Advanced Topics in Software Engineering

CSE 6324 – Section 001

Team 9

Iteration 1

(Written Deliverable)

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

TABLE OF CONTENT

I. Abstract:

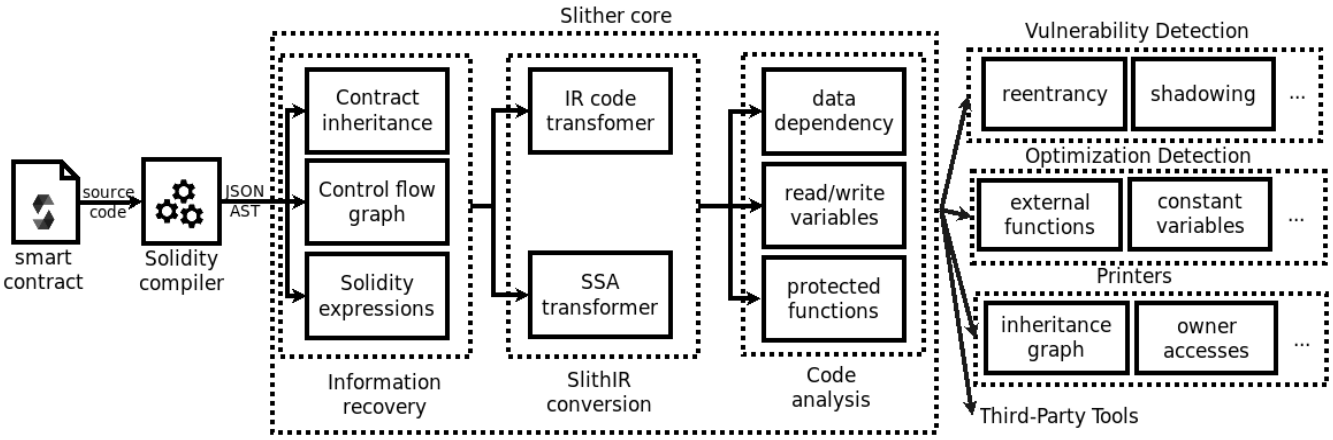
Slither, a widely utilized Solidity security analysis tool. It is enhanced by incorporating functionality to handle nested structs. Basically, in Solidity smart contracts, nested structs are a prevalent feature. However, Slither currently lacks support for them. Consequently, Slither is unable to conduct security vulnerability analysis on those areas in contracts utilizing nested structs.

Integrating support for nested structs in Slither would yield the benefit of Enabling Slither users to analyze a broader range of Solidity smart contracts for security vulnerabilities. Integrating support for nested structs in Slither would also yield the benefit of Enhancing Slither's comprehensiveness and precision in evaluating Solidity smart contracts.

GitHub Issue - https://github.com/crytic/slither/issues/2077

II. Architecture:

- Slither is a smart tool that checks smart contracts, which has several key components.
- At its core is a strong analysis engine that uses advanced methods to thoroughly examine smart code contracts.
- This engine works with different detectors, each designed to find certain weaknesses or patterns.
- Additionally, a detailed parser moves through Solidity source code, picking out important features for further analysis.
- The tool's architecture smoothly fits with Ethereum's system, offering information about security flaws and possible improvements.



Slither Architecture

III. Project Plan:

Iterations	Goals	Achieved Goals
1.	Setup and Configuration of tools. Selecting the Issue. Integrate Slither and Smart Contract with Hardhat. Codebase and Folder Structure Analysis.	The tool was installed successfully, and the sample smart contract was successfully executed. Added the smart contract file with appropriate Solidity version and also analyzed the existing Slither codebase and its folder structure.
2.	Detecting Nested Arrays Implementing the detector. New Detector Development. Work on reviews from Iteration 1.	(Incomplete)
3.	Solidity Contract Analysis and Report Generation. Work on reviews from Iteration 2.	(Incomplete)

IV. About the Detector:

- The initial step involves a comprehensive review of the Solidity language specifications, focusing on how Nested Structs are structured and utilized. This in-depth understanding enables us to formulate the logic and patterns required to detect these nested structures accurately.

- Next, we delve into Slither's existing architecture, acquainting ourselves with its AST (Abstract Syntax Tree) and relevant data structures. Understanding how Slither processes and interprets Solidity code assists us in identifying suitable hooks within the codebase to integrate the detector seamlessly.

- Having established the integration points, we proceed to design the detection algorithm. This algorithm entails parsing through the AST and analyzing variable declarations to pinpoint instances where nested structs are employed. By structuring an efficient algorithm, we optimize the detector's accuracy and speed.

