TRAIL OF BITS

Slither: API walkthrough

Who am I?

- **Josselin Feist (@montyly)**

- **Trail of Bits: trailofbits.com**
  - We help developers build safer software
  - R&D focused: we use the latest program analysis techniques
  - Slither, Echidna, Tealer, Caracal, solc-select, ..

ToB Twitter list

Agenda

- What is Slither
- Slither internals & API
- SlithIR

Slides & hackathon details: https://github.com/crytic/ethdam

Slither

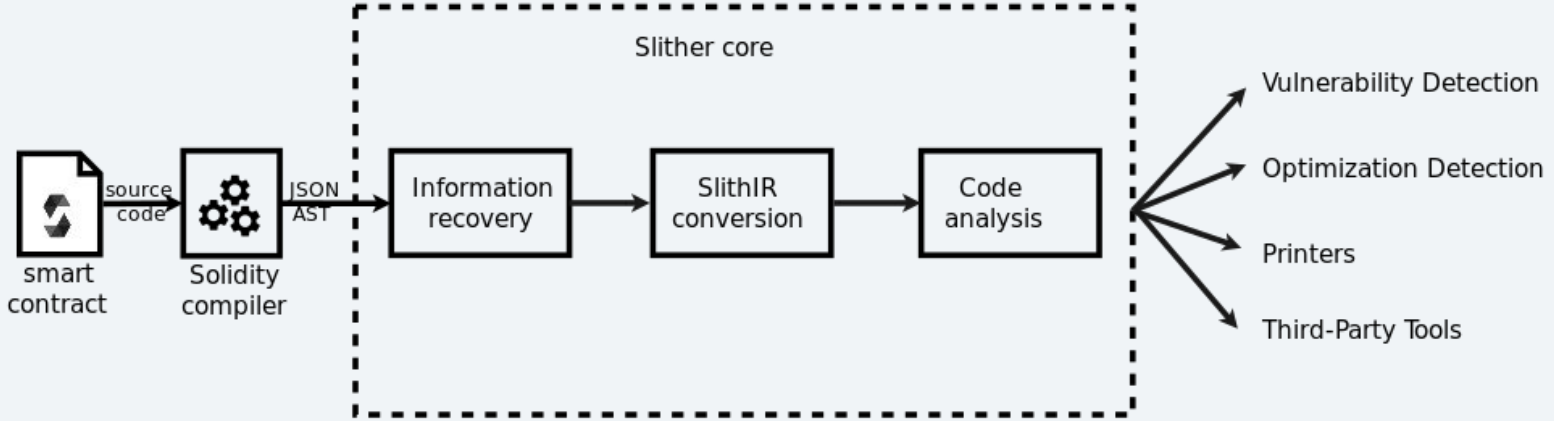
- **Static analysis framework for Solidity & Vyper**
  - Vulnerability detection
  - Optimization detection
  - Code understanding
  - Assisted code review

SLITHER

https://github.com/crytic/slither

```
pip3 install -u slither-analyzer
```

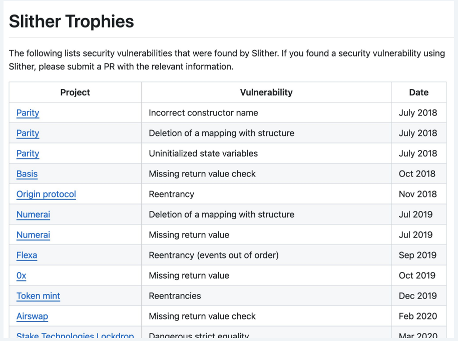
Slither


smart contract
source code
Solidity compiler
JSON AST
Slither core
Information recovery
SlithIR conversion
Code analysis
Vulnerability Detection
Optimization Detection
Printers
Third-Party Tools

