Iteration 3 Report

Adding new detectors to Slither

The University of Texas at Arlington

Advanced Topics in Software Engineering

Fall 2022, CSE 6324 - 001

GitHub link: https://github.com/AtulUpadhye17/CSE-6324-Team4

Team 4:

Atul Upadhye (1002030159)

Kundana Vaka(1001827398)

Yamini Dhulipalla(1001913007)

Srikar Sai Yarlagadda(1001860709)

Vigneshwar Selvaraj(1001863627)

Project Plan:

Objective:

For our project, we plan to implement new detectors and integrate them into the slither developer version to analyze the solidity smart contracts efficiently.

Overview of Slither differs from other static analysis tools:
"Trail of Bits" has published a paper on Slither and compares its bug detection with other static analysis tools by doing experiments for finding issues in Ethereum smart contracts in terms of speed, robustness, the balance of detection, and false positives.[1]

Comparison with Existing tool for Iteration 3:

We compared Slither with other static analysis frameworks such as Solhint, Ethlint :

Detectors	Solhint	Ethlint	Current Slither	Extended Slither (Our Tool)
Imports on Top	Yes	Yes	No	Yes
Pragma on Top	No	Yes	No	Yes

Iteration 3 Plan:

- Compare Slither with its competitors and identify the detectors that are missing in the tool.
- Write code and test detector for each of the identified detectors.
- Integrate the implemented detectors.
- Built and tested the tool.
- Merged the code and performed code review.

Iteration 4 Plan:

- Implement and integrate the remaining detectors.
- Test those detectors and check for any bugs and fix them.
- Create the documentation of the project.

Risks Faced during Iteration 3:

Risk	Type	Risk Exposure	Plan for Mitigating
The virtual Machine got crashed and we had to set up everything from scratch.	Major	This particular risk has a 50% chance of happening, and it will take 15 hours to fix it in this iteration. The risk exposure for this risk is therefore 7.5 additional hours of work resolving it.	Setup virtual machine again.
We have faced an issue in setting up the slither developer environment in our local machine. It took us more time to fix those compiler and runtime errors.	Major	This particular risk has a 75% chance of happening, and it will take 10 hours to fix it in this iteration. The risk exposure for this risk is therefore 7.5 additional hours of work resolving it.	We fixed those issues by installing the necessary compiler certificates which are not mentioned in the developer instructions.

Understanding the slither code was not apparent enough.	Minor	This particular risk has a 40% chance of happening, and it will take 9 hours to fix it in this iteration. The risk exposure for this risk is therefore 4 additional hours of work resolving it.	We have tried different test cases with many sample sol files to catch on with the code.
Integrating our detectors with the slither tool was complex since we had to modify our code with slither API features. Thus, making sure the newly added detector flow is in accord with the existing detectors.	Major	This particular risk has a 75% chance of happening, and it will take 10 hours to fix it in this iteration. The risk exposure for this risk is therefore 7.5 additional hours of work resolving it.	We asked for Shovon's help figure out the part of integration and additional detectors we can implement.

Implemented Detectors:

Imports on Top: In general, the Solidity import statement imports code from one module into another. To import all the code from a module, we specify the import keyword followed by the module name.

Import statements appear at the top of the solidity file, beneath any comments and docstrings that may exist, and before module globals and constants. This is because importing modules or packages at the top of a file makes the structure of the code clearer.

This detector also checks through nested input solidity files to find whether the import statement follows the style guide convention.[7]

Detector Code : Imports on Top :

```python
from slither.detectors.abstract_detector import AbstractDetector, DetectorClassification

class ImportsOnTop(AbstractDetector):
    """
    Detect if import statements are placed at the top of the file.
    """
    ARGUMENT = "imports-on-top"
    HELP = "Import statements should always be placed at the top of the file."
    IMPACT = DetectorClassification.OPTIMIZATION
    CONFIDENCE = DetectorClassification.MEDIUM

    WIKI = "https://github.com/trailofbits/slither/wiki/Adding-a-new-detector"
    WIKI_TITLE = "Imports On Top"
    WIKI_DESCRIPTION = "Detect if import statements are placed at the top of the file."
    WIKI_EXPLOIT_SCENARIO = ".."
    WIKI_RECOMMENDATION = "Imports Statement should always be placed at top of the file."

    def _detect(self):
        results = []

        for n in self.slither.crytic_compile.filenames:

            rootFileName = n.absolute
            word = 'import'
            word_number = 0
            contract_num = 0
            contract_list = []
            with open(rootFileName, 'r') as fp:
                # read all lines in a list
                lines = fp.readlines()
                for line in lines:
                    # check if string present on a current line
                    if line.find(word) != -1:
                        word_number = lines.index(line)
                    elif line.startswith('contract') or line.startswith('struct') or line.startswith('library'):
                        contract_list.append(lines.index(line))
                if not len(contract_list) == 0:
                    contract_num = min(contract_list)
            if word_number > contract_num:
                info = ["Import statement should be on top : "+rootFileName,"\n"]

                res = self.generate_result(info)

                results.append(res)

        return results
```

