Analysis of Smart Contract vulnerability and extension of Oyente Tool

Team Guide: **Dr. Kunwar Singh**

Submitted By

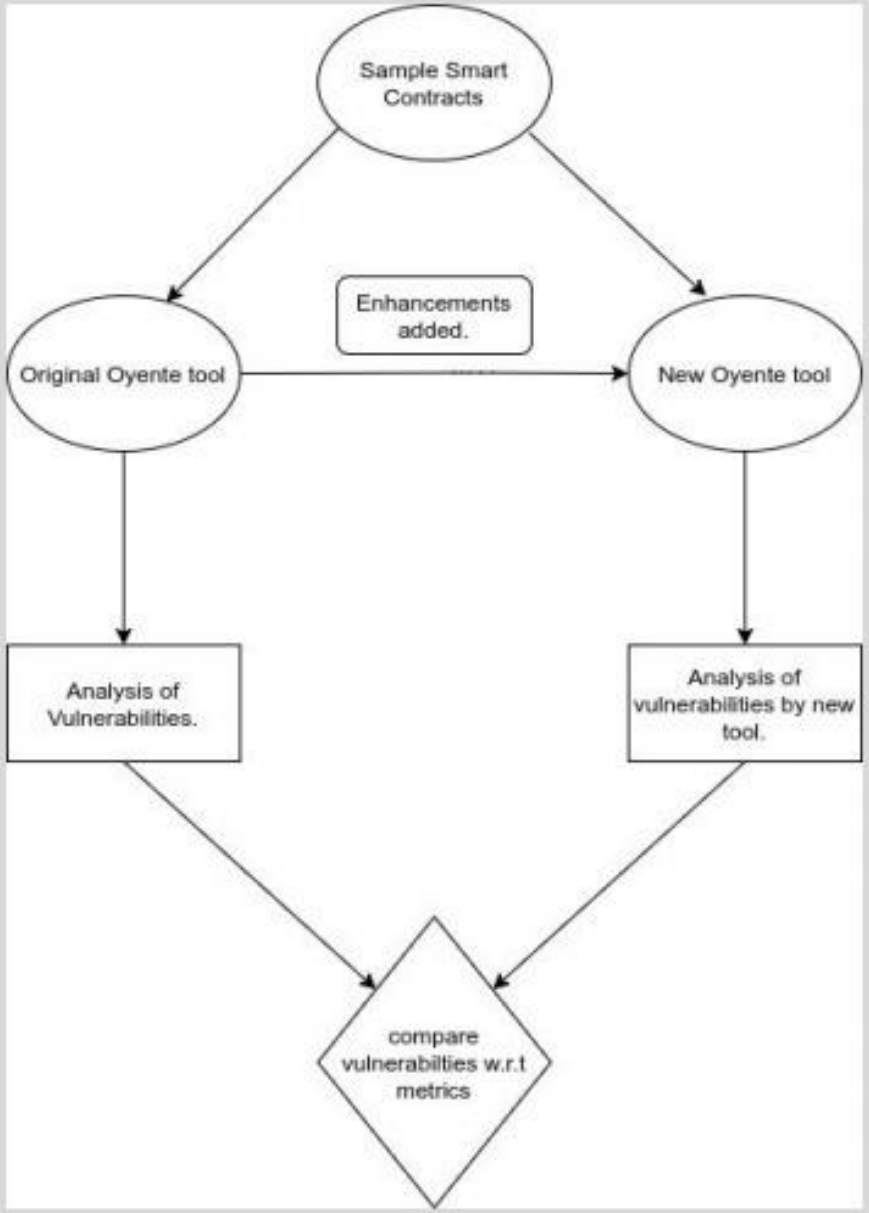
Ayush Sharma (106117018)

Mayank Garg (106117048)

