# HACKING

# smart contracts

@konstanthacker

Konstantinos Karagiannis
CTO, Security Consulting

BT

# ethereum is not bitcoin

*“The key component is this idea of a Turing-complete blockchain”*

--Vitalik Buterin

# smart contracts

- Business logic programs
- Semi autonomous
- Move value, enforce agreements
- Creativity the limit

# literally a billion reasons

Stephan Tual Follow
Slock.it Founder, Blockchain and Smart Contract Expert, Former CCO Ethereum
Jun 12 · 3 min read

## No DAO funds at risk following the Ethereum smart contract 'recursive call' bug discovery

Our team is blessed to have Dr. Christian Reitwießner, Father of Solidity, as its Advisor. During the early development of the DAO Framework 1.1 and thanks to his guidance we were made aware of a generic vulnerability common to all Ethereum smart contracts. We promptly circumvented this so-called "recursive call vulnerability" or "race to empty" from the DAO Framework 1.1 as can be seen on line 580:

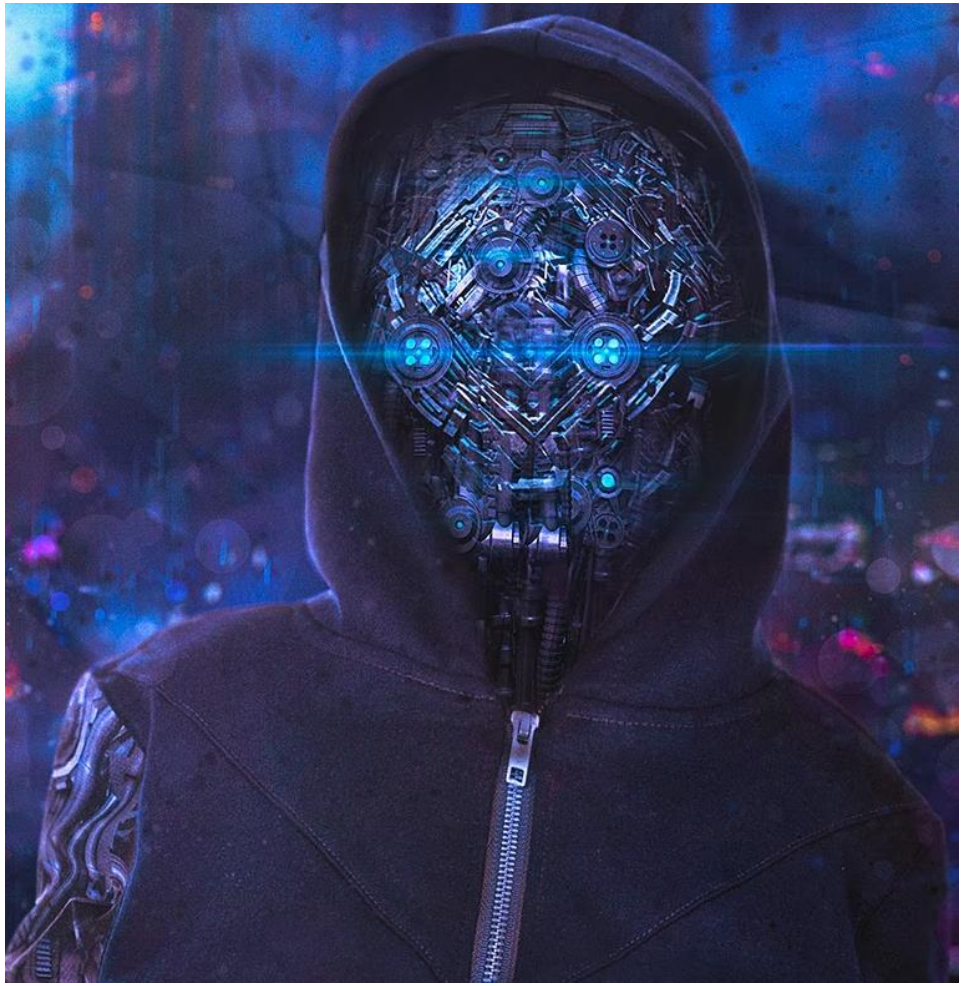
WIRED A $50 Million Hack Just Showed That the DAO Was All Too Human

SHARE
SHARE
TWEET
PIN
COMMENT 13
EMAIL

# A $50 MILLION HACK JUST SHOWED THAT THE DAO WAS ALL TOO HUMAN

# caveats

- No zero days
- No customer code
- Yes, a methodology
- No, I doubt smart contracts will get that smart

