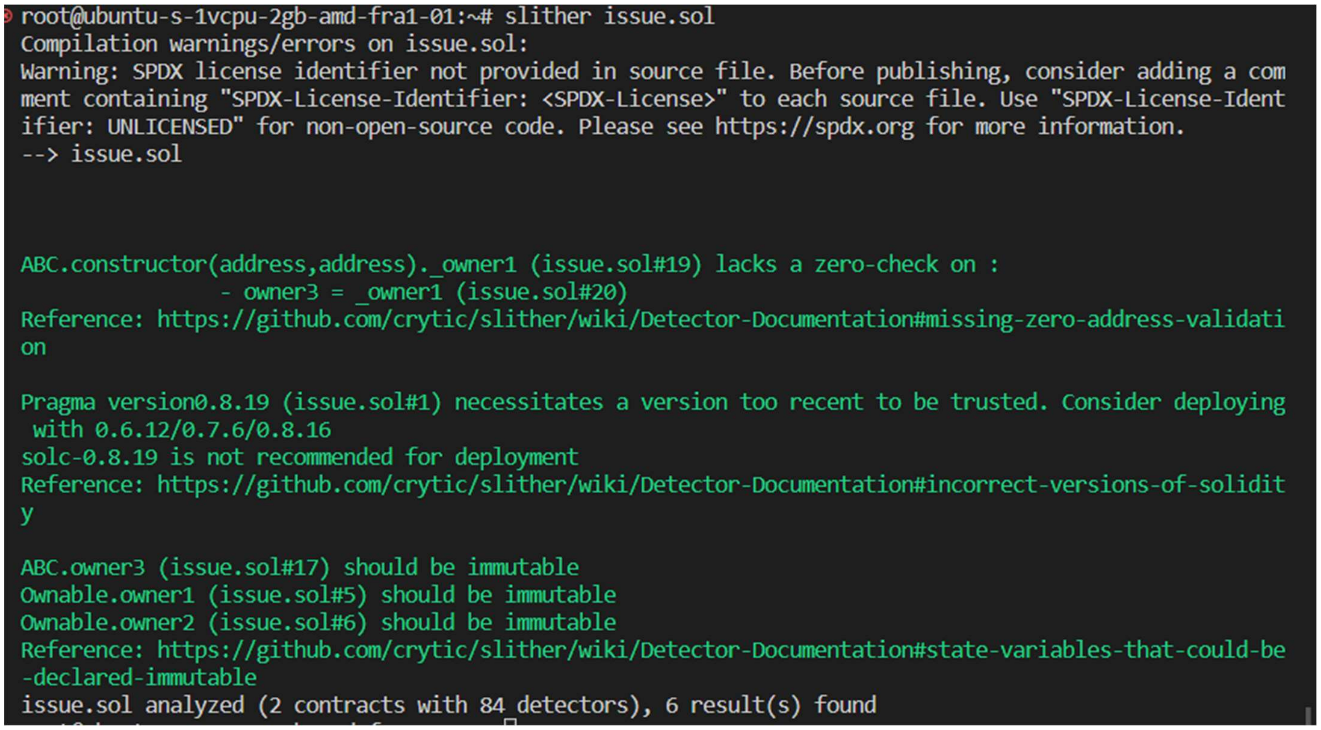
Slither output



```
ABC.constructor(address,address)._owner1 (contracts/ABC.sol#21) lacks a zero-check on :
                - owner3 = _owner1 (contracts/ABC.sol#22)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
. analyzed (2 contracts with 103 detectors), 1 result(s) found
```