- Following the algorithm's design, we implement and integrate it into the Slither tool, testing thoroughly to ensure its effectiveness across a spectrum of Solidity smart contracts. We then fine-tune the detector based on the test results, aiming for precise detection while minimizing false positives.

V. List of Biggest Risk:

- **Maintaining and compatibility:** Overtime when the solidity code might evolve the detection algorithm might become outdated or incompatible to the newer versions of the code, hence maintaining the algorithm's accuracy and maintaining the compatibility is a complex task.
- **False Positives and False Negatives:** Slither and similar tools may produce false positives and false negatives. False positives occur when the tool reports issues that are not actual vulnerabilities, leading to wasted time and resources in investigating and mitigating non-existent problems.
- **Limited Scope:** Smart contract analysis tools, including Slither, focus on specific patterns and known vulnerabilities. They may not cover all possible security risks or novel attacks. Malicious actors are continuously evolving their tactics, and new vulnerabilities may emerge that the tools are not equipped to detect. Additionally, these tools may not consider the specific context of a contract or its use case. Developers may implement custom logic and features that are not covered by automated analysis, leaving potential vulnerabilities unnoticed.
- **Thorough Testing and Validation:** Rigorously test the detection algorithm on a variety of smart contracts to identify and address false positives and false negatives. Continuous validation is essential to ensure the algorithm's accuracy.

VI. Plans to deal with the risk and issues:

Risk/ Issue Factor	Mitigation Plan	Risk Exposure
Maintaining and compatibility	Sticking with a fixed version of solidity in the detector.	Risk impact: 2 weeks Probability that risk will materialize: 92% Risk Exposure: 1 week approx.
Thorough Testing and Validation	Create detailed list of tests for the detector	Incomplete.
Technical issue - Inexperience with Python	Learning python via tutorials.	Risk impact: 5 weeks Probability that risk will materialize: 96% Risk Exposure: 3 weeks approx.
Technical issue- Inexperience with Solidity	Learning solidity language concepts by tutorials.	Risk impact: 4 weeks Probability that risk will materialize: 90% Risk Exposure: 4 weeks approx.
Technical issue- Complexity Building the Nested Struct and using in a Function	Finding out how to build nested structs and write them in the smart contracts.	Analyzing existing solidity open-source code and going through Solidity Documentation and relevant work. (-)

VII. Specification and Design:

- **Installation on Ubuntu OS:**

- Hardhat

```
npm install -save-dev hardhat
```

```
npx hardhat init
npm install --save-dev @nomicfoundation/hardhat-toolbox@^3.0.0
```

- Python

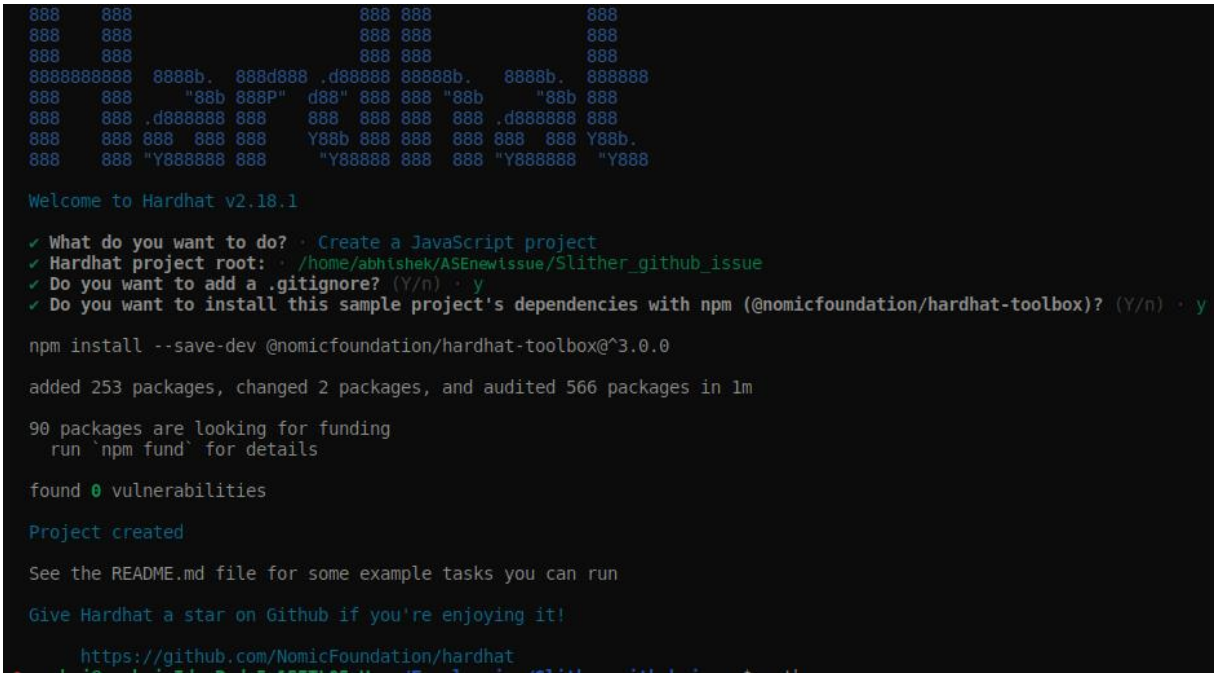
```
sudo apt update
 sudo apt install python3
sudo apt-get -y install python3-pip
```