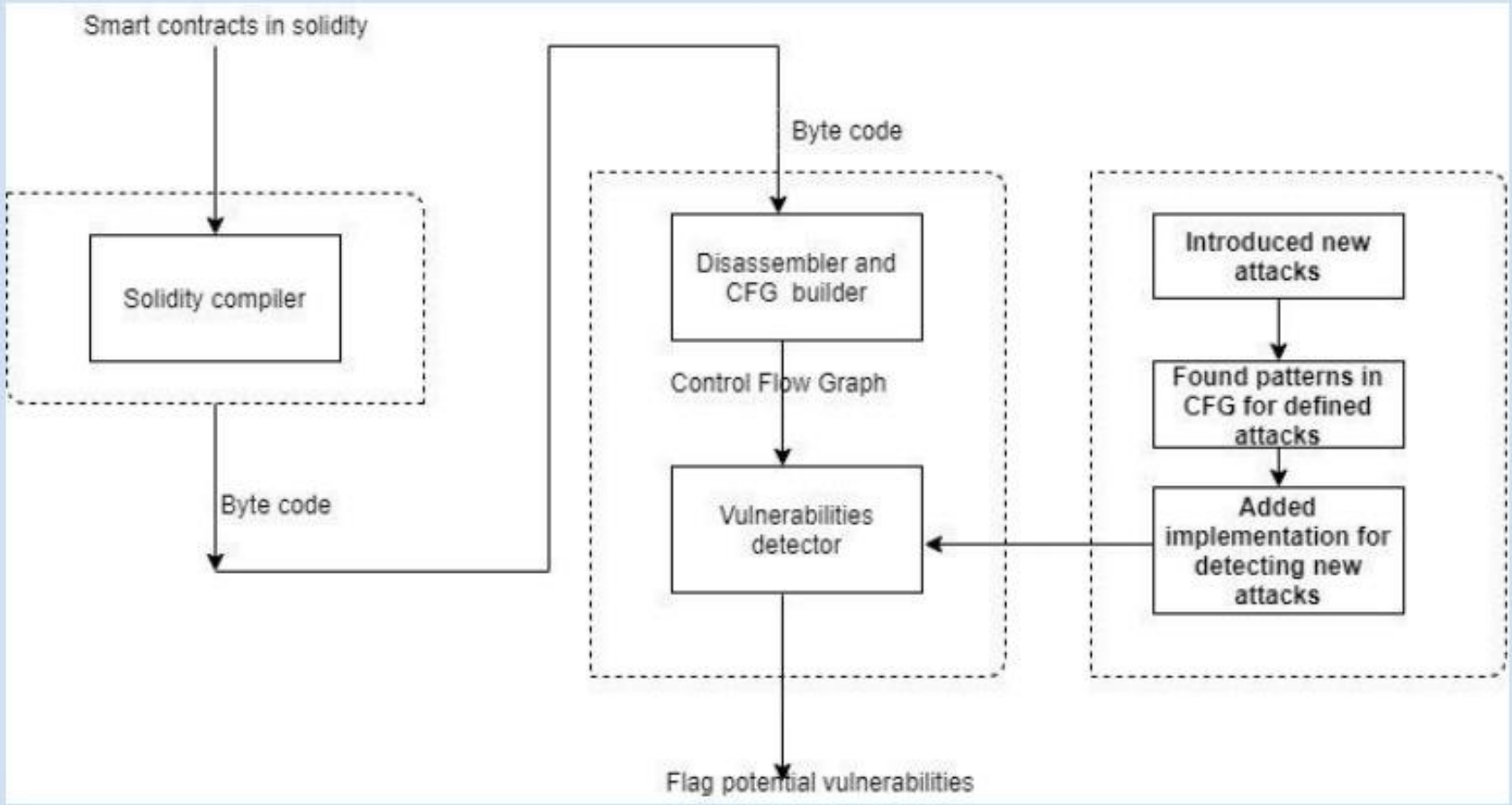
Vulnerabilities Detection in Oyente Tool

Old Oyente Tool	New Oyente Tool
• Integer Underflow	• Integer Underflow
• Integer Overflow	• Integer Overflow
• Call Stack depth attack	• Call Stack depth attack
• Transaction ordering Dependence	• Transaction ordering Dependence
• Timestamp dependency	• Timestamp dependency
• Re-Entrancy vulnerability	• Re-Entrancy vulnerability
• Parity Multisig Bug	• Parity Multisig Bug
	• **Contract Referencing**
	• **Efficient Transaction ordering dependence**
	• **Tx_origin**