Contract Sample:

Users > upadhyeatul > Desktop > cargps.sol

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.16;
3
4  contract CarGPS {
5
6      using GPS for *;
7
8      address endpoint;
9
10     constructor() {
11         endpoint = SIGNAL_TOWER_A;
12     }
13
14 }
15
16 import "./gps.sol";
17
```

Users > upadhyeatul > Desktop > gps.sol

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.16;
3
4  struct Coordinates {
5      uint latitude;
6      uint longitude;
7  }
8
9  library GPS {
10     // some library functions...
11 }
12 import "./endpoints.sol";
13
```

Output :

```
(slither-dev) [11/03/22]seed@VM:~/.../slither_detectors$ slither cargps.sol

solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Import statement should be on top : /home/seed/ase_fall22/slither_detectors/cargps.sol
Import statement should be on top : /home/seed/ase_fall22/slither_detectors/gps.sol
Reference: https://github.com/trailofbits/slither/wiki/Adding-a-new-detector

pragma statement should be on top : /home/seed/ase_fall22/slither_detectors/endpoints.sol
Reference: https://github.com/trailofbits/slither/wiki/Adding-a-new-detector
cargps.sol analyzed (2 contracts with 85 detectors), 4 result(s) found
(slither-dev) [11/03/22]seed@VM:~/.../slither_detectors$
```

Pragma on Top:

Pragma statements appear at the top of the solidity file, beneath any comments and docstrings that may exist, and before module globals and constants.[8]

This detector also checks through nested input solidity files to find whether the pragma statement follows the style guide convention.

Detector Code : Pragma on Top :

```
from slither.detectors.abstract_detector import AbstractDetector, DetectorClassification

class PragmasOnTop(AbstractDetector):
    """
    Detect if pragma statements are placed at the top of the file.
    """

    ARGUMENT = "pragme-on-top"
    HELP = "pragma statements should always be placed at the top of the file."
    IMPACT = DetectorClassification.OPTIMIZATION
    CONFIDENCE = DetectorClassification.MEDIUM

    WIKI = "https://github.com/trailofbits/slither/wiki/Adding-a-new-detector"
    WIKI_TITLE = "pragma On Top"
    WIKI_DESCRIPTION = "Detect if pragma statements are placed at the top of the file."
    WIKI_EXPLOIT_SCENARIO = ".."
    WIKI_RECOMMENDATION = "pragma Statement should always be placed at top of the file."

    def _detect(self):
        results = []

        for n in self.slither.crytic_compile.filenames:

            rootFileName = n.absolute
            word = 'pragma'
            word_number = 0
            contract_num = 0
            contract_list = []
            with open(rootFileName, 'r') as fp:
                # read all lines in a list
                lines = fp.readlines()
                for line in lines:
                    # check if string present on a current line
                    if line.find(word) != -1:
                        word_number = lines.index(line)
                    elif line.startswith('contract') or line.startswith('struct') or line.startswith('library'):
                        contract_list.append(lines.index(line))
                if not len(contract_list) == 0:
                    contract_num = min(contract_list)
            if word_number > contract_num:
                info = ["pragma statement should be on top : "+rootFileName,"\n"]

                res = self.generate_result(info)

                results.append(res)

        return results

```

Contract Sample :

Users > upadhyeatul > Desktop > demoContract.sol

```
1  // SPDX-License-Identifier: MIT
2
3  contract DemoContract {
4
5      function first() public{
6
7      }
8      function second() public {
9
10     }
11
12     constructor() {
13
14     }
15
16 }
17 pragma solidity ^0.8.16;
```

Output :

```
(slither-dev) [11/03/22]seed@VM:~/.../slither_detectors$ slither demoContract.sol

solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Incorrect constructor Order found in DemoContract (demoContract.sol#4-17)
Reference: https://github.com/trailofbits/slither/wiki/Adding-a-new-detector

pragma statement should be on top : /home/seed/ase_fall22/slither_detectors/demoContract.sol
Reference: https://github.com/trailofbits/slither/wiki/Adding-a-new-detector
demoContract.sol analyzed (1 contracts with 85 detectors), 3 result(s) found
(slither-dev) [11/03/22]seed@VM:~/.../slither_detectors$
```

Setup Virtual Environment for Slither (Ubuntu):

User Hands-On Setup :

First Install Slither for checking the pseudocode for detectors:
$ sudo pip3 install slither-analyzer

Now you can analyse some sample contracts using Slither:
Command:

$ slither <filename>.sol

Developer Instructions Setup:

$ sudo add-apt-repository ppa:ethereum/ethereum
$ sudo apt-get update
$ sudo apt-get install solc

For Developer Instructions, it was recommended to use Python Virtual Envwrapper:

$ sudo pip3 install virtualenvwrapper
$ export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
$ source /usr/local/bin/virtualenvwrapper.sh
$ mkvirtualenv --python=`which python3` slither-dev

Now once you have your virtual env ready, clone the slither directory from git:

```
$ git clone https://github.com/trailofbits/slither
$ cd slither/
```

Below commands installs the necessary packages required by slither :

```
$ pip install -e ".[dev]"
```

There are chances of solc version mismatch, you need to use slither specific version, for ex :

```
$ solc-select use 0.8.17
```

Adding new Detectors to Slither :

Detector Skeleton:

```python
from slither.detectors.abstract_detector import AbstractDetector, DetectorClassification


class Skeleton(AbstractDetector):
    """
    Documentation
    """

    ARGUMENT = 'mydetector' # slither will launch the detector with slither.py --detect mydetector
    HELP = 'Help printed by slither'
    IMPACT = DetectorClassification.HIGH
    CONFIDENCE = DetectorClassification.HIGH

    WIKI = ''

    WIKI_TITLE = ''
    WIKI_DESCRIPTION = ''
    WIKI_EXPLOIT_SCENARIO = ''
    WIKI_RECOMMENDATION = ''

    def _detect(self):
        info = ['This is an example']
        res = self.generate_result(info)

        return [res]
```

- **ARGUMENT** lets you run the detector from the command line
- **HELP** is the information printed from the command line
- **IMPACT** indicates the impact of the issue. Allowed values are:
    - **DetectorClassification.OPTIMIZATION**: printed in green
    - **DetectorClassification.INFORMATIONAL**: printed in green
    - **DetectorClassification.LOW**: printed in green
    - **DetectorClassification.MEDIUM**: printed in yellow
    - **DetectorClassification.HIGH**: printed in red
- **CONFIDENCE** indicates your confidence in the analysis. Allowed values are:
    - **DetectorClassification.LOW**
    - **DetectorClassification.MEDIUM**
    - **DetectorClassification.HIGH**
- **WIKI** constants are used to generate automatically the documentation.[9]

Integration :

You can integrate your detector into Slither by:

- Adding it in slither/detectors/all_detectors.py.(We followed this approach)
- or, by creating a plugin package.

```
86 from .functions.protected_variable import ProtectedVariables
87 from .functions.permit_domain_signature_collision import DomainSeparatorCollision
88
89 from .constructor.incorrect_constructor_name import IncorrectConstructorName
90 from .constructor.incorrect_constructor_order import IncorrectConstructorOrder
91 from .style.imports_on_top import ImportsOnTop
92 from .style.pragmas_on_top import PragmasOnTop
```

To check if detectors are properly integrated run below command :

```
(slither-dev) [11/03/22]seed@VM:~/.../slither_detectors$ slither --list-detectors
```

Integrated Detectors :

```
|  81 |       imports-on-top       |                Import statements should always be placed at the top of the file.                |  Optimization |   Mediu
m   |
|  82 |  incorrect-constructor-name |                    Constructor Declaration should use 'constructor' keyword.                    |  Optimization |   Mediu
m   |
|  83 | incorrect-constructor-order |                             Constructor not in sequence with functions                          |  Optimization |   Mediu
m   |
|  84 |       pragme-on-top        |                pragma statements should always be placed at the top of the file.                |  Optimization |   Mediu
m   |
+-----+----------------------------+-----------------------------------------------------------------------------------------+---------------+--------
```

Customers and Users:

- Any security audit firm, smart contract developer ,security expert, or academic researcher can use this tool with our new detectors added to Slither to make the smart contract auditing process more efficient.

Feedback from Shovan :

Shovan suggested us to review the Solidity style guidelines and try to implement the detectors for them, as well as work on integrating all the implemented detectors. For this iteration, we have focused on implementing and integrating the detectors.

References:

[1] Overview. (n.d.). Retrieved October 15, 2022, from https://blog.trailofbits.com/2019/05/27/slither-the-leading-static-analyzer-for-smart-contracts/
[2] GitHub - crytic/slither: Developer installation. (n.d.). GitHub. Retrieved October 15, 2022, from https://github.com/crytic/slither/wiki/Developer-installation
[3] Rule Index of Solhint. (n.d.). Solhint. Retrieved October 17, 2022, from https://protofire.github.io/solhint/docs/rules.html
[4] User Guide — Solium 1.0.0 documentation. (n.d.). https://ethlint.readthedocs.io/en/latest/user-guide.html
[5] Adding a new detector · crytic/slither Wiki. (n.d.-b). GitHub. https://github.com/crytic/slither/wiki/Adding-a-new-detector
[6] imports-on-top Solhint. (n.d.). Solhint. https://protofire.github.io/solhint/docs/rules/order/imports-on-top.html
[7] Style Guide — Solidity 0.8.17 documentation. (n.d.). https://docs.soliditylang.org/en/v0.8.17/style-guide.html