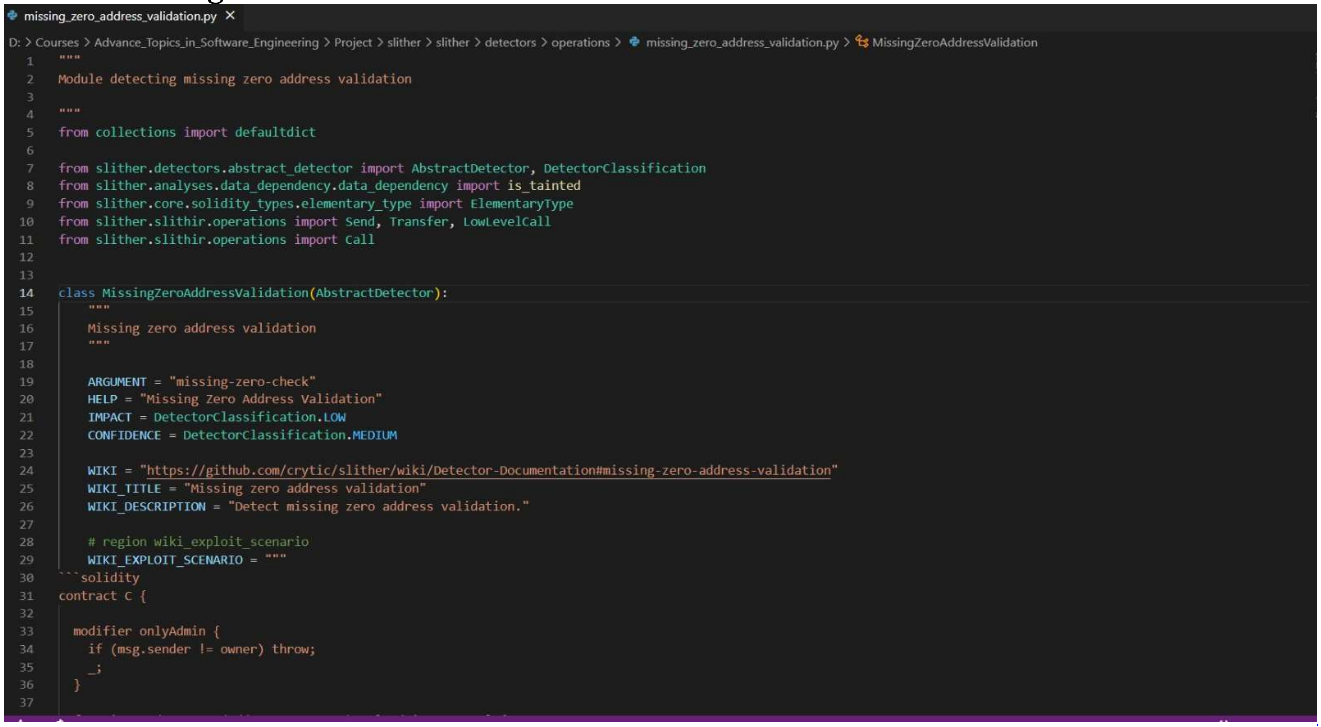
[Figure,[8] https://github.com/crytic/slither/issues/981]

We managed to set up the Slither tool correctly and were able to replicate an identical error message as depicted in the below image



```
root@ubuntu-s-1vcpu-2gb-amd-fra1-01:~# slither issue.sol
Compilation warnings/errors on issue.sol:
Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a com
ment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Ident
ifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.
--> issue.sol



ABC.constructor(address,address)._owner1 (issue.sol#19) lacks a zero-check on :
                - owner3 = _owner1 (issue.sol#20)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validati
on

Pragma version0.8.19 (issue.sol#1) necessitates a version too recent to be trusted. Consider deploying
 with 0.6.12/0.7.6/0.8.16
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidit
y

ABC.owner3 (issue.sol#17) should be immutable
Ownable.owner1 (issue.sol#5) should be immutable
Ownable.owner2 (issue.sol#6) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be
-declared-immutable
issue.sol analyzed (2 contracts with 84 detectors), 6 result(s) found
```