Flow of the project



Implementation of the project



Old Oyente Tool Result

```
/usr/bin/python2.7 /home/shaz/Downloads/oyente/oyente.py
WARNING:root:You are using solc version 0.4.24, The latest supported version is 0.4.19
INFO:root:contract tx_origin.sol:TxUserWallet:
INFO:symExec:   ============ Results ===========
INFO:symExec:     EVM Code Coverage:             94.4%
INFO:symExec:     Integer Underflow:             False
INFO:symExec:     Integer Overflow:              False
INFO:symExec:     Parity Multisig Bug 2:         False
INFO:symExec:     Callstack Depth Attack Vulnerability:  False
INFO:symExec:     Transaction-Ordering Dependence (TOD): False
INFO:symExec:     Timestamp Dependency:          False
INFO:symExec:     Re-Entrancy Vulnerability:         False
INFO:symExec:   ====== Analysis Completed ======

Process finished with exit code 0
```

Newly Added Vulnerabilities

1. Contract Referencing Vulnerability

```
import "referenced_one.sol";
import "referenced_two.sol";

contract Reference {
    referenced_one x;
    function calculate(){
        x.sum();
    }
    referenced_two y;
    function operate(){
        y.difference();
    }
}
```

Referencing 1st external contract

```
contract referenced_two {
    function difference(){
        int y=5;
        int z=10;
        int x=z*y;
    }
}
```

Referencing 2nd external smart contract

```
contract referenced_one {
    function sum(){
        int y=5;
        int z=10;
        int x=z-y;
    }
}
```

Calling a function of external contract

An external contract might not give us the proper results that we need to get. So whenever we call an external contract we need to create an instance of it first so that it can't be manipulated.

Disassembled Code

This is the code generated by disassembler, in which we need to find the pattern for the vulnerability.

```
('0', 'PUSH', '1', '80')
('2', 'PUSH', '1', '40')
('4', 'MSTORE', '', '')
('5', 'PUSH', '1', '04')
('7', 'CALLDATASIZE', '', '')
('8', 'LT', '', '')
('9', 'PUSH', '2', '004c')
('12', 'JUMPI', '', '')
('13', 'PUSH', '1', '00')
('15', 'CALLDATALOAD', '', '')
('16', 'PUSH', '29', '0100000000000000000000000000000000000000000000000000000000')
('46', 'SWAP', '1', '')
('47', 'DIV', '', '')
('48', 'PUSH', '4', 'ffffffff')
('53', 'AND', '', '')
('54', 'DUP', '1', '')
('55', 'PUSH', '4', '7159a618')
('60', 'EQ', '', '')
('61', 'PUSH', '2', '0051')
('64', 'JUMPI', '', '')
('65', 'DUP', '1', '')
('66', 'PUSH', '4', 'ca77ab65')
('71', 'EQ', '', '')
('72', 'PUSH', '2', '0068')
('75', 'JUMPI', '', '')
('76', 'JUMPDEST', '', '')
('77', 'PUSH', '1', '00')
('79', 'DUP', '1', '')
('80', 'REVERT', '', '')
('81', 'JUMPDEST', '', '')
('82', 'CALLVALUE', '', '')
```

```
('82', 'CALLVALUE', '', '')
('83', 'DUP', '1', '')
('84', 'ISZERO', '', '')
('85', 'PUSH', '2', '005d')
('88', 'JUMPI', '', '')
('89', 'PUSH', '1', '00')
('91', 'DUP', '1', '')
('92', 'REVERT', '', '')
('93', 'JUMPDEST', '', '')
('94', 'POP', '', '')
('95', 'PUSH', '2', '0066')
('98', 'PUSH', '2', '007f')
('101', 'JUMP', '', '')
('102', 'JUMPDEST', '', '')
('103', 'STOP', '', '')
('104', 'JUMPDEST', '', '')
('105', 'CALLVALUE', '', '')
('106', 'DUP', '1', '')
('107', 'ISZERO', '', '')
('108', 'PUSH', '2', '0074')
('111', 'JUMPI', '', '')
('112', 'PUSH', '1', '00')
('114', 'DUP', '1', '')
('115', 'REVERT', '', '')
('116', 'JUMPDEST', '', '')
('117', 'POP', '', '')
('118', 'PUSH', '2', '007d')
('121', 'PUSH', '2', '011f')
('124', 'JUMP', '', '')
('125', 'JUMPDEST', '', '')
('126', 'STOP', '', '')
('127', 'JUMPDEST', '', '')
```