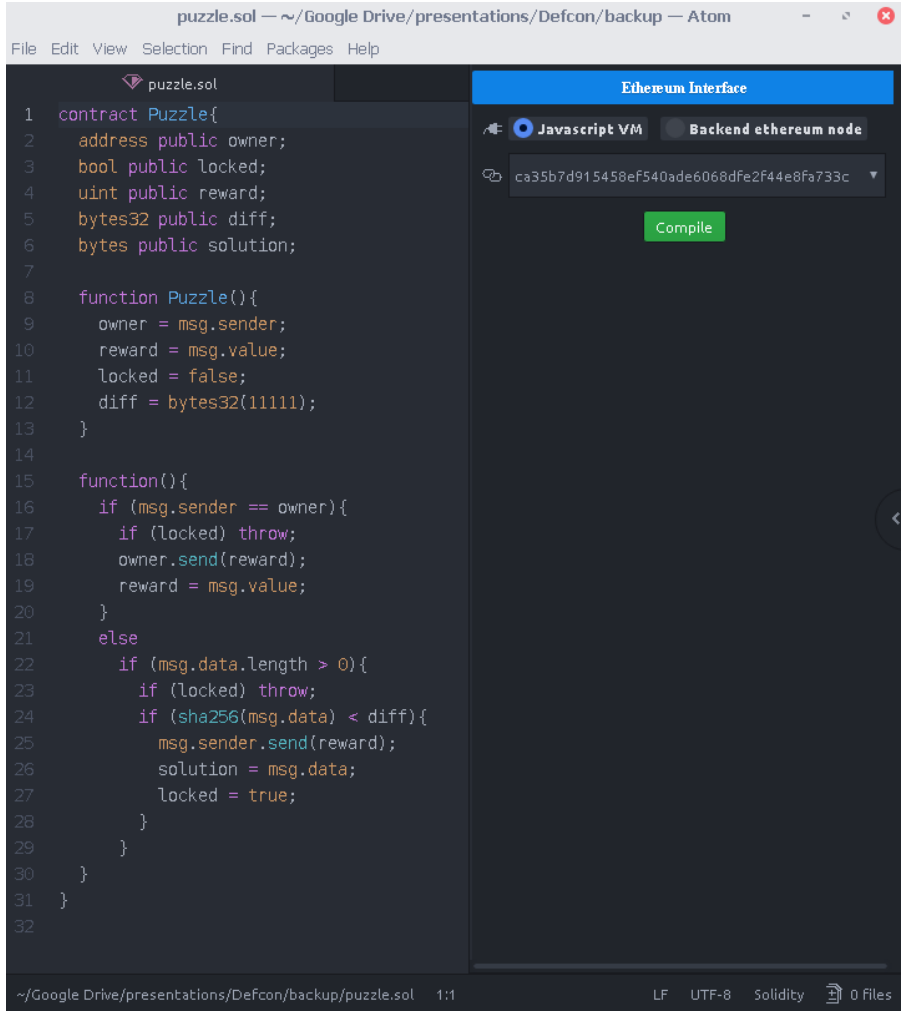


# solidity

- Language of choice
- High level, compiles to bytecode
- Similarities to JavaScript and C
- Supports:
  - libraries
  - inheritance
  - user-defined types
  - assembly inline

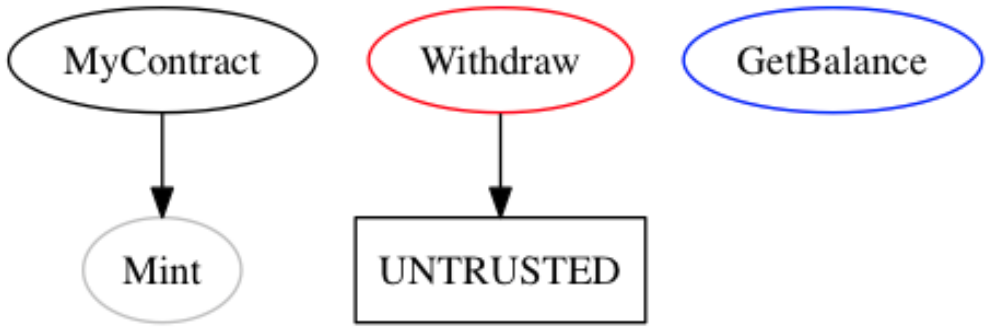
# dev tools

- .sol files > bytecode > blockchain
- Auditing .sol easier with highlighting
- Atom my fave, with plugins
  - language-ethereum
  - etheratom
- Remix–browser based