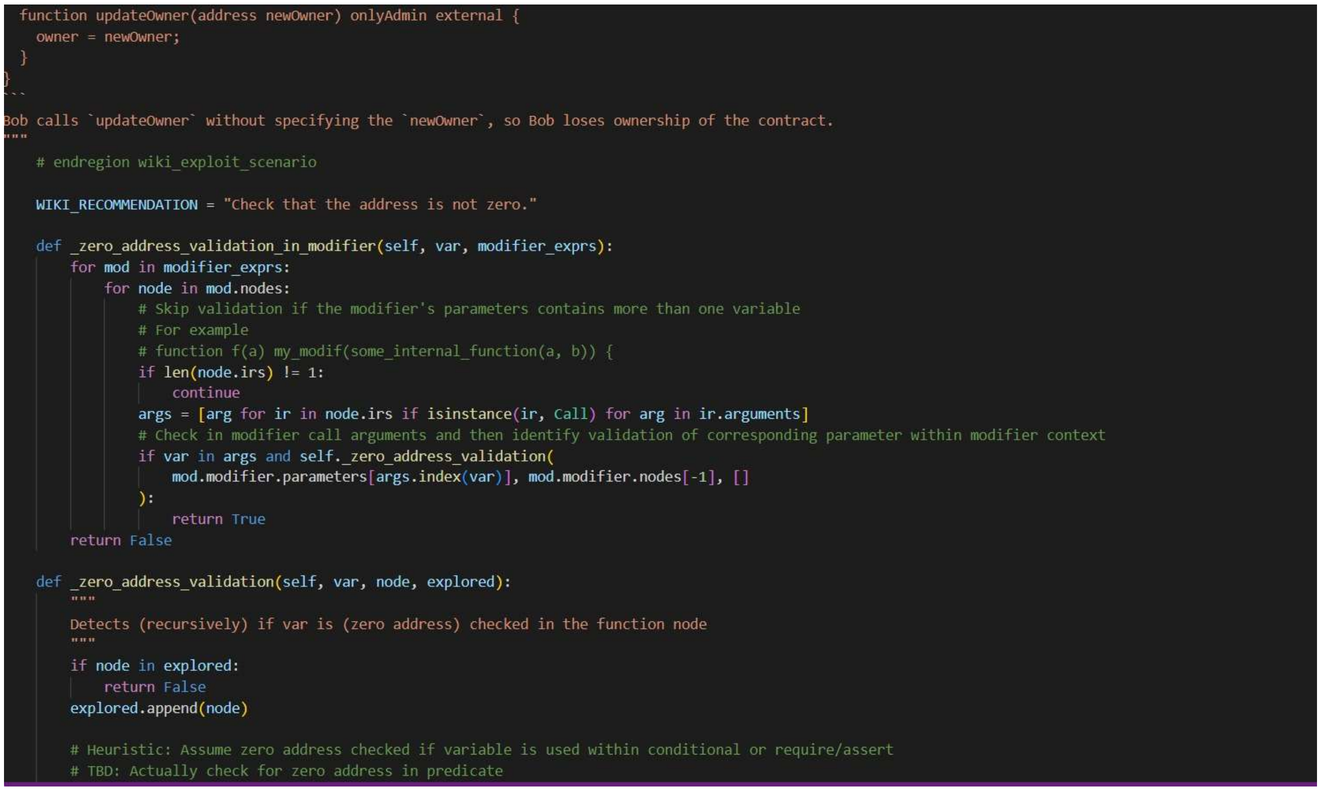
- ➢ We must examine the "missing-zero-check" detector, comprehend all its attributes, and grasp its operational flow.
- ➢ We need to modify “MissingzeroAddressValidation” and add an inheritance hierarchy to it.
- ➢ The below figure contains all the information regarding “MissingzeroAddressValidation”



missing_zero_address_validation.py ×
D: > Courses > Advance_Topics_in_Software_Engineering > Project > slither > slither > detectors > operations > missing_zero_address_validation.py > MissingZeroAddressValidation

~~~python
"""
Module detecting missing zero address validation

"""
from collections import defaultdict

from slither.detectors.abstract_detector import AbstractDetector, DetectorClassification
from slither.analyses.data_dependency.data_dependency import is_tainted
from slither.core.solidity_types.elementary_type import ElementaryType
from slither.slithir.operations import Send, Transfer, LowLevelCall
from slither.slithir.operations import Call


class MissingZeroAddressValidation(AbstractDetector):
    """
    Missing zero address validation
    """

    ARGUMENT = "missing-zero-check"
    HELP = "Missing Zero Address Validation"
    IMPACT = DetectorClassification.LOW
    CONFIDENCE = DetectorClassification.MEDIUM

    WIKI = "https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation"
    WIKI_TITLE = "Missing zero address validation"
    WIKI_DESCRIPTION = "Detect missing zero address validation."

    # region wiki_exploit_scenario
    WIKI_EXPLOIT_SCENARIO = """
```solidity
contract C {

  modifier onlyAdmin {
    if (msg.sender != owner) throw;
    _;
  }

~~~

[Figure, [4] https://github.com/crytic/slither]