Disassembled code continued

```
('127', 'JUMPDEST', '', '')
('128', 'PUSH', '1', '01')
('130', 'PUSH', '1', '00')
('132', 'SWAP', '1', '')
('133', 'SLOAD', '', '')
('134', 'SWAP', '1', '')
('135', 'PUSH', '2', '0100')
('138', 'EXP', '', '')
('139', 'SWAP', '1', '')
('140', 'DIV', '', '')
('141', 'PUSH', '20', 'ffffffffffffffffffffffffffffffffffffffff')
('162', 'AND', '', '')
('163', 'PUSH', '20', 'ffffffffffffffffffffffffffffffffffffffff')
('184', 'AND', '', '')
('185', 'PUSH', '4', 'dd56653c')
('190', 'PUSH', '1', '40')
('192', 'MLOAD', '', '')
('193', 'DUP', '2', '')
('194', 'PUSH', '4', 'ffffffff')
('199', 'AND', '', '')
('200', 'PUSH', '29', '0100000000000000000000000000000000000000000000000000000000')
('230', 'MUL', '', '')
('231', 'DUP', '2', '')
('232', 'MSTORE', '', '')
('233', 'PUSH', '1', '04')
('235', 'ADD', '', '')
('236', 'PUSH', '1', '00')
('238', 'PUSH', '1', '40')
('240', 'MLOAD', '', '')
('241', 'DUP', '1', '')
('242', 'DUP', '4', '')
```

```
('241', 'DUP', '1', '')
('242', 'DUP', '4', '')
('243', 'SUB', '', '')
('244', 'DUP', '2', '')
('245', 'PUSH', '1', '00')
('247', 'DUP', '8', '')
('248', 'DUP', '1', '')
('249', 'EXTCODESIZE', '', '')
('250', 'ISZERO', '', '')
('251', 'DUP', '1', '')
('252', 'ISZERO', '', '')
('253', 'PUSH', '2', '0105')
('256', 'JUMPI', '', '')
('257', 'PUSH', '1', '00')
('259', 'DUP', '1', '')
('260', 'REVERT', '', '')
('261', 'JUMPDEST', '', '')
('262', 'POP', '', '')
('263', 'GAS', '', '')
('264', 'CALL', '', '')
('265', 'ISZERO', '', '')
('266', 'DUP', '1', '')
('267', 'ISZERO', '', '')
('268', 'PUSH', '2', '0119')
('271', 'JUMPI', '', '')
('272', 'RETURNDATASIZE', '', '')
('273', 'PUSH', '1', '00')
('275', 'DUP', '1', '')
('276', 'RETURNDATACOPY', '', '')
('277', 'RETURNDATASIZE', '', '')
('278', 'PUSH', '1', '00')
('280', 'REVERT', '', '')
```