Vulnerability Detection

Detectors

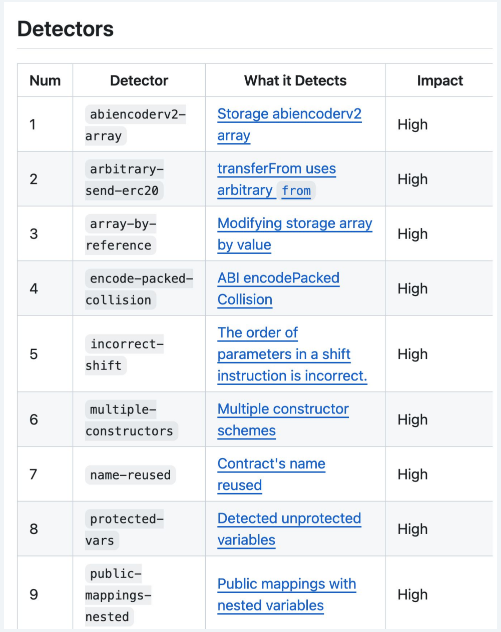
- **90+ public detectors**
- **From high severity to optimization**
- **Based on audit reports**
- **Proven track of vulnerabilities**



Slither Trophies

The following lists security vulnerabilities that were found by Slither. If you found a security vulnerability using Slither, please submit a PR with the relevant information.

Project	Vulnerability	Date
Parity	Incorrect constructor name	July 2018
Parity	Deletion of a mapping with structure	July 2018
Parity	Uninitialized state variables	July 2018
Basis	Missing return value check	Oct 2018
Origin protocol	Reentrancy	Nov 2018
Numerai	Deletion of a mapping with structure	Jul 2019
Numerai	Missing return value	Jul 2019
Flexa	Reentrancy (events out of order)	Sep 2019
0x	Missing return value	Oct 2019
Token mint	Reentrancies	Dec 2019
Airswap	Missing return value check	Feb 2020
Stake Technologies Lockdrop	Dangerous strict equality	Mar 2020



Detectors

Num	Detector	What it Detects	Impact
1	`abiencoderv2-array`	Storage abiencoderv2 array	High
2	`arbitrary-send-erc20`	transferFrom uses arbitrary `from`	High
3	`array-by-reference`	Modifying storage array by value	High
4	`encode-packed-collision`	ABI encodePacked Collision	High
5	`incorrect-shift`	The order of parameters in a shift instruction is incorrect.	High
6	`multiple-constructors`	Multiple constructor schemes	High
7	`name-reused`	Contract's name reused	High
8	`protected-vars`	Detected unprotected variables	High
9	`public-mappings-nested`	Public mappings with nested variables	High

Detectors cheatsheet

- **List the detectors**
  - slither –list-detectors
- **Run a given detector**
  - slither [target] –detect DETECTOR_NAME

Detectors triage

- **Triage a detector**
  - //slither-disable-next-line DETECTOR_NAME
  - // slither-disable-start [detector] … // slither-disable-end [detector]
- **Triage a reentrancy**
  - @custom:security non-reentrant

Detectors triage

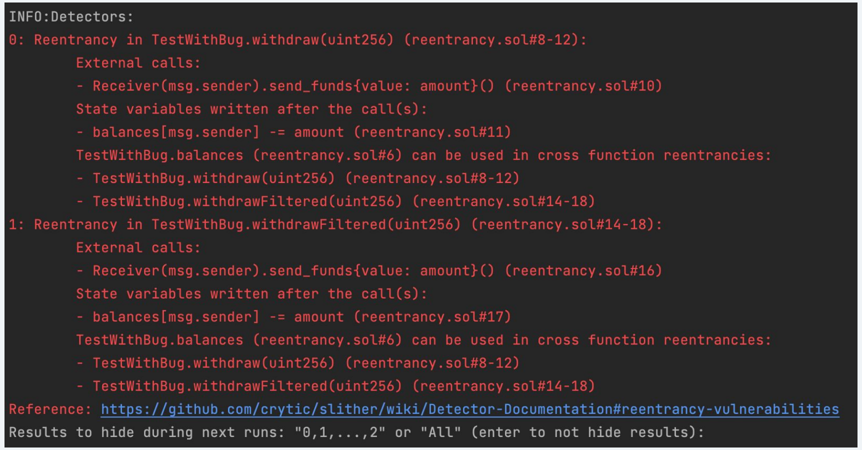
```
// slither-disable-start reentrancy-eth
function withdraw(uint amount) public{
     require(amount <= balances[msg.sender]);
     Receiver(msg.sender).send_funds{value: amount}();
     balances[msg.sender] -= amount;
}
// slither-disable-end reentrancy-eth
```

Detectors triage

```
/// @custom:security non-reentrant
OwnerContract owner_contract;


function withdraw(uint amount) public{
     require(amount <= balances[msg.sender]);
     owner_contract.external_call{value: amount}();
     balances[msg.sender] -= amount;
}
```