- Slither

```
pip3 install slither-analyzer
```

- **Code and Screenshots**

- Hardhat installed



- Python Installed

```
abhi@abhishek-IdeaPad-5-15ITL05-Ua: $ python3 - -version
Python 3.10.12
```

- Pip3 installed

```
abhi@abhishek-IdeaPad-5-15ITL05-Ua: $ pip3 - -version
pip 22.0.2 from /us/lib/python3/dist-packages/pip (python 3.10)
```

- Slither installed

```
abhi@abhishek-IdeaPad-5-15ITL05-Ua:~/ASEnewissue/Slither_github_issues$ slither--version
0.9.6
```

- Smart Contract

```
contracts > NestedStructs.sol
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.19;
3
4  contract NestedStructExample {
5      struct HomeAddress {
6          string street;
7          string city;
8          string country;
9      }
10
11     struct Person {
12         string name;
13         uint age;
14         HomeAddress location;
15     }
16
17     mapping(address => Person) public people;
18
19     function addPerson(string memory name, uint age, string memory street, string memory city, string memory country) public {
20         HomeAddress memory addr = HomeAddress(street, city, country);
21         Person memory newPerson = Person(name, age, addr);
22         people[msg.sender] = newPerson;
23     }
24
25     function getPerson() public view returns (string memory, uint, string memory, string memory, string memory) {
26         Person storage person = people[msg.sender];
27         return (person.name, person.age, person.location.street, person.location.city, person.location.country);
28     }
29 }
```

- Slither command on Smart Contract

```
abhi@abhishek-IdeaPad-5-15ITL05-Ua:~/Freelancing/Slither_github_issue$ slither
'npx hardhat clean' running (wd: /home/abhishek/ASEnewissue/Slither _github issue)
'npx hardhat clean --global' running (wd: /home/abhishek/ASEnewissue/Slither_github_issue)
'npx hardhat compile --force' running (wd: /home/abhishek/ASEnewissue/Slither_github_issue)
INFO: Detectors:
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Slither: analyzed (1 contracts with 88 detectors), 1 result (s) found
```

The current slither detector is unable to detect the nested Structs. We will improve the detector to detect and suggest better!

VIII. Comparison:

		Slither	Securify	SmartCheck	Solhint
Accuracy	False positives	10.9%	25%	73.6%	91.3%
	Flagged contracts	112	8	793	81
	Detections per contract	3.17	2.12	10.22	2.16
Performance	Average execution time	0.79 ± 1	41.4 ± 46.3	10.9 ± 7.14	0.95 ± 0.35
	Timed out analyses	0%	20.4%	4%	0%
Robustness	Failed analyses	0.1%	11.2%	10.22%	1.2%
Reentrancy examples	DAO	✓	✗	✓	✗
	Spankchain	✓	✗	✗	✗

IX. Target Users/Customers:

Customers:
1. Dr. Christoph Csallner
2. Mohammed Rifat Arefin
3. Team 10

Users:
CSE 6324-001 class, Blockchain Developers, Smart Contract Auditors, Security Professionals, Open-source Contributors, Educators, and Trainees.

X. References:

1. Slither: https://github.com/crytic/slither
2. Issue: https://github.com/crytic/slither/issues/2077
3. Install Hardhat:hardhat.org/hardhat-runner/docs/getting-started#overview
4. Install Slither: https://github.com/crytic/slither#how-to-install
5. https://www.immunebytes.com/blog/slither-a-solidity-static-analyzer-for-smart-contracts/

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