MIS|TI™ PRESENTS

InfoSec World

Conference & Expo 2018

# ETHEREUM AND SMART CONTRACT SECURITY

*Konstantinos Karagiannis*

*Chief Technology Officer, Security Consulting*

***@KonstantHacker***

# ETHEREUM IS NOT BITCOIN

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

# SMART CONTRACTS

# 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

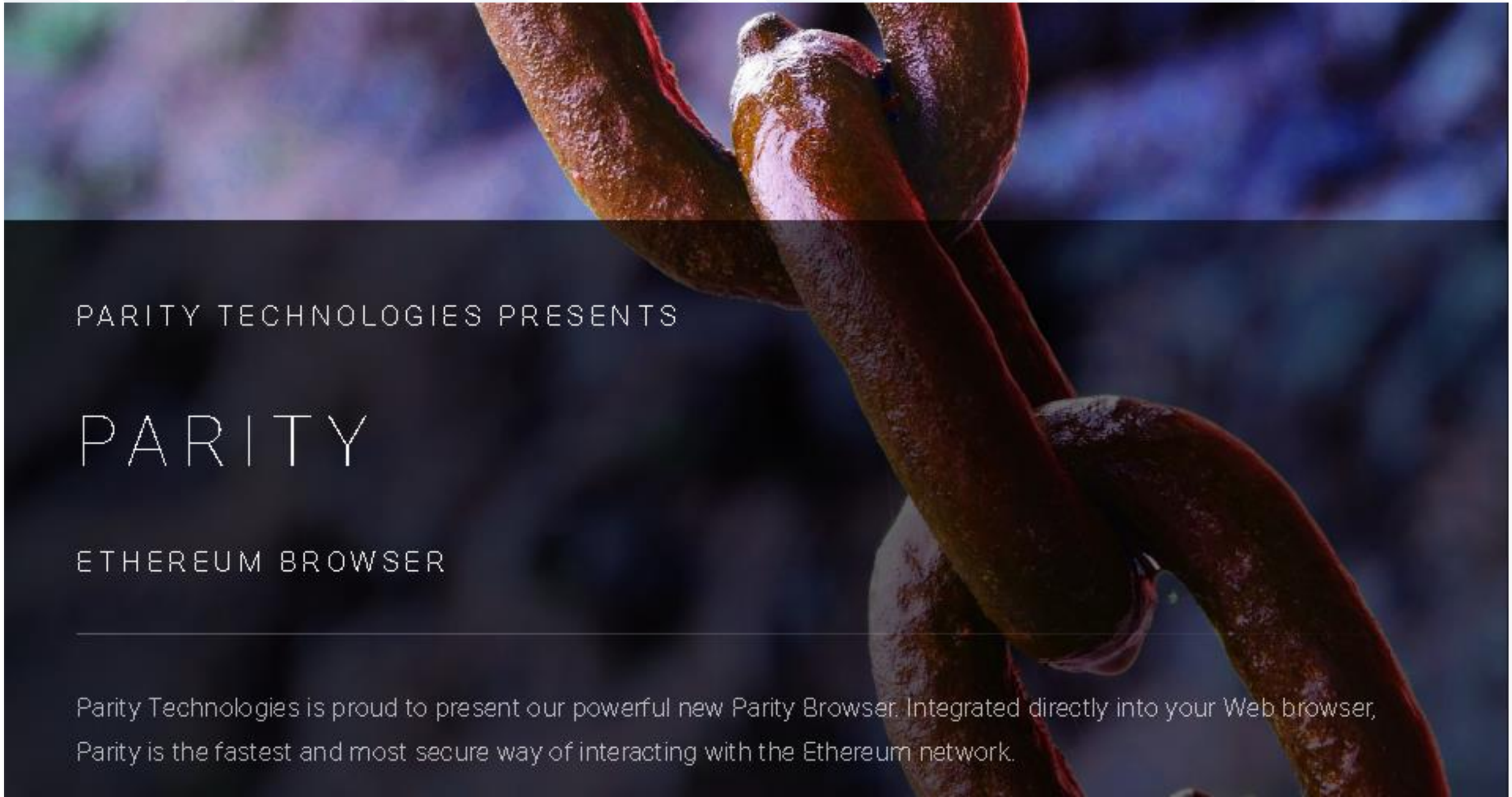
PIN

COMMENT 13

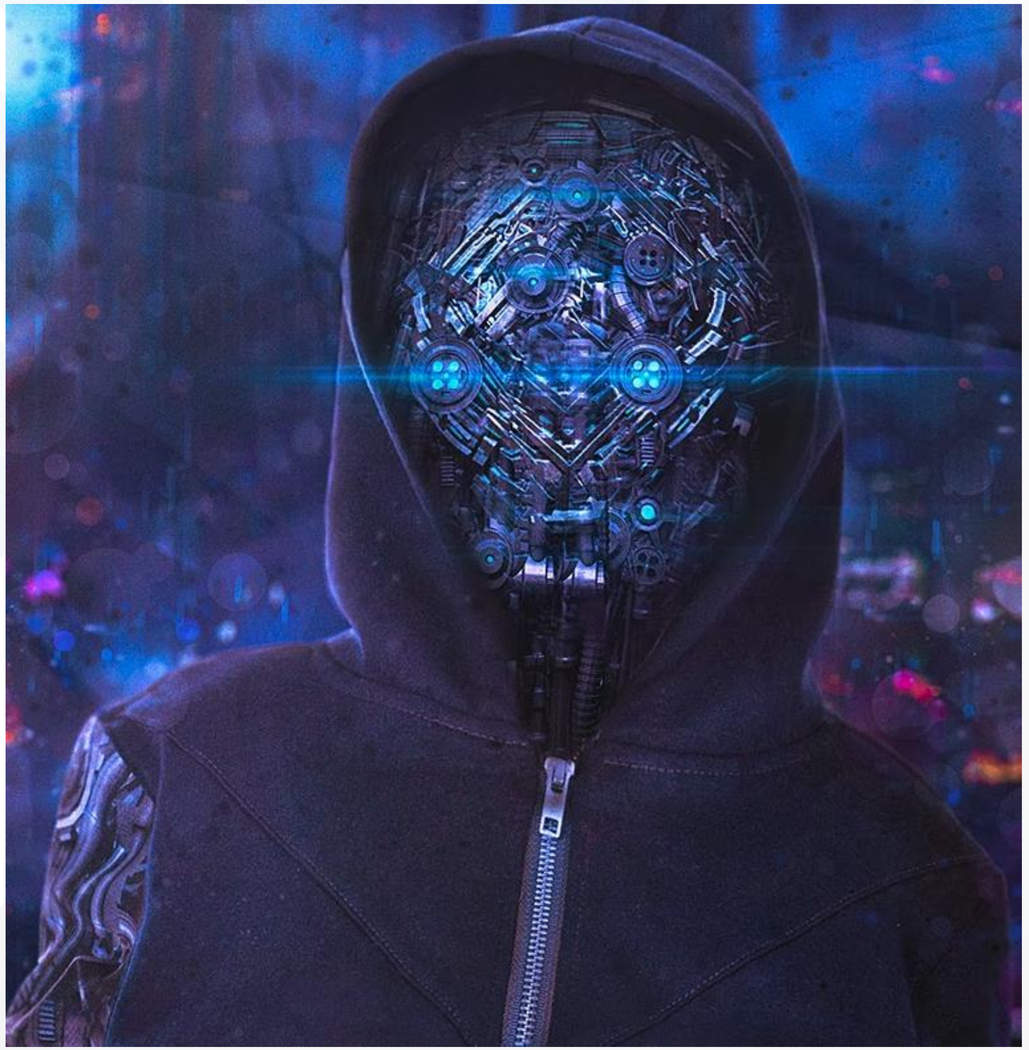