Detectors triage

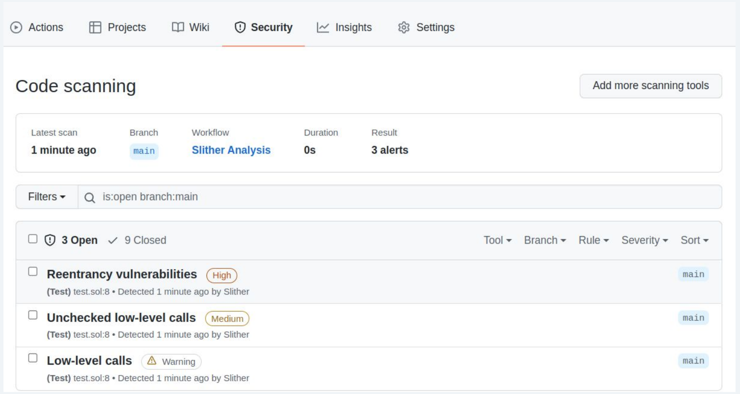
- **Interactive triage mode**
  - slither . --triage-mode
  - Create a slither.db.json file



```
INFO:Detectors:
0: Reentrancy in TestWithBug.withdraw(uint256) (reentrancy.sol#8-12):
        External calls:
        - Receiver(msg.sender).send_funds{value: amount}() (reentrancy.sol#10)
        State variables written after the call(s):
        - balances[msg.sender] -= amount (reentrancy.sol#11)
        TestWithBug.balances (reentrancy.sol#6) can be used in cross function reentrancies:
        - TestWithBug.withdraw(uint256) (reentrancy.sol#8-12)
        - TestWithBug.withdrawFiltered(uint256) (reentrancy.sol#14-18)
1: Reentrancy in TestWithBug.withdrawFiltered(uint256) (reentrancy.sol#14-18):
        External calls:
        - Receiver(msg.sender).send_funds{value: amount}() (reentrancy.sol#16)
        State variables written after the call(s):
        - balances[msg.sender] -= amount (reentrancy.sol#17)
        TestWithBug.balances (reentrancy.sol#6) can be used in cross function reentrancies:
        - TestWithBug.withdraw(uint256) (reentrancy.sol#8-12)
        - TestWithBug.withdrawFiltered(uint256) (reentrancy.sol#14-18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
Results to hide during next runs: "0,1,...,2" or "All" (enter to not hide results):
```

Detectors triage

- crytic/slither-action



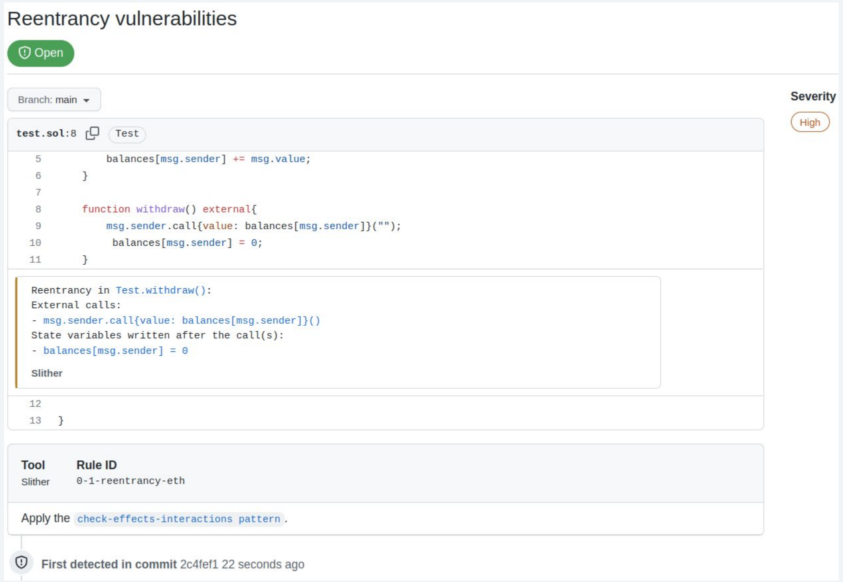
Code scanning

Latest scan	Branch	Workflow	Duration	Result
1 minute ago	main	Slither Analysis	0s	3 alerts

Reentrancy vulnerabilities High — main
(Test) test.sol:8 • Detected 1 minute ago by Slither

