ADVANCED TOPICS IN SOFTWARE ENGINEERING CSE 6324-001

ITERATION-2 PRESENTATION

TEAM 9

ABHISHEK WADHWANI – 10020352719

MOUNIKA KOTTAPALLI – 1002085510

NITIN RAJ THUMMA – 1002080555

SAI RAGHU RAMI REDDY DONTIREDDY - 1002014523

PROJECT PROPOSAL

The Proposal is to enhance the Slither Framework Analysis by adding functionality to handle nested structs.

It provides fine-grained information about smart contract codes and has the necessary flexibility to support many applications.

Enabling support for nested structs in Slither would expand its capabilities, allowing users to analyze a wider array of Solidity smart contracts for potential security vulnerabilities.

This enhancement would also bolster Slither's effectiveness by increasing its thoroughness and precision in evaluating Solidity code.

ISSUE

Slither-read-storage: Add support for Nested Structs #2077

Enhancing Slither with nested struct support would enable broader Solidity contract security analysis.

This improvement will expand Slither's scope and accuracy in evaluating contracts, making it more comprehensive and precise.

It addresses the current limitation in analyzing security vulnerabilities within smart contracts that utilize nested structs. [2]

#	Task Description	Status
1.	Installation a) Python v3.6+ b) Hardhat c) Ubuntu OS d) Slither	Completed
2.	Decide the detector to start working on	Completed
3.	a) Find the sample solidity contract to analyze it using Slither b) Work on reviews from Inception	Completed
4.	a) Integrate Slither and Smart Contract with Hardhat. b) Codebase and Folder Structure Analysis.	Completed
5.	a) Load the solidity smart contract. b) Set up the solidity compiler version based on the sample solidity smart contract. c) Work on reviews from Iteration 1	Completed
6.	a) Create a .sol python file. b) Installing dotenv c) Create dotenv file	Completed
7.	a) Write a detector to detect vulnerability. b) Analyze the solidity contract with nested structs.	In Progress
8.	a) Generate reports b) Work on reviews from iteration 2	Incomplete

PROJECT PLAN

ABOUT THE DETECTOR

- We designed a detection algorithm for nested structs, integrated it into Slither, and fine-tuned it for accuracy.
- Our current focus pertains to the "read_storage.py" file within the Slither tool addressing the function "_find_struct_var_slot()"
- At present, our contract incorporates a nested structure, but the function fails to identify the slots for the nested struct. It mistakenly attributes slot 0 to the "Person" struct.
- We've made adjustments by assigning slot size arrays. We are actively exploring methods to determine the slot size for such dynamic arrays.
- The code for the function in question has been modified to successfully identify the slot, yet we are still working on resolving the issue related to extracting further details from the nested struct. Please note that this is an ongoing work in progress.

DETECTOR CODE

read_storage.py > AssignmentOperation

```python
class SlitherReadStorage:
    @staticmethod
    def _find_struct_var_slot(
        elems: List[StructureVariable], slot_as_bytes: bytes, struct_var: str
    ) -> Tuple[str, str, bytes, int, int]:
        """
        Finds the slot of a structure variable.
        Args:
            elems (List[StructureVariable]): Ordered list of structure variables.
            slot_as_bytes (bytes): The slot of the struct to begin searching at.
            struct_var (str): The target structure variable.
        Returns:
            info (str): Info about the target variable to log.
            type_to (str): The type of the target variable.
            slot (bytes): The storage location of the target variable.
            size (int): The size (in bits) of the target variable.
            offset (int): The size of other variables that share the same slot.
        """
        slot = int.from_bytes(slot_as_bytes, "big")
        offset = 0
        type_to = ""
        slot_size = 256
        for var in elems:
            var_type = var.type
            if isinstance(var_type, ElementaryType):
                size = var_type.size
                # if offset >= 256:
                #     slot += 1
                #     offset = 0
                # if struct_var == var.name:
                #     type_to = var_type.name
                #     break  # found struct var
                # offset += size
            elif isinstance(var_type, ArrayType):
                size = slot_size
            elif isinstance(var_type, MappingType):
                size = slot_size
            elif isinstance(var_type, UserDefinedType) and isinstance(var_type.type, Structure):
                var_type = var_type.type
                assert var_type

                (
```

