Oyente: Making Smart Contracts Smarter

Loi Luu, Duc-Hiep Chu, Hrishi Olickel
Prateek Saxena, Aquinas Hobor
National University of Singapore, Yale-NUS College
Programming securely is hard

“Security can be no stronger than its weakest link”
Programming Secure Smart Contracts is Harder

• Smart contracts != normal programs
  – Self-executed
  – One-shot programs
    • Cannot patch

• New language
  – Solidity != JavaScript
  – Serpent != Python
I think TheDAO is getting drained right now

Etherdice is down for maintenance. We are having troubles with our smart contract and will probably need to invoke

King of the Ether Throne

An Ethereum DApp (a "contract"), living on the blockchain, that will make you a King or Queen, might grant you riches, and will immortalize your name.

Important Notice

A SERIOUS ISSUE has been identified that can cause monarch compensation payments to not be sent.

DO NOT send payments to the contract previously referenced on this page, or attempt to claim the throne. Refunds will CERTAINLY NOT be made for any payments made after this issue was identified on 2016-02-07.
Questions?

• Are there other bugs?
  – Apart from call-stack and reentrancy?
• How many contracts are vulnerable?

<table>
<thead>
<tr>
<th>No. of contracts</th>
</tr>
</thead>
<tbody>
<tr>
<td>35,000</td>
</tr>
<tr>
<td>70,000</td>
</tr>
<tr>
<td>105,000</td>
</tr>
<tr>
<td>140,000</td>
</tr>
</tbody>
</table>

The number of contracts grew significantly from 25/8/15 to 10/7/16.
Challenges

• Contracts code are not always available

```
contract Greetings {
  string greeting;
  function Greetings (string _greeting) public {
    greeting = _greeting;
  }
  /* main function */
  function greet() constant returns (string) {
    return greeting;
  }
}
```

• Too many contracts
  – Manual analysis is impossible
Contribution

• Identify New Smart Contract Bugs
  – Transaction Ordering Dependence (TOD)
  – Timestamp Dependence

• Oyente: An analyzer for smart contracts
  – Use symbolic execution
  – Detect all popular bugs
    • TOD
    • Timestamp dependence
    • Reentrancy
    • Mishandling exceptions (e.g. send)
  – Flags 8836/ 19366 contracts as vulnerable
    • As of May 2016
New Smart Contract Bugs

Transaction Ordering Dependence
Example: Puzzle Solver

Anyone can submit a solution to claim the reward

Owner can update the reward anytime
Scenario 1: SubmitSolution is triggered

Miners

Solution for Puzzle

Random TXs

Other TXs

Balance: 100

Balance: 0

PuzzleSolver Contract

SetDifficulty

reward = 100

SubmitSolution(solution)

if isCorrect(solution):
  Send(reward)

UpdateReward(newReward)

reward = newReward

Block

- Random TXs
- SubmitSolution
- Other TXs

+100
Scenario 2: Both SubmitSolution and UpdateReward are triggered

PuzzleSolver
SetDifficulty
reward = 100

PuzzleSolver Contract
Balance: 0
PuzzleSolver()
SetDifficulty
reward = 100
SubmitSolution(solution)
if isCorrect(solution):
Send(reward)
UpdateReward(newReward)
reward = newReward

Block
- UpdateReward = 0
- SubmitSolution
- Other TXs

Miners
Solution for Puzzle
Update Reward to $0!
Other TXs

+0
Transaction Ordering Dependence

• Observed state ≠ execution state
  • The expectation of the state of the contract may not be true during execution.
  • Miners decide the order of TXs

• Can be coincidence
  • Two transactions happen at the same time
Transaction Ordering Dependence

- Observed state != execution state
  - The expectation of the state of the contract may not be true during execution.
  - Miners decide the order of TXs

- Can be coincidence
  - Two transactions happen at the same time

- Can be malicious
  - Saw the targeted TX from the victim
  - Submit the second TX to update the reward
  - Both TXs enter the race

Solution for Puzzle
Update Reward to $0!
Other TXs
New Smart Contract Bugs

Timestamp Dependence
Contract: TheRun

```solidity
contract theRun {
    uint private LastPayout = 0;
    uint256 salt = block.timestamp;

    function random() returns (uint256 result){
        uint256 y = salt * block.number/(salt%5);
        uint256 seed = block.number/3 + (salt%300) + LastPayout + y;

        //h = the blockhash of the seed-th last block
        uint256 h = uint256(block.blockhash(seed));

        //random number between 1 and 100
        return uint256(h % 100) + 1;
    }
}
```
function lendGovernmentMoney(address buddy) 
  returns (bool) {

  if (lastTimeOfNewCredit + TWELVE_HOURS > block.timestamp) {
    msg.sender.send(amount);
    // Sends jackpot to the last creditor
    creditorAddresses[nCreditors - 1].send(profitFromCrash);
    owner.send(this.balance);

  }
}

Timestamp can be manipulated

- Miners can vary the block timestamp
  \[
  \text{block.timestamp} \leq \text{now} + 900 \land \text{block.timestamp} \geq \text{parent.timestamp}
  \]

- Bias the output of contract execution to their benefit
  - Timed puzzles, time-based RNGs
Oyente: An Analyzer for Smart Contracts
Architecture

- Based on symbolic execution
- Have separate modules
  - Can add more analysis separately

EXPLORER

ByteCode

Ethereum State

Z3 Bit-Vector Solver

CFG BUILDER

CORE ANALYSIS

VALIDATOR

Visualizer
Symbolic Execution

Is there any value of $x$?

$C_1 \land C_2 \land C_3 \land (z = x + 2)$

Theorem Prover

Is there any value of $x$?

$\text{NO}$  $\text{YES}$

$\mathbf{x} = 10$

$C_1: (\mathbf{x} > 0)$
$C_2: (\mathbf{z} < 15)$
$C_3: (\mathbf{z} < 8)$

$\mathbf{z} = x + 2$;
What Can Oyente Do?

- Detect Bugs In Existing Smart Contracts
  - Run with 19,366 contracts
  - 30 mins timeout per contract

- Test generation
  - Cover all possible paths of each program

Flagged Buggy Contracts

<table>
<thead>
<tr>
<th>Callstack</th>
<th>TOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>5411</td>
<td>3056</td>
</tr>
</tbody>
</table>

| 1385 | 135 |

<table>
<thead>
<tr>
<th>Reentrancy</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>340</td>
<td>186</td>
</tr>
<tr>
<td>83</td>
<td>52</td>
</tr>
</tbody>
</table>

Total

Unique
Oyente is Open Source

- https://github.com/ethereum/oyente
- Future work
  - Support more opcodes
  - Handle loops
  - Combine static and dynamic symbolic executions
More in the papers

• Solutions for all bugs
  – Semantic changes

• Details of Oyente’s design

• Some interesting statistics
  – All smart contracts
  – Evaluation results
Thanks!

loiluu@comp.nus.edu.sg
loi_luu