Unchecked low-level calls Medium — main
(Test) test.sol:8 • Detected 1 minute ago by Slither

Low-level calls Warning — main
(Test) test.sol:8 • Detected 1 minute ago by Slither



Reentrancy vulnerabilities

test.sol:8 Test

```
        balances[msg.sender] += msg.value;
    }

    function withdraw() external{
        msg.sender.call{value: balances[msg.sender]}("");
         balances[msg.sender] = 0;
    }
```

```
Reentrancy in Test.withdraw():
External calls:
- msg.sender.call{value: balances[msg.sender]}()
State variables written after the call(s):
- balances[msg.sender] = 0
```

Slither

```

}
```

Tool	Rule ID
Slither	0-1-reentrancy-eth

Apply the `check-effects-interactions pattern`.

First detected in commit 2c4fef1 22 seconds ago

Detectors cheatsheet

- **Filtering paths**
    - slither . --filter-paths "openzepellin"
    - slither . --filter-paths "Migrations.sol|ConvertLib.sol"
- **Use python regex**

Target cheatsheet

- ***slither [target]***
  - [target] must be compileable
- **Direct solc**
  - slither file.sol
  - Not recommended if remapping is needed
- **Compilation framework**
  - [target] is the root directory of the project
    - Support for foundry, hardhat, truffle, dapp, etc
    - If you run from the directory, run “slither .”
- **Extra foundry support**
  - slither src/file.sol supports foundry remapping automatically

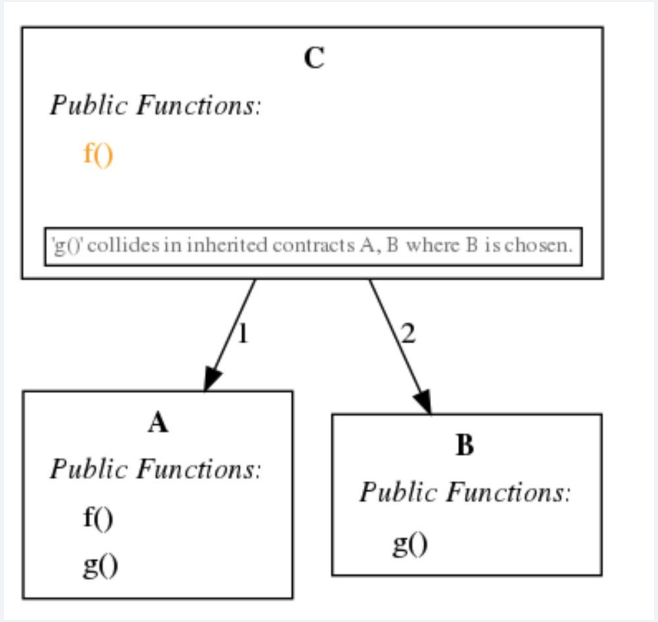
Target cheatsheet

- **Onchain analysis**
  - slither 0x…
    - Require source code on etherscan
  - slither NETWORK:0x..
    - mainet,optim,goerli,sepolia,tobalaba,bsc,testnet.bsc,arbi,testnet.arbi,poly,mumbai,avax,testnet.avax,ftm,goerli.base,base,gno,polyzk
    - Default mainet

Code Understanding

Printers

- **Visual representation of the code**
- **Quick review**
  - human-summary
  - inheritance-graph
  - contract-sumarry
  - loc
- **In-depth review**
  - call-graph
  - cfg
  - function-summary
  - vars-and-auth
  - not-pausable