read_storage.py > AssignmentOperation

```python
class SlitherReadStorage:
    def _find_struct_var_slot(
            elif isinstance(var_type, UserDefinedType) and isinstance(var_type.type, Structure):
                var_type = var_type.type
                assert var_type

                (
                    _,
                    tmp_type_to,
                    tmp_slot_as_bytes,
                    size,
                    offset,
                ) = SlitherReadStorage._find_struct_var_slot(
                    elems=var_type.elems_ordered,
                    slot_as_bytes=int.to_bytes(slot, 32, byteorder="big"),
                    struct_var=struct_var,
                )

                tmp_slot = int.from_bytes(tmp_slot_as_bytes, "big")

                # found struct var
                if tmp_type_to:
                    slot = tmp_slot
                    break
                else:
                    size = (tmp_slot - slot) * slot_size

            else:
                size = slot_size
                logger.info(f"{type(var_type)} is current not implemented in _find_struct_var_slot")
                    # found struct var
            if struct_var == var.name:
                type_to = var_type.name
                break

            offset += size

            if offset >= slot_size:
                slot += 1
                offset = 0

        slot_as_bytes = int.to_bytes(slot, 32, byteorder="big")
        info = f"\nStruct Variable: {struct_var}"
        return info, type_to, slot_as_bytes, size, offset
```

SPECIFICATION AND DESIGN

A. Figuring out blockchains onto which we can deploy on Smart Contract

- The Slither tool can only analyze storage slots in smart contracts that are deployed on a blockchain.
- After researching many options, we opted for Polygon Mumbai.
- We chose Polygon Mumbai because it's efficient, scalable, and friendly to developers. [6]

B. Nested Structure Smart Contract

Creating a .sol file consisting of the following smart contract [12].

contracts > NestedStructs.sol

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

contract NestedStructExample {
    struct HomeAddress {
        string street;
        string city;
        string country;
    }

    struct Person {
        string name;
        uint age;
        HomeAddress location;
    }

    mapping(address => Person) public people;

    function addPerson(string memory name, uint age, string memory street, string memory city, string memory country) public {
        HomeAddress memory addr = HomeAddress(street, city, country);
        Person memory newPerson = Person(name, age, addr);
        people[msg.sender] = newPerson;
    }

    function getPerson() public view returns (string memory, uint, string memory, string memory, string memory) {
        Person storage person = people[msg.sender];
        return (person.name, person.age, person.location.street, person.location.city, person.location.country);
    }
}
```

C. Installing dotenv package

1. Need dotenv package to manage environment variables in our applications.
2. This is needed so that we don’t expose our private key to the public. [10]