Disassembled code continued

```
('280', 'REVERT', '', '')
('281', 'JUMPDEST', '', '')
('282', 'POP', '', '')
('283', 'POP', '', '')
('284', 'POP', '', '')
('285', 'POP', '', '')
('286', 'JUMP', '', '')
('287', 'JUMPDEST', '', '')
('288', 'PUSH', '1', '00')
('290', 'DUP', '1', '')
('291', 'SWAP', '1', '')
('292', 'SLOAD', '', '')
('293', 'SWAP', '1', '')
('294', 'PUSH', '2', '0100')
('297', 'EXP', '', '')
('298', 'SWAP', '1', '')
('299', 'DIV', '', '')
('300', 'PUSH', '20', 'ffffffffffffffffffffffffffffffffffffffff')
('321', 'AND', '', '')
('322', 'PUSH', '20', 'ffffffffffffffffffffffffffffffffffffffff')
('343', 'AND', '', '')
('344', 'PUSH', '4', '853255cc')
('349', 'PUSH', '1', '40')
('351', 'MLOAD', '', '')
('352', 'DUP', '2', '')
('353', 'PUSH', '4', 'ffffffff')
('358', 'AND', '', '')
('359', 'PUSH', '29', '0100000000000000000000000000000000000000000000000000000000')
('389', 'MUL', '', '')
('390', 'DUP', '2', '')
('391', 'MSTORE', '', '')
('392', 'PUSH', '1', '04')
```

```
('392', 'PUSH', '1', '04')
('394', 'ADD', '', '')
('395', 'PUSH', '1', '00')
('397', 'PUSH', '1', '40')
('399', 'MLOAD', '', '')
('400', 'DUP', '1', '')
('401', 'DUP', '4', '')
('402', 'SUB', '', '')
('403', 'DUP', '2', '')
('404', 'PUSH', '1', '00')
('406', 'DUP', '8', '')
('407', 'DUP', '1', '')
('408', 'EXTCODESIZE', '', '')
('409', 'ISZERO', '', '')
('410', 'DUP', '1', '')
('411', 'ISZERO', '', '')
('412', 'PUSH', '2', '01a4')
('415', 'JUMPI', '', '')
('416', 'PUSH', '1', '00')
('418', 'DUP', '1', '')
('419', 'REVERT', '', '')
('420', 'JUMPDEST', '', '')
('421', 'POP', '', '')
('422', 'GAS', '', '')
('423', 'CALL', '', '')
('424', 'ISZERO', '', '')
('425', 'DUP', '1', '')
('426', 'ISZERO', '', '')
('427', 'PUSH', '2', '01b8')
('430', 'JUMPI', '', '')
('431', 'RETURNDATASIZE', '', '')
('432', 'PUSH', '1', '00')
```

```
('432', 'PUSH', '1', '00')
('434', 'DUP', '1', '')
('435', 'RETURNDATACOPY', '', '')
('436', 'RETURNDATASIZE', '', '')
('437', 'PUSH', '1', '00')
('439', 'REVERT', '', '')
('440', 'JUMPDEST', '', '')
('441', 'POP', '', '')
('442', 'POP', '', '')
('443', 'POP', '', '')
('444', 'POP', '', '')
('445', 'JUMP', '', '')
('446', 'STOP', '', '')
```

Output

Here we can see in the result that Contract Referencing vulnerability has been flagged true by our Oyente tool.

```
shaz@ubuntu:~/Downloads/oyente$ python oyente.py -s reference.sol
WARNING:root:You are using solc version 0.4.24, The latest supported version is 0.4.19
INFO:root:contract reference.sol:Reference:
INFO:symExec:   ============ Results ===========
INFO:symExec:     EVM Code Coverage:                      92.3%
INFO:symExec:     Integer Underflow:                      False
INFO:symExec:     Integer Overflow:                       False
INFO:symExec:     Parity Multisig Bug 2:                  False
INFO:symExec:     Transaction Ordering Dependence Efficient  False
INFO:symExec:     TX.Origin Vulnerability  False
INFO:symExec:     Contract Referencing True
INFO:symExec:     Callstack Depth Attack Vulnerability:  False
INFO:symExec:     Transaction-Ordering Dependence (TOD): False
INFO:symExec:     Timestamp Dependency:                   False
INFO:symExec:     Re-Entrancy Vulnerability:              False
INFO:symExec:   ====== Analysis Completed ======
INFO:root:contract referenced_one.sol:referenced_one:
INFO:symExec:   ============ Results ===========
INFO:symExec:     EVM Code Coverage:                      98.3%
INFO:symExec:     Integer Underflow:                      False
INFO:symExec:     Integer Overflow:                       False
INFO:symExec:     Parity Multisig Bug 2:                  False
INFO:symExec:     Transaction Ordering Dependence Efficient  False
INFO:symExec:     TX.Origin Vulnerability  False
INFO:symExec:     Contract Referencing  False
INFO:symExec:     Callstack Depth Attack Vulnerability:  False
INFO:symExec:     Transaction-Ordering Dependence (TOD): False
INFO:symExec:     Timestamp Dependency:                   False
INFO:symExec:     Re-Entrancy Vulnerability:              False
INFO:symExec:   ====== Analysis Completed ======
INFO:root:contract referenced_two.sol:referenced_two:
INFO:symExec:   ============ Results ===========
INFO:symExec:     EVM Code Coverage:                      98.3%
INFO:symExec:     Integer Underflow:                      False
INFO:symExec:     Integer Overflow:                       False
INFO:symExec:     Parity Multisig Bug 2:                  False
INFO:symExec:     Transaction Ordering Dependence Efficient  False
INFO:symExec:     TX.Origin Vulnerability  False
INFO:symExec:     Contract Referencing  False
INFO:symExec:     Callstack Depth Attack Vulnerability:  False
INFO:symExec:     Transaction-Ordering Dependence (TOD): False
INFO:symExec:     Timestamp Dependency:                   False
INFO:symExec:     Re-Entrancy Vulnerability:              False
INFO:symExec:   ====== Analysis Completed ======
shaz@ubuntu:~/Downloads/oyente$
```

