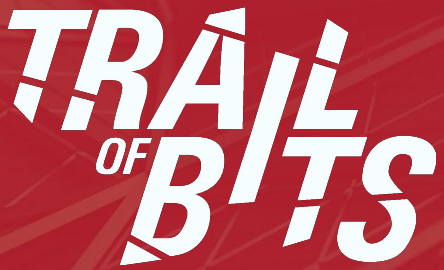


Write Better Smart Contracts By Checking Them With Slither's Python API

Troy Sargent

Background

- **Security Engineer at Trail of Bits**
  - Work on smart contracts and blockchains
  - Contributor to Slither
  - @0xalpharush / troy.sargent@trailofbits.com
- **Trail of Bits:**
  - We help organizations build high assurance software
  - R&D focused: we use the latest program analysis techniques
    - Slither, Echidna, Tealer, Amarna, Circomspect …

Plan

- What is Slither
- How it works
- How to use it
- Conclusion

Slither

Slither

- **Static analysis framework for smart contracts**
  - Vulnerability detection
  - Optimization detection
  - Assisted code review

https://github.com/crytic/slither

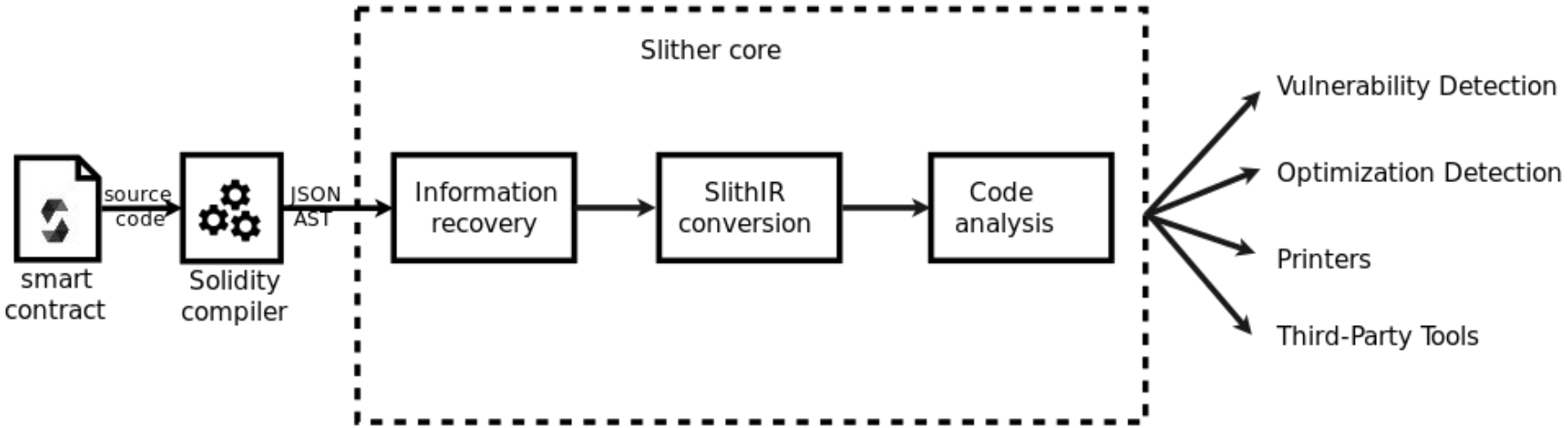
```
pip3 install -u slither-analyzer
```

Features

- 80+ detectors
- Supports Solidity from 0.4 to 0.8
- Supports compilation frameworks out of the box
  - Hardhat, brownie, foundry, ...
- Supports deployed contracts through Etherscan

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

Demo Contract

```
contract AlarmClock {
    address owner;
    uint gasBonus = 5000;

    modifier onlyOwner() {
        require(msg.sender == owner);
        _;
    }

    constructor () {
        owner = msg.sender;
    }

    function updateGasBonus(uint newGasBonus) external onlyOwner {
        gasBonus = newGasBonus;
    }

    function cancelAndRefund(address who) external {
        uint startGas = gasleft();
        // SOME EXPENSIVE CALL
        who.call(msg.data);

        uint gasUsed = startGas - gasleft() + gasBonus;

        uint rewardOwed = gasUsed * tx.gasprice;

        payable(msg.sender).transfer(rewardOwed);
    }

}
```

