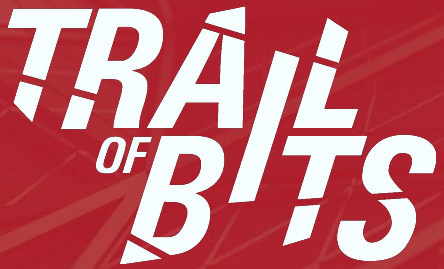


Slither: A Static Analysis Framework for Smart Contracts

EthCC 2019

Who am I?

- **Josselin Feist (josselin@trailofbits.com, @Montyly)**

- **Trail of Bits: trailofbits.com**
  - We help organizations build safer software
  - R&D focused: we use the latest program analysis techniques
    - https://github.com/trailofbits/manticore
    - https://github.com/trailofbits/echidna/
    - https://github.com/trailofbits/ethersplay

Plan

- What is Slither
- What are Slither applications
- Slither internals
- Conclusion and roadmap

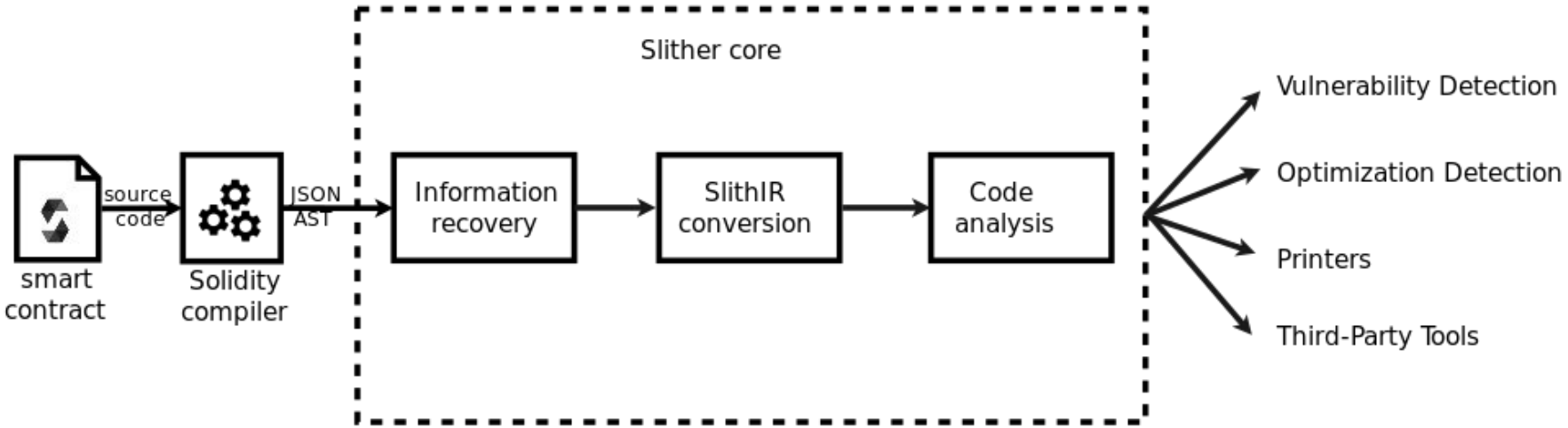
Slither

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

https://github.com/trailofbits/slither

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

Slither


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

Vulnerability Detection

Vulnerability Detection

- **~30 public vulnerability detectors**
- **From critical issues:**
  - Reentrancy,
  - Shadowing,
  - Uninitialized variables,
  - ...
- **To informational issues**
  - Naming convention
  - Old solc versions,
  - ...

Vulnerability Detection

```
tob:$ catc uninitialized.sol
pragma solidity ^0.5.5;

contract Uninitialized{
    address payable destination;

    function buggy() external{
        destination.transfer(address(this).balance);
    }
}
tob:$ slither uninitialized.sol
INFO:Detectors:
Uninitialized.destination (uninitialized.sol#4) is never initialized. It is used in:
        - buggy (uninitialized.sol#6-8)
Reference: https://github.com/trailofbits/slither/wiki/Detectors-Documentation#uninitialized-state-variables
INFO:Slither:uninitialized.sol analyzed (1 contracts), 1 result(s) found
tob:$
```

https://asciinema.org/a/eYrdWBvasHXelpDob4BsNi6Qg

Vulnerability Detection

Uninitialized state variables

Configuration

- Check: `uninitialized-state`
- Severity: `High`
- Confidence: `High`

Description

Uninitialized state variables.

Exploit Scenario:

```
contract Uninitialized{
    address destination;

    function transfer() payable public{
        destination.transfer(msg.value);
    }
}
```

Bob calls `transfer`. As a result, the ethers are sent to the address 0x0 and are lost.