# solgraph

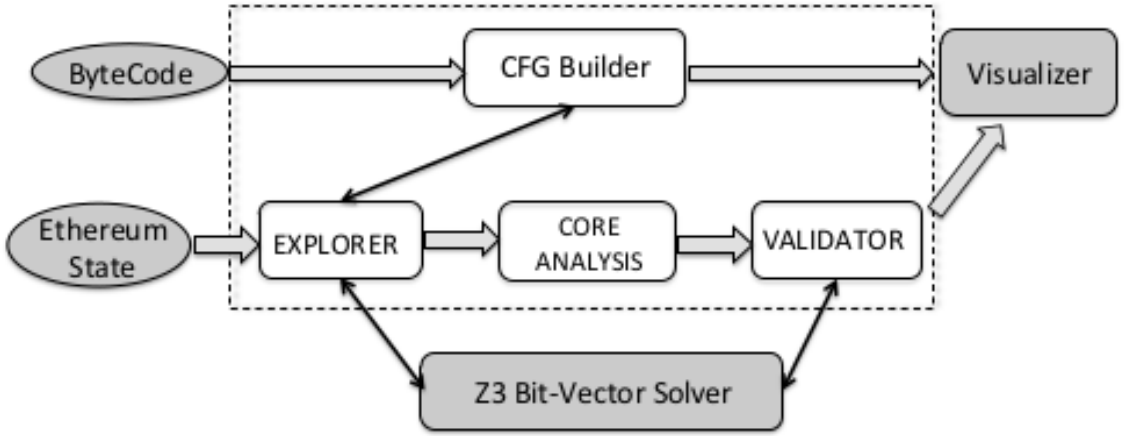
```
contract MyContract {
  uint balance;

  function MyContract() {
    Mint(1000000);
  }

  function Mint(uint amount) internal {
    balance = amount;
  }

  function Withdraw() {
    msg.sender.send(balance);
  }

  function GetBalance() constant returns(uint) {
    return balance;
  }
}
```




**Legend:**

- Black: Public function
- Gray: Internal function
- Red: Send to external address
- Blue: Constant function

# oyente




```
root@f962f954fab1: /oyente
root@f962f954fab1:/oyente# python oyente.py -s defcon.sol
INFO:root:Contract defcon.sol:greeter:
INFO:symExec:Running, please wait...
INFO:symExec:   ============ Results ===========
INFO:symExec:     CallStack Attack:      False
INFO:symExec:     Concurrency Bug:       False
INFO:symExec:     Time Dependency:       False
INFO:symExec:     Reentrancy bug exists: False
INFO:symExec:   ====== Analysis Completed ======
root@f962f954fab1:/oyente#
```

- Symbolic execution tool
- Works with EVM byte code or .sol files
- Detects 4* vulns
- Low false positive rate

# basic methodology

- Interview devs
- Load .sol file, preferably with highlighting
- Try compiling
- Dissect code flow—optional solgraph
- Run oyente (cross fingers)
- Manually verify 3/4 vuln yay/nays
- Proceed to manually check for following vulns…

# reentrancy

```
1   contract ReEntrancy {
2
3     mapping (address => uint) private expendableTokens;
4
5     function stealTokens() public {
6         uint amountToLose = expendableTokens[msg.sender];
7         if (!(msg.sender.call.value(amountToLose)())) { throw; }
8         expendableTokens[msg.sender] = 0;
9     }
10  }
```

# leave off the first “re” for savings

```
1   contract Entrancy {
2
3     mapping (address => uint) private expendableTokens;
4
5     function stealTokens() public {
6         uint amountToLose = expendableTokens[msg.sender];
7         expendableTokens[msg.sender] = 0;
8         if (!(msg.sender.call.value(amountToLose)())) { throw; }
9     }
10  }
```

# unchecked send in king of the ether

GitHub, Inc. [US] | https://github.com/kieranelby/KingOfTheEtherThrone/blob/v0.4.0/contr

