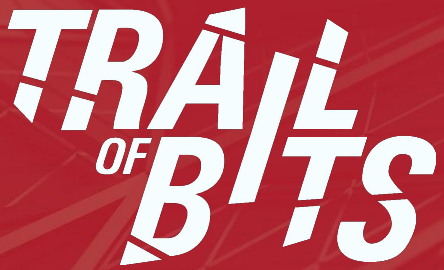


Building a Practical Static Analyzer for Smart Contracts

Rigorous Methods for Smart Contracts

Who am I?

- **Josselin Feist**
  - josselin@trailofbits.com, @Montyly
  - github.com/montyly/publications

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

Plan

- What is Slither
- How it works
- Industry & academic impacts
- Conclusion

Slither

Slither

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

https://github.com/crytic/slither

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

Features

- 70+ public detectors
- Support Solidity from 0.4 to 0.8
- Support the compilation frameworks out of the box
  - Hardhat, truffle, dapp, embark, ...
- Support deployed contracts through etherscan/bscan/...

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:$
```

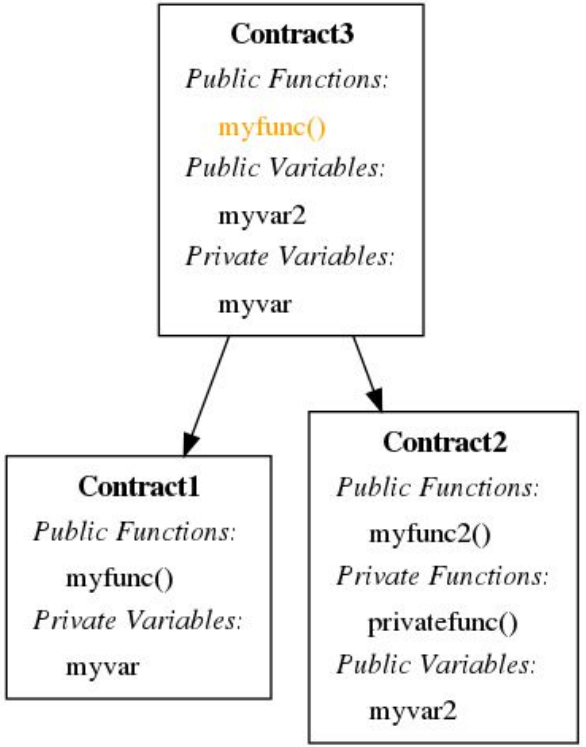
https://asciinema.org/a/eYrdWBvasHXelpDob4BsNi6Qg

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
}
```



Inbuilt tools

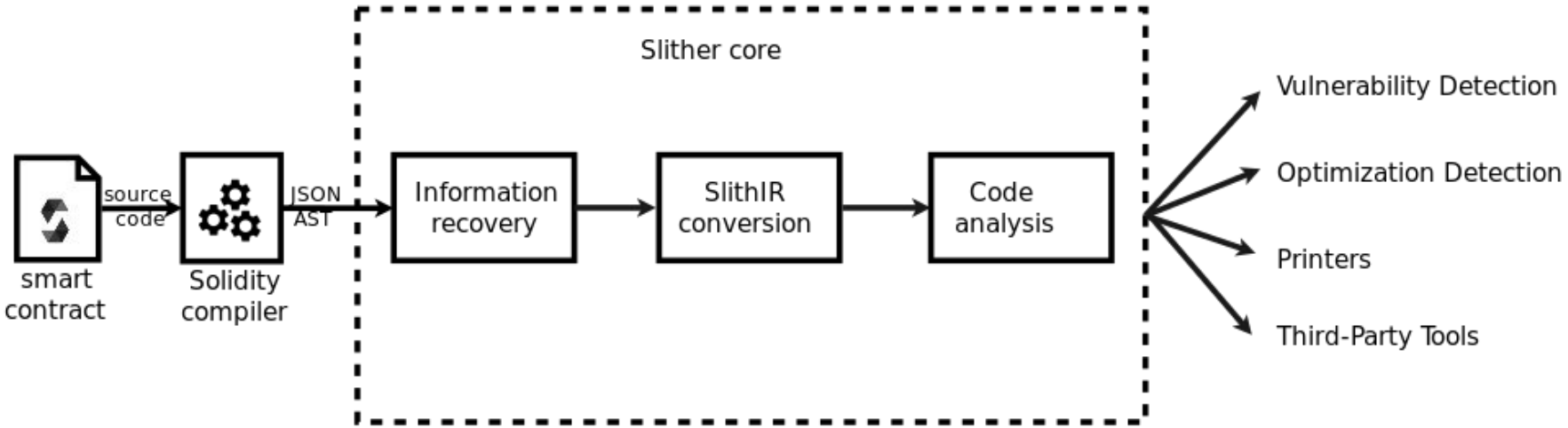
- slither-check-erc
  - Check for ERC specification conformance
- slither-check-upgradability
  - Help to review delegatecall proxy contract
- slither-prop
  - Automatic unit test and property generation
- slither-simil
  - ML based code similarity
- Echidna's integration
  - Helping fuzzing through static information

Custom scripts

- **Python API to help during a code review**
  - Inspect contract information
  - Including data dependency/taint analysis
- **Ex: creating a whitelist of protected functions**
  - Every function must have `onlyOwner`, or being whitelisted
  - https://github.com/trailofbits/publications/blob/master/reviews/AdvancedBlockchain.pdf

How it works

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

SlithIR

- **Codebase information from solc's AST**
  - Contracts, functions, CFG
- **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: removal 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 SSA