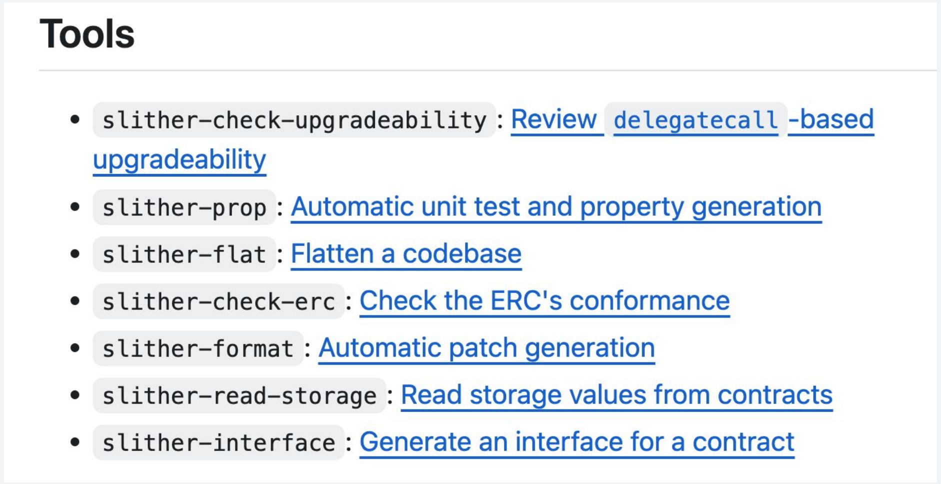


Printers cheatsheet

- **List printers**
  - slither –list-printers
- **Run a printer**
  - slither [target] –print PRINTER_NAME

Generic Static Analysis Framework

Assisted code review



Tools

- `slither-check-upgradeability`: Review `delegatecall`-based upgradeability
- `slither-prop`: Automatic unit test and property generation
- `slither-flat`: Flatten a codebase
- `slither-check-erc`: Check the ERC's conformance
- `slither-format`: Automatic patch generation
- `slither-read-storage`: Read storage values from contracts
- `slither-interface`: Generate an interface for a contract

Python API

- **Python API to help during a code review**
  - Inspect contract information
  - Including data dependency/taint analysis

Python API

- **Ex: What functions can modify a state variable:**

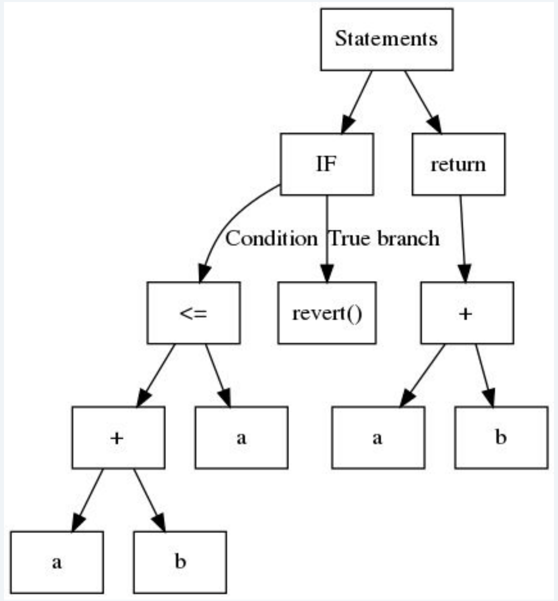
```python
slither = Slither('function_writing.sol')
contract = slither.get_contract_from_name('Contract')[0]
var_a = contract.get_state_variable_from_name('a')

functions_writing_a = contract.get_functions_writing_variable(var_a)

print('The function writing "a" are {}'.format([f.name for f in functions_writing_a]))
```

Slither Internals

Slither Internals

- **Input: solc AST**

```
function safeAdd(uint256 a, uint256 b) ...
    if (a + b <= a) {
        revert();
    }
    return a + b;
}
```



Slither Layers

- **Compilation units**
  - ~ group of files used by one call to solc
- **Contracts**
  - Inheritance, state variables, functions
- **Functions**
  - Attributes, CFG
- **Control Flow Graphs**
  - Nodes
- **Expression & IR**
  - Operations

Slither object

```
from slither import Slither

# Create the slither object

sl = Slither("test.sol")



# Works with the other supported target, ie :

sl = Slither("0xdac17f958d2ee523a2206206994597c13d831ec7") # Load USDT

# Add etherscan_api_key for rate limit

sl = Slither("0xdac17f958d2ee523a2206206994597c13d831ec7", etherscan_api_key="..")
```

Compilation unit

- **~ group of files used by one call to solc**
- **Most targets have 1 compilation, but not always true**
  - Partial compilation for optimization
  - Multiple solc version used
  - Etc..

```
sl = Slither("test.sol")

sl.compilation_units # array of SlitherCompilationUnit
```

Compilation unit

- **Why compilation unit matters?**
  - Some APIs might be not intuitive
  - Ex: looking for a contract based on the name?
    - Can have multiple contracts
- **For hacking you can *(probably)* use the first compilation unit**
  - compilation_unit = sl.compilation_units[0]