Recommendation

Initialize all the variables. If a variable is meant to be initialized to zero, explicitly set it to zero.

https://github.com/trailofbits/slither/wiki/Detectors-Documentation

Vulnerability Detection

- **List of public detectors:**
  https://github.com/trailofbits/slither/#detectors

- **Private detectors include:**
  - Race conditions
  - Incorrect tokens manipulation
  - ...

Vulnerability Detection

- Fast (1-2 seconds)
- No configuration
- Low # false alarms
- Easy integration into CI (Truffle)

Optimization Detection

Code Optimization Detection

- **Detect optimizations that are missed by solc**
- **Examples:**
  - Variables that should be constant
  - Functions that should be external

Code Understanding

Code Understanding

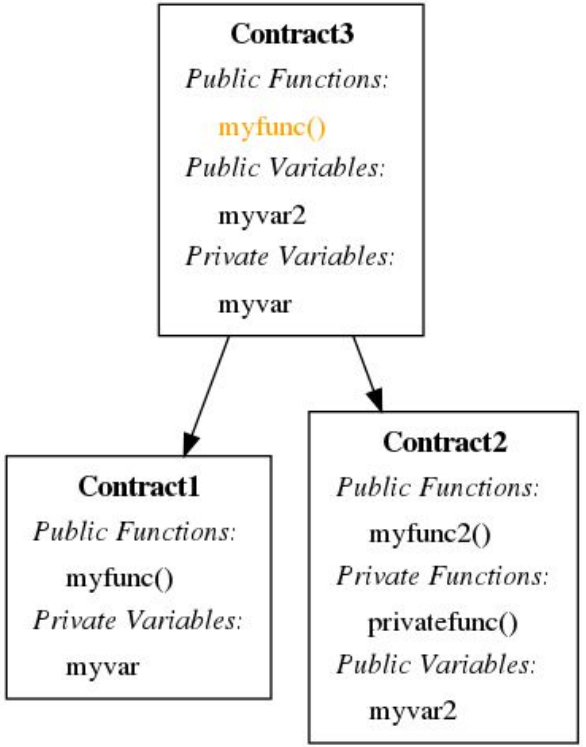
- **Printers: visual representations**
- **Examples:**
  - Graph-based representations (inheritance graph, CFG, call-graph)
  - Read/Write/Call summary
  - Access control summary
  - Human-readable summary (code complexity, minting restrictions, ..)
- https://github.com/trailofbits/slither/#printers

Printers: Inheritance Graph

```
contract Contract1{
   uint myvar;
   function myfunc() public{}
}

contract Contract2{
   uint public myvar2;

   function myfunc2() public{}
   function privatefunc() private{}
}
contract Contract3 is Contract1, Contract2{
   function myfunc() public{} // override myfunc
}
```



Generic Static Analysis Framework

Assisted Code Review

- **Library for tooling**
  - slither-check-upgradability: Help to review delegatecall proxy contract
  - slither-find-paths: Find all the paths that can reach a given function

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

Assisted Code Review

Ex: What functions can modify a state variable:

```
slither = Slither('function_writing.sol')
contract = slither.get_contract_from_name('Contract')
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 Engine

- **Input: solc AST**

- **Use refinement parsing (joern)**
  - Parse through multiple stages/layers

Slither Layers

- **Contracts**
  - Inheritance, state variables, functions
- **Functions**
  - Attributes, CFG
- **Control Flow Graphs**
  - Nodes
- **Nodes**
  - Expressions as AST -> SlithIR

Code Analysis

- **Read/Write of variables**
  - Level: node/function/contract
- **Protected functions**
  - What functions need ownership?
- **Data dependency**
  - What variable's value can influence `myOwner` variable?

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 Features

- **SSA (Static Single Assignment) support**
  - Include state variables
  - Precise data dependency analysis

- **Alias analysis on storage references**
  - Allow analysis of complex codebase

Taint Example

```
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;
    }

}
```

direct_set

- var_1 depends on i

Indirect_set

- var_2 depends on var_1

MyContract:

- var_1 depends on i
- var_2 depends on var_1, i

Conclusion

Conclusion

- **Vulnerability and optimization detection**
  - Fast and precise
  - No configuration
  - CI support
- **Code review**
  - In-depth information about the codebase
- **A foundation for research**
  - Generic library for static analysis

Roadmap

- **More detectors!**
- **Improve developer integration**
  - Visual Studio plugin (90)
  - slither-format: automatic patching (150)
- **New language support**
  - Vyper (39)
- **SlithIR improvements**
  - Formal semantics
  - Symbolic Computation/Symbolic Execution/Abstract Interpretation

Slither

- https://github.com/trailofbits/slither

- **Crytic: SaaS to ensure safe contracts**
  - Includes Slither private detectors and formal verification
  - For more information: Dan Guido (dan@trailofbits.com)

- **Need Help?**
  - Slack: https://empireslacking.herokuapp.com (#ethereum)
  - Office Hours: free 1-hour consultation on Hangouts every two weeks