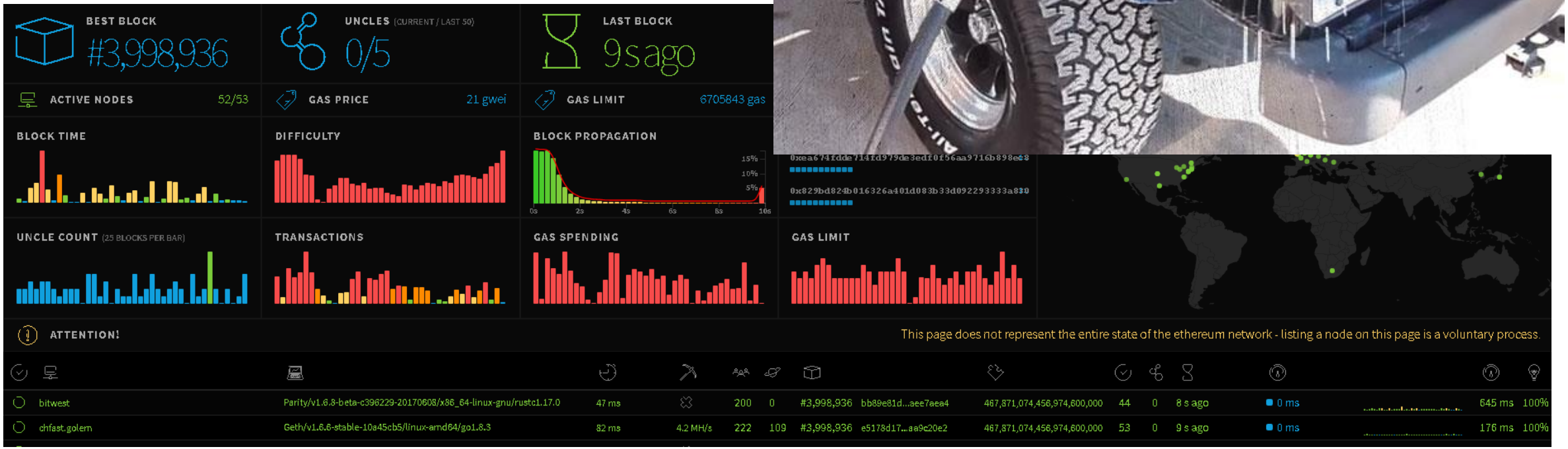
```
117         uint compensation = valuePaid - wizardCommission;
118
119         if (currentMonarch.etherAddress != wizardAddress) {
120             currentMonarch.etherAddress.send(compensation);
121         } else {
122             // When the throne is vacant, the fee accumulates for the wizard.
123         }
124
```

# unchecked send

```
1  if (kingOfLosingDone && !( compensationSent ) ) {
2    monarch.send(500);
3    compensationSent = True;
4  }
```

```
1  if (kingOfLosingDone && !( compensationSent ) ) {
2    if (monarch.send(500))
3      compensationSent = True;
4    else throw;
5  }
```

# gas limits


BEST BLOCK
#3,998,936
UNCLES (CURRENT / LAST 50)
0/5
LAST BLOCK
9s ago
ACTIVE NODES 52/53
GAS PRICE 21 gwei
GAS LIMIT 6705843 gas
BLOCK TIME
DIFFICULTY
BLOCK PROPAGATION
UNCLE COUNT (25 BLOCKS PER BAR)
TRANSACTIONS
GAS SPENDING
GAS LIMIT
ATTENTION!
This page does not represent the entire state of the ethereum network - listing a node on this page is a voluntary process.
bitwest Parity/v1.6.8-beta-c396229-20170608/x86_64-linux-gnu/rustc1.17.0 47 ms 200 0 #3,998,936 bb89e81d...aee7aea4 467,871,074,456,974,600,000 44 0 8 s ago 0 ms 645 ms 100%
chfast.golem Geth/v1.6.6-stable-10a45cb5/linux-amd64/go1.8.3 82 ms 4.2 MH/s 222 109 #3,998,936 e5178d17...aa9c20e2 467,871,074,456,974,600,000 53 0 9 s ago 0 ms 176 ms 100%