Compilation unit - cheatsheet

- slither/core/compilation_unit.py
- **contracts: List[Contract]**
  - List of all the contracts
- **contracts_derived(self): List[Contract]**
  - List of the most derived contracts. I.e. contract not inherited
- **get_contract_from_name(contract_name): List[Contract]**
  - *Usually: returns one contract*
- **Top level objects**
  - [structures | enums | events | variables | functions]_top_level

Compilation unit - example

```
from slither import Slither

sl = Slither("0xdac17f958d2ee523a2206206994597c13d831ec7")

compilation_unit = sl.compilation_units[0]


# Print all the contracts from the USDT address

print([str(c) for c in compilation_unit.contracts])


# Print the most derived contracts from the USDT address

print([str(c) for c in compilation_unit.contracts_derived])
```

Compilation unit - example

```
% python test.py

['SafeMath', 'Ownable', 'ERC20Basic', 'ERC20', 'BasicToken', 'StandardToken', 'Pausable',
'BlackList', 'UpgradedStandardToken', 'TetherToken']


['SafeMath', 'UpgradedStandardToken', 'TetherToken']
```

Contract - cheatsheet

- slither/core/declarations/contract.py
- **name: str**
- **Inheritance**
  - inheritance: List[Contract]: c3 linearization order
  - derived_contracts: List[Contract]: contracts derived from it
- **General objects**
  - enums | events | structures
- **Variables**
  - state_variables: List[StateVariable]: list of accessible variables
  - state_variables_ordered: List[StateVariable]: all variable ordered by declaration
- **get_function_from_signature(sig)**

Contract - example

```python
from slither import Slither

sl = Slither("0xdac17f958d2ee523a2206206994597c13d831ec7")

compilation_unit = sl.compilation_units[0]
```

```python
# Print all the state variables of the USDT token

contract = compilation_unit.get_contract_from_name("TetherToken")[0]

print([str(v) for v in contract.state_variables])
```

Contract - example

```
% python test.py

['owner', 'paused', '_totalSupply', 'balances', 'basisPointsRate', 'maximumFee',
'allowed', 'MAX_UINT', 'isBlackListed', 'name', 'symbol', 'decimals', 'upgradedAddress',
'deprecated']
```

Function - cheatsheet

- [core/declarations/function.py]
- **solidity_signature: str**
- **entry_point: Node**
- **Elements**
  - expressions, variables, nodes, modifiers
- **Operations**
  - [state |local]_variable_[read |write]
  - All can be prefixed by “all_” for recursive lookup
    - Ex: all_state_variable_read: return all the state variables read in internal calls
  - slithir_operations

Function - example 1

```
from slither import Slither
sl = Slither("0xdac17f958d2ee523a2206206994597c13d831ec7")
compilation_unit = sl.compilation_units[0]
contract = compilation_unit.get_contract_from_name("TetherToken")[0]
```

```
# Print all the state variables read by the totalSupply function
totalSupply = contract.get_function_from_signature("totalSupply()")
print([str(v) for v in totalSupply.state_variables_read])
```

Function - example 1

```
% python test.py
['_totalSupply', 'deprecated', 'upgradedAddress']
```

Function - example 2

```
[..]

transfer = contract.get_function_from_signature("transfer(address,uint256)")

# Print all the state variables read by the transfer function
print([str(v) for v in transfer.state_variables_read])
# Print all the state variables read by the transfer function and its internal calls
print([str(v) for v in transfer.all_state_variables_read])
```

Function - example 2

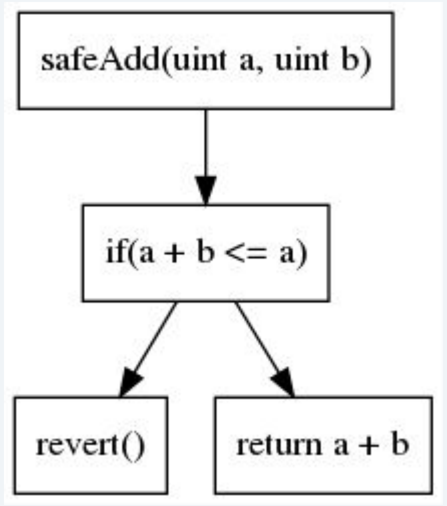
```
% python test.py

['deprecated', 'isBlackListed', 'upgradedAddress']

['owner', 'basisPointsRate', 'deprecated', 'paused', 'isBlackListed', 'maximumFee', 'upgradedAddress', 'balances']
```

```
function transfer(address _to, uint _value) public whenNotPaused {
    require(!isBlackListed[msg.sender]);
    if (deprecated) {
        return UpgradedStandardToken(upgradedAddress).transferByLegacy(msg.sender, _to, _value);
    } else {
        return super.transfer(_to, _value);
    }
}
```