- **SSA (Static Single Assignment) form**
  - A variable is assigned only one time
  - Needed for precise data dependency analysis
  - Usually, ϕ indicates multiple definitions of a variable

```
a = 0
if(){
    a = b;
}

a = a + 1;
```

```
a_0 = 0
if(){
    a_1 = b_0;
}

a_2 = ϕ(a_0, a_1)
a_3 = a_2 + 1;
```

SlithIR SSA

- **SlithIR SSA features**
  - Include:
    - State variables
    - Alias analysis on *storage* reference pointers
  - Inter-procedural
    - Track internal calls
  - Inter-transactional
    - Take in consideration the state-machine aspect of smart contracts

Data dependency

```
uint my_state_A;

uint my_state_B;

function direct_set(uint input) public {
    my_state_A = input;
}


function indirect_set() public {
     my_state_B = my_state_A;
}
```

Data dependency

```
uint my_state_A;

uint my_state_B;

function direct_set(uint input) public {
    my_state_A = input;
}


function indirect_set() public {
     my_state_B = my_state_A;
}
```

Dependencies:

- my_state_A depends on input
- my_state_B depends on my_state_A
  - But also input?

SSA Inter-Transactional Example

```
uint my_state_A;

uint my_state_B;

function direct_set(uint input) public {
    my_state_A = input;
}


function indirect_set() public {
     my_state_B = my_state_A;
}
```

```
my_state_A_0;
my_state_B_0;

direct_set(uint input_0):

    my_state_A_1 := input_0


indirect_set():

    my_state_A_2 := ϕ(my_state_A_0,
                      my_state_A_1)

    my_state_A_2 := my_state_A_2
```

SlithIR: Code Analysis

- **Data dependency**
  - Pre-computed, free for analyses
  - Level: function/contract
- **Read/Write of variables**
  - Level: node/function/contract
- **Protected functions**
  - What functions need ownership?

Industry & academic impacts

Industry impact

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
E&Y's Nightfall	Missing return value	May 2020
E&Y's Nightfall	Empty return value	May 2020
DefiStrategies	Modifier can return the default value	May 2020
DefiStrategies	Dangerous strict equality allows the contract to be trapped	May 2020
DOSnetwork	Abi `encodedPacked` collision	May 2020
EthKids	`msg.value` is used two times to compute a price	May 2020
HQ20	Reentrancy	May 2020
Dloop	Dangerous `block.timestamp` usage	Jun 2020
Atomic Loans	Uninitialized state variable	Jul 2020
Atomic Loans	State variable shadowing	Jul 2020
Atomic Loans	Reentrancy	Jul 2020
Amp	Duplicate contract name	Aug 2020
PerlinXRewards	Multiple reentrancies	Aug 2020
Linkswap	Lack of return value check	Nov 2020
Linkswap	Uninitialized state variable	Nov 2020
Cryptex	Lack of return value check	Nov 2020
Hermez	Reentrancy	Nov 2020
Unoswap	Contract locking ethers	Nov 2020
Idle	Dangerous divide before multiply operations	Dec 2020
RariCapital	Lack of return value check	Dec 2020
RariCapital	Uninitialized state variable	Dec 2020
wfil-factory	Reentrancy	Dec 2020
Origin Dollar	Reentrancy	Jan 2021
Origin Dollar	Variable shadowing	Jan 2021
OriginTrait	Reentrancy	Jan 2021
AlphaHomoraV2	Dangerous divide before multiply operations	Jan 2021
Mimo Defi	Lack of return value check	Jan 2021

github.com/crytic/slither/blob/master/trophies.md

Academic impact

Title	Usage	Authors	Venue
ReJection: A AST-Based Reentrancy Vulnerability Detection Method	AST-based analysis built on top of Slither	Rui Ma, Zefeng Jian, Guangyuan Chen, Ke Ma, Yujia Chen	CTCIS 19
MPro: Combining Static and Symbolic Analysis forScalable Testing of Smart Contract	Leverage data dependency through Slither	William Zhang, Sebastian Banescu, Leodardo Pasos, Steven Stewart, Vijay Ganesh	ISSRE 2019
ETHPLOIT: From Fuzzing to Efficient Exploit Generation against Smart Contracts	Leverage data dependency through Slither	Qingzhao Zhang, Yizhuo Wang, Juanru Li, Siqi Ma	SANER 20
Verification of Ethereum Smart Contracts: A Model Checking Approach	Symbolic execution built on top of Slither's CFG	Tam Bang, Hoang H Nguyen, Dung Nguyen, Toan Trieu, Tho Quan	IJMLC 20
Smart Contract Repair	Rely on Slither's vulnerabilities detectors	Xiao Liang Yu, Omar Al-Bataineh, David Lo, Abhik Roychoudhury	TOSEM 20
Demystifying Loops in Smart Contracts	Leverage data dependency through Slither	Ben Mariano, Yanju Chen, Yu Feng, Shuvendu Lahiri, Isil Dillig	ASE 20
Trace-Based Dynamic Gas Estimation of Loops in Smart Contracts	Use Slither's CFG to detect loops	Chunmiao Li, Shijie Nie, Yang Cao, Yijun Yu, Zhenjiang Hu	IEEE Open J. Comput. Soc. 1 (2020)

github.com/crytic/slither/blob/master/README.md#external-publications

Conclusion

Conclusion

- **Slither: a general static analyzer for smart contracts**
- **Researchers can leverage its engineering work**
  - Inbuilt analyses
  - Multiple solidity versions and frameworks support
  - Maintained codebase

Conclusion

- **Try our tutorials in** building-secure-contracts
  - Got an research idea? Contact us for help
    - Slack: https://empireslacking.herokuapp.com (#ethereum)
    - josselin@trailofbits.com
- **Crytic prize: $10k for best open academic researches**
  - Include Slither, Echidna, Tealer, Manticore, ...