```
  function updateOwner(address newOwner) onlyAdmin external {
    owner = newOwner;
  }
}
```
Bob calls `updateOwner` without specifying the `newOwner`, so Bob loses ownership of the contract.
"""
    # endregion wiki_exploit_scenario

    WIKI_RECOMMENDATION = "Check that the address is not zero."

    def _zero_address_validation_in_modifier(self, var, modifier_exprs):
        for mod in modifier_exprs:
            for node in mod.nodes:
                # Skip validation if the modifier's parameters contains more than one variable
                # For example
                # function f(a) my_modif(some_internal_function(a, b)) {
                if len(node.irs) != 1:
                    continue
                args = [arg for ir in node.irs if isinstance(ir, Call) for arg in ir.arguments]
                # Check in modifier call arguments and then identify validation of corresponding parameter within modifier context
                if var in args and self._zero_address_validation(
                    mod.modifier.parameters[args.index(var)], mod.modifier.nodes[-1], []
                ):
                    return True
        return False

    def _zero_address_validation(self, var, node, explored):
        """
        Detects (recursively) if var is (zero address) checked in the function node
        """
        if node in explored:
            return False
        explored.append(node)

        # Heuristic: Assume zero address checked if variable is used within conditional or require/assert
        # TBD: Actually check for zero address in predicate
```

[Figure, [4] https://github.com/crytic/slither]

Risks :

Risks	Major/Minor and it's Probability and Exposure	Solution	Current Status
Installation of Slither (Configuration and setup)	Major risk P = 20% and E = 20 , so extra 4 hrs	For new users to reduce this risk, it's crucial that they thoroughly read the Slither team's documentation.	Completed
Unfamiliarity with the tool	Minor risk P = 30% and E = 10, so extra 3 hrs	It's crucial for new users to read the material given by the Slither team in order to lower this risk and watch the handson videos on youtube.	Completed
```

| | | | |
|---|---|---|---|
| Installation of dependencies(Python) | Minor risk<br>P = 15% and E = 10, so extra 1.5 hrs | Users face difficult in setting up compatable version of python to run slither. | Completed |
| Failure to meet iteration targets | Major risk<br>P = 30% and E = 15 , so extra 4.5 hrs | The idea is to divide the work evenly and work together in a group meeting following each session or on weekends. | In progress |
| Logic failure | Major risk<br><br>P = 30% and E = 20, so extra 6 hrs | The danger can be mitigated by thoroughly testing and debugging the code. | In progress |

**Customers and Users :**

- Slither is used by **Block chain Developers** to run runs a suite of vulnerability detectors, prints visual information about contract details, and provides an API to easily write custom analyses. Slither enables developers to find vulnerabilities, enhance their code comprehension, and quickly prototype custom analyses. [4]
- Slither helps automate security reviews for **Block chain organizations**. Slither provides an API to inspect Solidity code via custom scripts. We use this API to rapidly answer unique questions about the code we're reviewing. We have used Slither to:

1. Identify code that can modify a variable's value.
2. Isolate the conditional logic statements that are influenced by a particular variable's value.

3. Find other functions that are transitively reachable as a result of a call to a particular function. [5]

- **Block chain developers** uses slither to view high-level information about the contract using predefined printers.
- **Researchers** uses its own intermediate representation, SlithIR, to build innovative vulnerability analyses on Solidity. It provides access to the CFG of the functions, the inheritance of the contracts, and lets you inspect Solidity expressions. [5]

Security researchers uses slither to detect and describe security issues with underlying vulnerabilities, severity, and recommended fixes for our smart contract. [6]

Slither is most useful for the following 4 things: [7]

- Automated Vulnerability Detection: Easily detect vulnerabilities or security bugs in your code with low or no human effort.
- Automated optimization detection: Slither can detect code optimizations that the compiler misses while compiling.
- Slither can help you understand code better by summarizing and displaying contract information.
- Slither also helps with code reviews as its API can be easily interacted with by a user

**Feedback from customer:**

It's good to see that the goals set for each iteration of the project seem achievable. It's also encouraging to know that the team has factored in extra hours to account for potential risks. If the team invests the extra hours that were allocated for risk management, I am confident that they will be successful in completing the project. By being proactive and prepared, the team can effectively manage any challenges that may arise and ensure that they stay on track to achieve their goals.

## References:

• [1] https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/

• [2] https://doi.org/10.48550/arXiv.1908.09878

• [3] https://doi.org/10.48550/arXiv.1809.02702

• [4] https://github.com/crytic/slither

• [5] https://blog.trailofbits.com/2018/10/19/slither-a-solidity-static-analysis-framework/

• [6] https://medium.com/coinmonks/automated-smart-contract-security-review-with-slither-1834e9613b01

• [7] https://www.linkedin.com/pulse/how-secure-smart-contracts-slither-damilare-d-fagbemi/?trk=pulse-article_more-articles_related-content-card

• [8] https://github.com/crytic/slither/issues/981

• [9] https://app.diagrams.net/