Control flow graph

- **CFG**
- **Common code representation**

```
function safeAdd(uint256 a, uint256 b) ...
    if (a + b <= a) {
        revert();
    }
    return a + b;
}
```



Control flow graph

- core/cfg/node.py
- **If order does not matter**
  - for node in function.nodes
- **If order matters**
  - Walk through the nodes

```
def visit_node(node: Node, visited: List[Node]):

    if node in visited:
        return
    visited += [node]

    # custom action
    for son in node.sons:
        visit_node(son, visited)
```

Control flow graph

- **If need to iterate more than once:**
    - Bound the iteration X times
    - Create a fix-point - abstract interpretation style analysis

Only for advanced usages

SlithIR

SlithIR

- **Slither Intermediate Representation**
    - Solidity -> Human usage
    - SlithIR -> Code analysis usage

SlithIR

- **Less than 40 instructions**
- **Linear IR (no jump)**
- **Based on Slither CFG**
- **Flat IR**
- **Code transformation/simplification**
    - Ex: remove of ternary operator

SlithIR Instructions

- **Binary/Unary**
  - `LVALUE = RVALUE + RVALUE`
  - `LVALUE = ! RVALUE`
  - `...`
- **Index**
  - `REFERENCE -> LVALUE [ RVALUE ]`

SlithIR Instructions

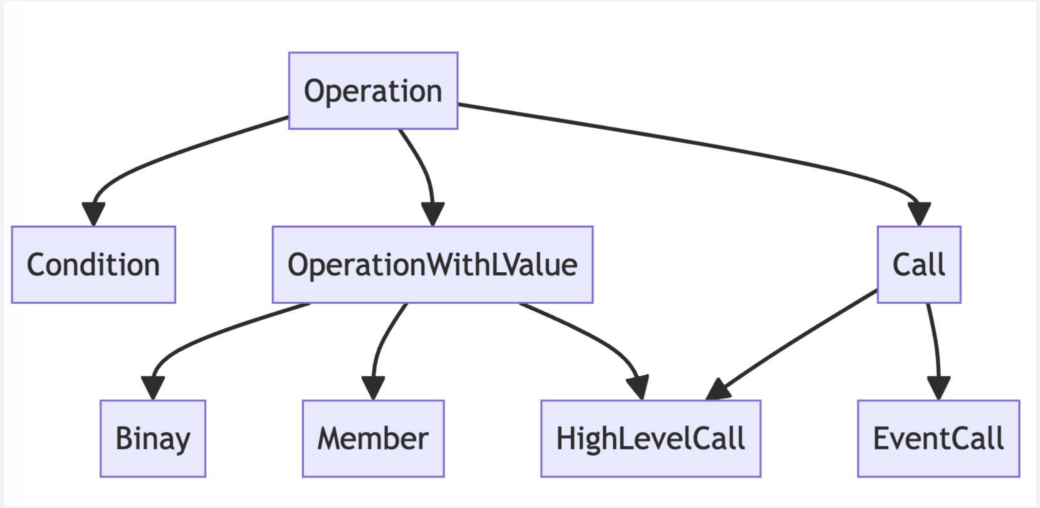
- **Member**
  - `REFERENCE -> LVALUE . RVALUE`
- **New**
  - `LVALUE = NEW_ARRAY ARRAY_TYPE DEPTH`
  - `LVALUE = NEW_CONTRACT CONSTANT`
  - `LVALUE = NEW_STRUCTURE STRUCTURE`

note: no new_structure operator in Solidity

SlithIR Instructions

```
Expression: allowance[_from][msg.sender] -= _value

        IRs:

            REF_1 -> allowance[_from]

            REF_2 -> REF_1[msg.sender]

            REF_2 -= _value
```

SlithIR hierarchie


$$\begin{tikzcd}
 & \text{Operation} \arrow[dl] \arrow[d] \arrow[drr] & & \\
\text{Condition} & \text{OperationWithLValue} \arrow[dl] \arrow[d] \arrow[dr] & & \text{Call} \arrow[dl] \arrow[d] \\
\text{Binay} & \text{Member} & \text{HighLevelCall} & \text{EventCall}
\end{tikzcd}$$