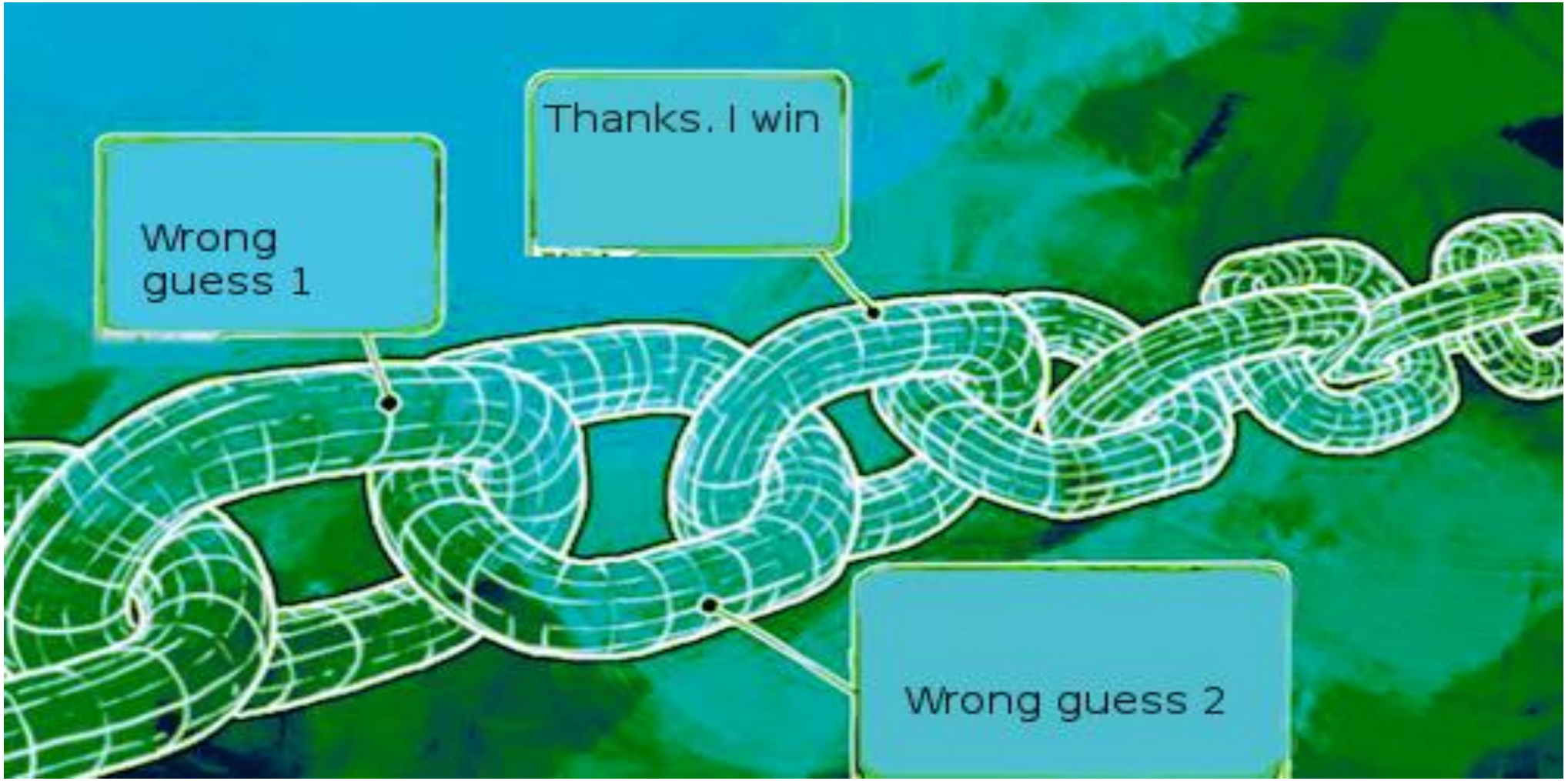
# withdraw don’t send

```
contract SendContract {
    address public richest;
    uint public mostSent;

    function SendContract() payable {
        richest = msg.sender;
        mostSent = msg.value;
    }

    function becomeRichest() payable returns (bool) {
        if (msg.value > mostSent) {
            richest.transfer(msg.value);
            richest = msg.sender;
            mostSent = msg.value;
            return true;
        } else {
            return false;
```

# withdrawn not sent

```
contract WithdrawalContract {
    address public richest;
    uint public mostSent;

    mapping (address => uint) pendingWithdrawals;

    function WithdrawalContract() payable {
        richest = msg.sender;
        mostSent = msg.value;
    }

    function becomeRichest() payable returns (bool) {
        if (msg.value > mostSent) {
            pendingWithdrawals[richest] += msg.value;
            richest = msg.sender;
            mostSent = msg.value;
            return true;
        } else {
            return false;
        }
    }

    function withdraw() {
        uint amount = pendingWithdrawals[msg.sender];
        pendingWithdrawals[msg.sender] = 0;
        msg.sender.transfer(amount);
```