2. Efficient Transaction Ordering Dependence

We identified the “Point of Attack” and created an algorithm which flags any potential attack on smart contracts.

```
contract MarketPlace {
    uint public price;
    uint public stock;
    address public owner;
    int transaction_successful =0;

    function check( ){
        if (msg.value == price)
            transaction_successful = 1;
    }

    function updatePrice ( uint _price ){
        if (msg.sender == owner){
            price = _price;
        }

    }

    function buy ( uint quant ) returns ( uint ){
        if ( msg.value < quant*price || quant > stock ){
            revert();
         }
        check();
        stock -= quant ;
    }
}
```

Check function is being used

Output

```
shaz@ubuntu: ~/Downloads/oyente
shaz@ubuntu:~/Downloads/oyente$ python oyente.py -s TOD_efficient.sol
WARNING:root:You are using solc version 0.4.24, The latest supported version is 0.4.19
INFO:root:contract TOD_efficient.sol:MarketPlace:
INFO:symExec:   ============ Results ===========
INFO:symExec:     EVM Code Coverage:                      99.8%
INFO:symExec:     Integer Underflow:                      False
INFO:symExec:     Integer Overflow:                       False
INFO:symExec:     Parity Multisig Bug 2:                  False
INFO:symExec:     Transaction Ordering Dependence Efficient  True
INFO:symExec:     TX.Origin Vulnerability  False
INFO:symExec:     Contract Referencing  False
INFO:symExec:     Callstack Depth Attack Vulnerability:  False
INFO:symExec:     Transaction-Ordering Dependence (TOD): False
INFO:symExec:     Timestamp Dependency:                   False
INFO:symExec:     Re-Entrancy Vulnerability:              False
INFO:symExec:   ====== Analysis Completed ======
shaz@ubuntu:~/Downloads/oyente$
```

Here we can see in the result that Transaction Ordering dependence vulnerability has been flagged true by our Oyente tool.

3. tx_origin Vulnerability

```
contract TxUserWallet {
    address owner;

    constructor() {
        owner = msg.sender;
    }

    function transferTo(address dest, uint amount) public {
        require(tx.origin == owner);
        dest.transfer(amount);
    }
}
```

tx.origin is an environment variable which refers to the owner of the contract. It can easily be manipulated to make a malicious smart contract.

tx.origin is being used

Output

```
shaz@ubuntu: ~/Downloads/oyente
shaz@ubuntu:~/Downloads/oyente$ python oyente.py -s tx_origin.sol
WARNING:root:You are using solc version 0.4.24, The latest supported version is 0.4.19
INFO:root:contract tx_origin.sol:TxUserWallet:
INFO:symExec:   ============ Results ===========
INFO:symExec:     EVM Code Coverage:                      94.4%
INFO:symExec:     Integer Underflow:                      False
INFO:symExec:     Integer Overflow:                       False
INFO:symExec:     Parity Multisig Bug 2:                  False
INFO:symExec:     Transaction Ordering Dependence Efficient  False
INFO:symExec:     TX.Origin Vulnerability  True
INFO:symExec:     Contract Referencing  False
INFO:symExec:     Callstack Depth Attack Vulnerability:  False
INFO:symExec:     Transaction-Ordering Dependence (TOD): False
INFO:symExec:     Timestamp Dependency:                   False
INFO:symExec:     Re-Entrancy Vulnerability:              False
INFO:symExec:   ====== Analysis Completed ======
shaz@ubuntu:~/Downloads/oyente$
```

Here we can see in the result that tx.origin vulnerability has been flagged true by our Oyente tool.

Conclusion

In order to achieve our goals,

- We have read relevant research papers, other relevant texts and documentations.
- We devised algorithms to detect two new vulnerabilities and coded the implementation.
- We also improved an existing vulnerability which could not detect attack in some critical cases.
- We performed testing on both the old and enhanced Oyente tools. The enhanced new Oyente tool has better performance than the old tool.

Signature of Student

Signature of Guide