```
up to date, audited 567 packages in 1s

91 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

D. Creating a dotenv file

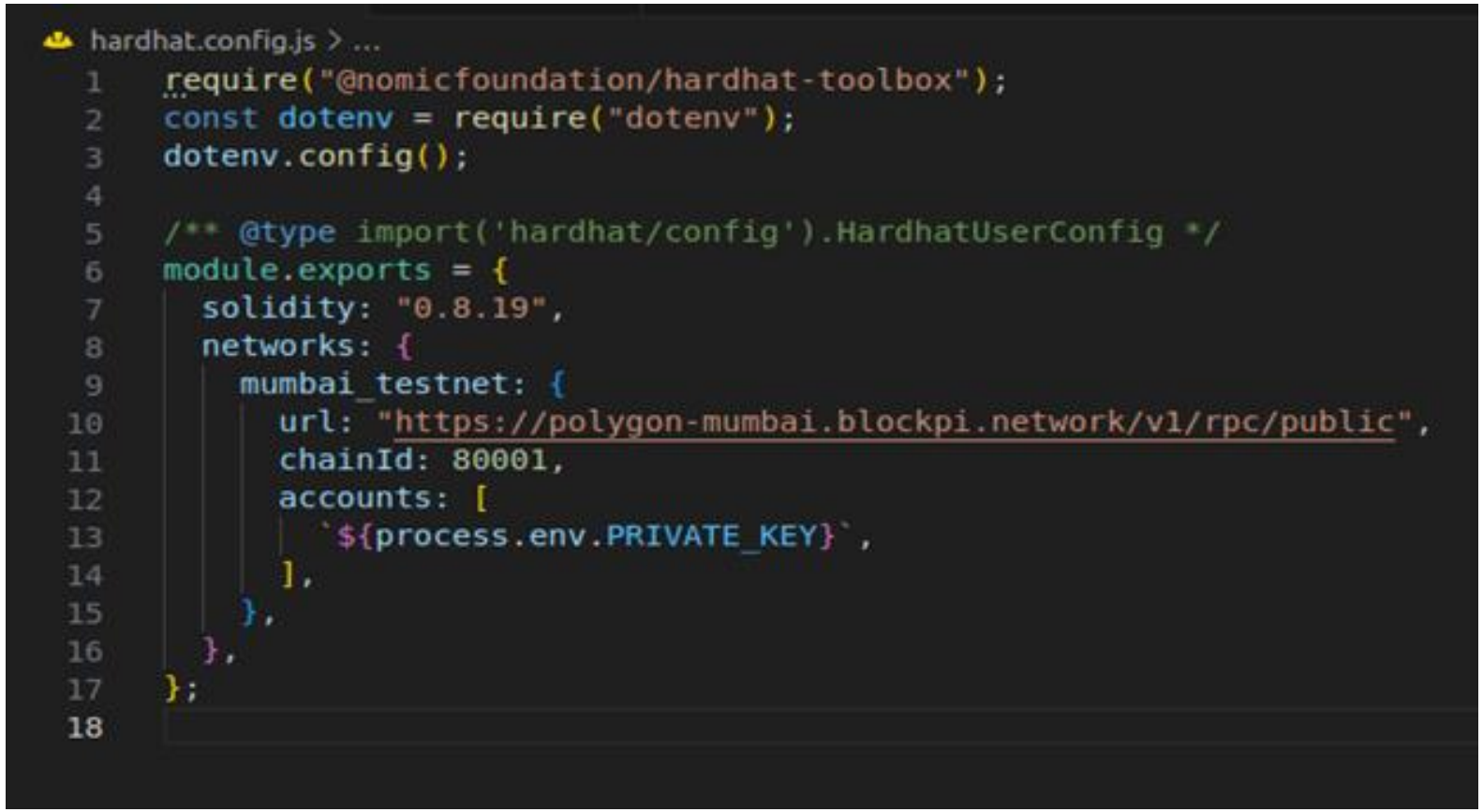
- dotenv is used to store sensitive or configuration-related information separately from the code by using environment variables. These variables are then accessed within the code to provide security and flexibility. [9]

```
.env
1
2  PRIVATE_KEY=your private key
3  ACCOUNT_ADDRESS=your account address
4  ETHERSCAN_API_FOR_POLYGON=your etherscan api key
5
6
```

E. Writing Deployment Scripts for Hardhat Tool

i. Install the Hardhat toolbox
ii. Import the dotenv package
iii. Load the environment variables from the .env file into the current process

iv. Specify the RPC URL of the Mumbai testnet
v. The chainId field specifies the chain ID of the Mumbai testnet which is 80001.
vi. Specify the private keys of the account that Hardhat should use to deploy and test your contracts on the Mumbai testnet. [7]



F. Command for deploying the contract on the polygon blockchain

```
-IdeaPad-5-15ITL05-Ua:~/Slither_github_issue$ npx hardhat run scripts/deploy.js --network mumbai_testnet
NestedStructExample deployed to: 0xB091F9c43b4D58C6040F65841E4DF8Be27726BC2
```