# encryption



# transaction-ordering dependence

```
1   contract Puzzle{
2     address public owner;
3     bool public locked;
4     uint public reward;
5     bytes32 public diff;
6     bytes public solution;
7
8     function Puzzle(){
9       owner = msg.sender;
10      reward = msg.value;
11      locked = false;
12      diff = bytes32(11111);
13    }
14
```

```
15    function(){
16      if (msg.sender == owner){
17        if (locked) throw;
18        owner.send(reward);
19        reward = msg.value;
20      }
21      else
22        if (msg.data.length > 0){
23          if (locked) throw;
24          if (sha256(msg.data) < diff){
25            msg.sender.send(reward);
26            solution = msg.data;
27            locked = true;
28          }
```

# call-stack depth limit

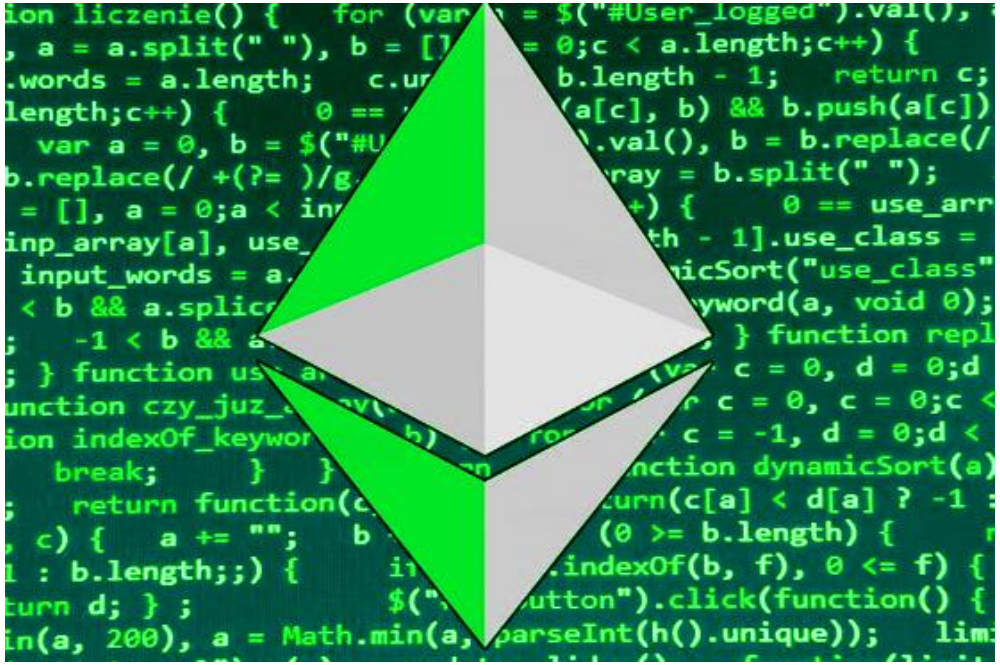
```
root@f962f954fab1:/oyente
root@f962f954fab1:/oyente# python oyente.py -s defcon.sol
INFO:root:Contract defcon.sol:greeter:
INFO:symExec:Running, please wait...
INFO:symExec:   ============ Results ===========
INFO:symExec:     CallStack Attack:      False
INFO:symExec:     Concurrency Bug:       False
INFO:symExec:     Time Dependency:       F
INFO:symExec:     Reentrancy bug exists: F
INFO:symExec:   ====== Analysis Completed
root@f962f954fab1:/oyente#
```

# variable or function ambiguity

```
1  Player[] public persons;
2
3  uint public payoutCursor_Id_ = 0;
4  uint public balance = 0;
5
6  address public owner;
7
8  uint public payoutCursor_Id=0;
9  ...
10 while (balance > persons[payoutCursor_Id_].deposit / 100 * 115) {
11   uint MultipliedPayout = persons[payoutCursor_Id_].deposit / 100 * 115;
12   persons[payoutCursor_Id].etherAddress.send(MultipliedPayout);
13
14   balance -= MultipliedPayout;
15   payoutCursor_Id_++;
```

# odds and ends

- Input validation – require(condition)
- Timestamp dependence
- Business logic flaws
- Separating public/private data



# get involved

# dox me … or just keep in touch

@konstanthacker

konstantinos.karagiannis@bt.com