SlithIR - Cheatsheet

- slither/slithir
- **Check if an operation is of a type:**
  - isinstance(ir, TYPE)
  - Ex: isinstance(ir, Call)
- **Check if the operation is an addition**
  - isinstance(ir, Binary) & ir.type == BinaryType.ADDITION
- **Check if the operation is a call to MyContract**
  - isinstance(ir, HighLevelCall) & ir.destination == MyContract
- **Every IR operation has its own methods**

SlithIR - example

```python
from slither import Slither

sl = Slither("0xdac17f958d2ee523a2206206994597c13d831ec7")

compilation_unit = sl.compilation_units[0]

contract = compilation_unit.get_contract_from_name("TetherToken")[0]

totalSupply = contract.get_function_from_signature("totalSupply()")
```

```python
# Print the external call made in the totalSupply function

for ir in totalSupply.slithir_operations:

    if isinstance(ir, HighLevelCall):

        print(f"External call found {ir} ({ir.node.source_mapping})")
```

SlithIR - example

```
% python test.py

External call found HIGH_LEVEL_CALL, [...]   (...TetherToken.sol#339)
```

SlithIR Advanced

- **SSA (Static Single Assignment) support**
  - Include state variables
  - Precise data dependency analysis
- **Alias analysis on storage references**
  - Allow analysis of complex codebase

Only for advanced usages

Data dependency

Data dependency

- **Is** a **dependent on** b **?**

```
function f(uint a) internal{
    uint b = a + 10;
    if(condition){
        c = b;
    }
```

Only for advanced usages

Data dependency

- **Is a dependent on b ?**
- **Can `c` be controlled by the user?**

```
function f(uint a) internal{
    uint b = a + 10;
    if(condition){
        c = b;
    }
```

Only for advanced usages

Data dependency

- [slither/analyses/data_dependency/data_dependency.py]()
- **is_dependent(target, source, context)**
  - context: [Contract | Function]
- **is_tainted(target, source)**
  - Shortcut for is_dependent against “controlled” variables
    - functions parameters
    - msg.sender, msg.data, …

Only for advanced usages

Data dependency - Context

```
contract MyContract{

    uint var_1;
    uint var_2;

    function direct_set(uint i) public {
        var_1 = i;
    }

    function indirect_set() public {
        var_2 = var_1;
    }

}
```

- **Context “indirect_set” (function)**
  - var_2 dependent of var_1
- **Context MyContract (contract)**
  - var_2 dependent of var_1, **i**

Data dependency - example

```python
sl = Slither("test.sol")


myContract = sl.get_contract_from_name("MyContract")[0]

var_1 = myContract.variables_as_dict["var_1"]

var_2 = myContract.variables_as_dict["var_2"]

indirect_set = myContract.get_function_from_signature("indirect_set()")

print(f"{var_2} is controlled by users in the context of {indirect_set}: {is_tainted(var_2, indirect_set)}")

print(f"{var_2} is controlled by users in the context of {myContract}: {is_tainted(var_2, myContract)}")
```

Data dependency - example

```
% python dep.py
var_2 is controlled by users in the context of indirect_set: False
var_2 is controlled by users in the context of MyContract: True
```

```
contract MyContract{

    uint var_1;
    uint var_2;

    function direct_set(uint i) public {
        var_1 = i;
    }

    function indirect_set() public {
        var_2 = var_1;
    }

}
```

Where to start

Where to start

- ***secure-contracts.com***
  - *program-analysis/slither*
  - *Base API + exercises*
- ***Demo project***
  - *git clone git@github.com:crytic/slither.git*
  - cd slither/tools/demo/
    - Default folder with argument parsing + Slither object creation
- **Read slither/detectors code**
  - APIs' usage

Conclusion

Slither

- **Open source framework to build custom analysis**
- **Try it out:**
  - Build a bot that from an address shows all the functions without onlyOwner that can transfer assets
  - Build a web interface that shows the storage layout on a deployed contract
  - Build a tool that given a function, shows all the locations where it can revert
- **Need help?**
  - Github (issues , discussions)
  - Slack (https://slack.empirehacking.nyc/, #ethereum)
- **Hackathon: https://github.com/crytic/ethdam**