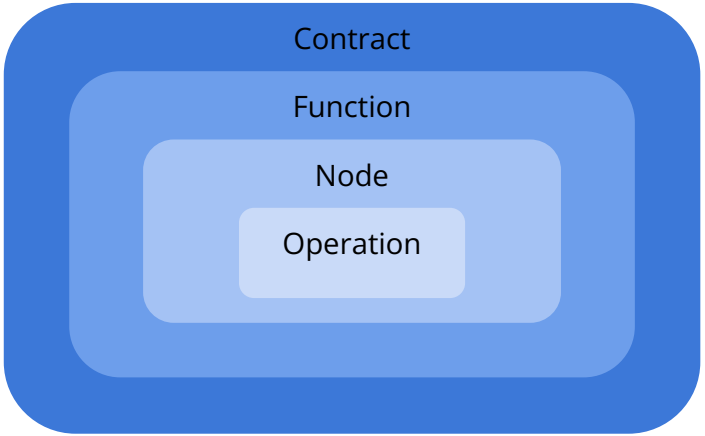
Alarm Clock:

- Has an owner
- Has privileged functions
- Has refund mechanism
- Has user input

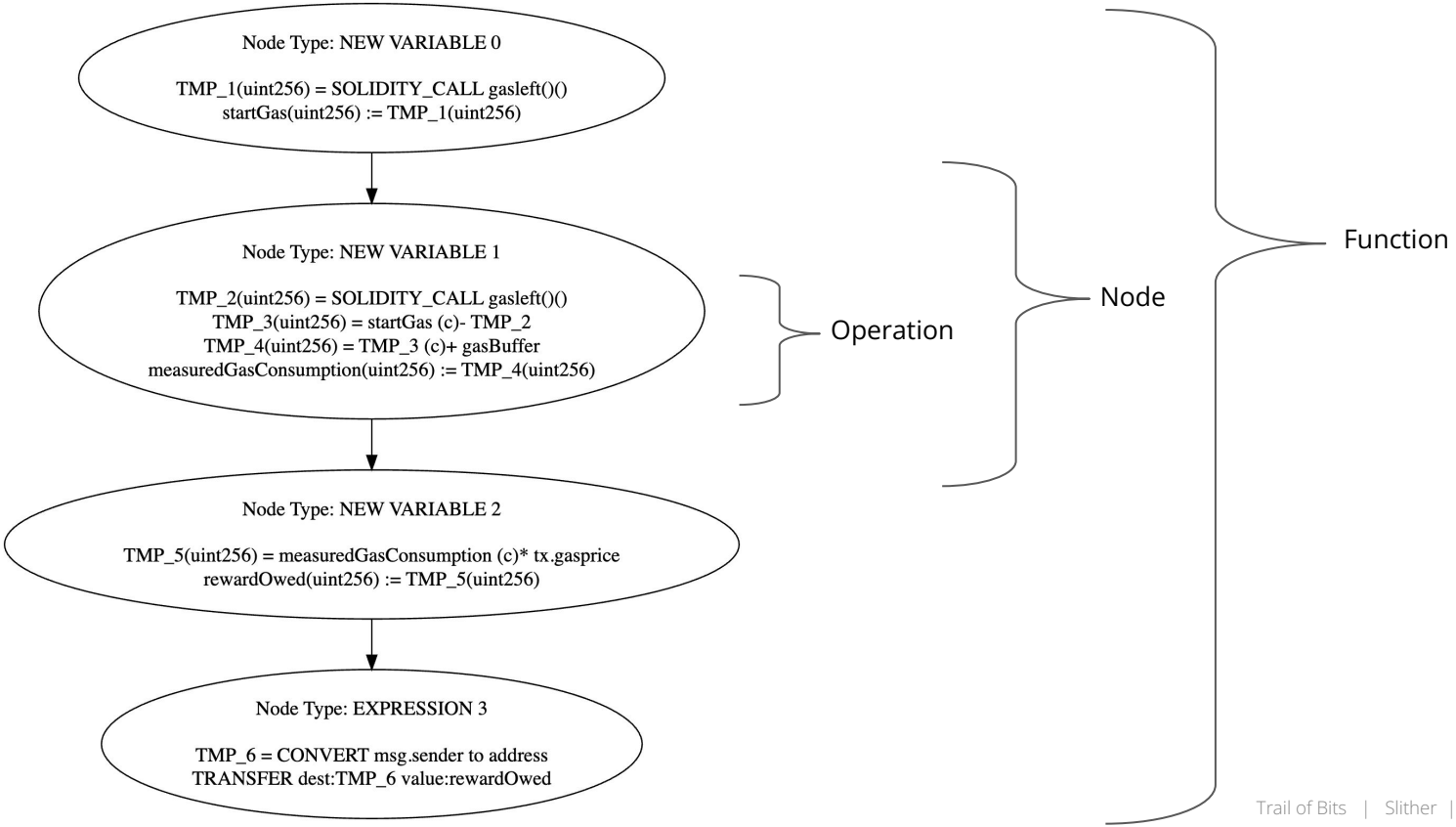
API Usage

The Basics

```python
from slither import Slither

analyzer = Slither("Contract.sol")

contracts: List[Contract] = analyzer.contracts

functions: List[Function] = contracts[0].functions

nodes: List[Node] = functions[0].nodes

operations: List[Operation] = node[0].irs
```

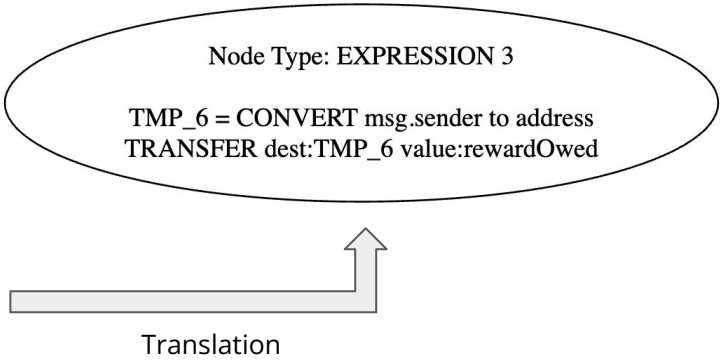


Demo Contract (slithIR and CFG)



Demo Contract (data dependency)

```
function cancelAndRefund(address who) external {
    uint startGas = gasleft();
    // SOME EXPENSIVE CALL
    uint measuredGasConsumption = startGas - gasleft() + gasBuffer;

    uint rewardOwed = measuredGasConsumption * tx.gasprice;

    payable(msg.sender).transfer(rewardOwed);
}
```



Root cause: a sensitive operation uses user-controlled input

Can the user manipulate refunds?

Check that all **Transfer** operations are not tainted by user input:

```
slither = Slither("AlarmClock.sol")
alarm = slither.get_contract_from_name("AlarmClock")[0]
for func in alarm.functions:
    for node in func.nodes:
        for operation in node.irs:
            if isinstance(operation, Transfer) and is_dependent(
                operation.call_value, SolidityVariableComposed("tx.gasprice"), func
            ):
                print(f"{node.expression} uses tainted input, tx.gasprice, in {func}")
```

Let's run it!

```
$ python3 detect.py AlarmClock.sol

"address(msg.sender).transfer(rewardOwed) uses tainted input, tx.gasprice, in cancelAndRefund"
```

Conclusion

Conclusion

- **Slither: a static analyzer for smart contracts**
- **Developers can leverage its powerful API**
  - Built-in analyses
  - Supports most solidity versions and frameworks
  - Actively maintained codebase

Conclusion

- **Try our tutorials and exercises in** building-secure-contracts
- **Have questions or ideas? Reach out**
  - @0xalpharush / troy.sargent@trailofbits.com
  - Slack: https://empireslacking.herokuapp.com
- **Find these slides:**
  https://github.com/trailofbits/publications#blockchain