The smart contract is deployed on polygon mumbai testnet with an address of 0xB091F9c43b4D58C6040F65841E4DF8Be27726BC2 [9] https://mumbai.polygonscan.com/address/0xB091F9c43b4D58C6040F65841E4DF8Be27726BC2

G. Verify the contract because Slither works only on Verified Contracts:

- Slither-read–storage task needs the source of the contracts available.
- Update hardhat.config.js for accommodating polygon api key.

```
16    },
17    etherscan: {
18      apiKey: {
19        // sepolia: process.env.ETHERSCAN_API_KEY as string,
20        polygonMumbai: process.env.ETHERSCAN_API_FOR_POLYGON,
21      },
22    },
```

Write the verified script.

- Mention the address of the contract to verify.
- Give the path in the following order along with the smart contract name.

scripts > verify.js > main

```
1  const Hre = require("hardhat");
2
3  async function main() {
4    await Hre.run("verify:verify", {
5      address: "0xB091F9c43b4D58C6040F65841E4DF8Be27726BC2",
6      contract: "contracts/NestedStructs.sol:NestedStructExample",
7    });
8
9    console.log("Verified!!!");
10 }
11
12 main()
13   .then(() => process.exit(0))
14   .catch((error) => {
15     console.error(error);
16     process.exit(1);
17   });
18
```

- Run the command to verify the contract. Mention the mumbai_testnet as defined in the hardhat.config.js file that we have made.

H. Running the command for the Detector on the Deployed Contract

- Need to mention mumbai and then the address as the contract is deployed on Polygon Mumbai Testnet

```
INFO:Slither-read-storage:
Contract 'NestedStructExample'
NestedStructExample.people with type mapping(address => NestedStructExample.Person) is located at slot: 0

INFO:Slither-read-storage:
Name: people
Type: mapping(address => NestedStructExample.Person)
Slot: 0
```

- This isn’t throwing any error as of now for nested structs. It shows slot 0 of our contract as can be seen. We are trying to modify the detector so that it gives further details about how the slots for Person and HomeAddress could be managed.
- The current slither detector is unable to detect the nested Structs. We will improve the detector to detect and suggest better!

RISK FACTORS AND MITIGATION PLAN

Risk Factor	Mitigation Plan	Risk Exposure
Maintaining and compatibility	Sticking with a fixed version of solidity in the detector.	Trying to maintain the same version and implement the code trying to make it compatible.
Thorough Testing and Validation	Create a detailed list of tests for the nested structs.	Incomplete.
Technical issue - Inexperience with Python	Learning python via tutorials.	With minimal knowledge of programming Language, it is difficult to work on the code.
Technical issue Inexperience with Solidity	Learning solidity language concepts by tutorials.	With no knowledge of Solidity, it is very difficult to work on Nested Struct and it takes time to get hand of it.
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.
Not receiving the anticipated output	Rechecking and writing the code and find where it went wrong	Analyzing the complete code and trying to enhance the code to get a better outcome.

CUSTOMERS AND USERS

- CSE 6324-001 class
- Aditya Vichare (CSE 6324-001 student)
- Blockchain Developers
- Smart Contract Auditors
- Open-source Contributors
- Educators and Trainees.

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

REFERENCES

1. https://github.com/crytic/slither
2. https://github.com/crytic/slither/issues/2077
3. hardhat.org/hardhat-runner/docs/getting-started#overview
4. https://github.com/crytic/slither#how-to-install
5. https://www.immunebytes.com/blog/slither-a-solidity-static-analyzer-for-smart-contracts/
6. https://www.alchemy.com/overviews/mumbai-testnet
7. https://hardhat.org/hardhat-runner/docs/guides/deploying
8. https://docs.chainstack.com/reference/polygon-getting-started
9. https://www.npmjs.com/package/dotenv
10. https://onboardbase.com/blog/env-file-guide/
11. https://docs.arbitrum.io/stylus/how-tos/local-stylus-dev-node
12. https://stackoverflow.com/questions/70011797/return-nested-struct-inside-struct-in-solidity

Questions?

Thank You!