EMAIL

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

# 30 MILLION REASONS WILL DO



# CAVEATS



# SOLIDITY

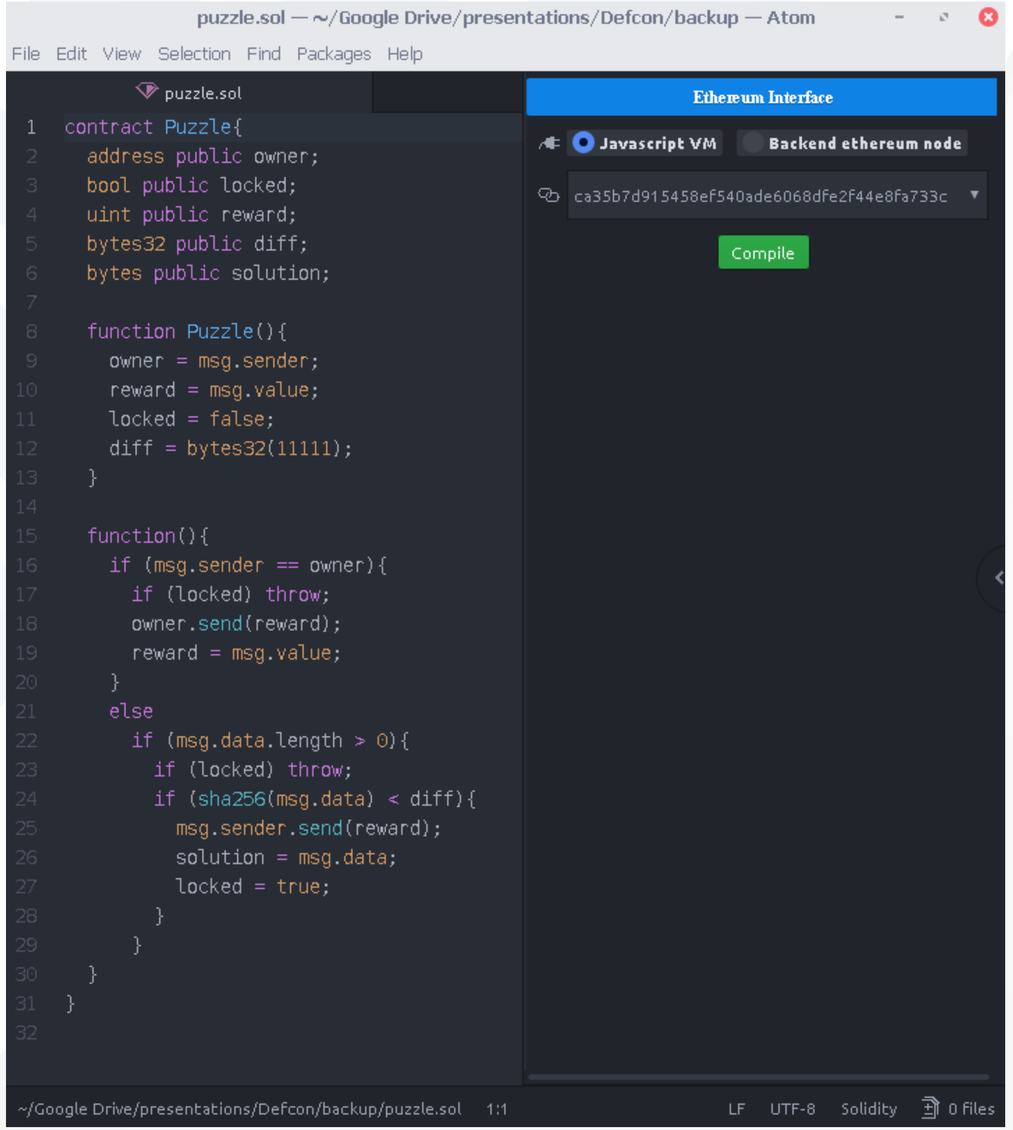
# DEV TOOLS

.sol files > bytecode > blockchain

Atom with plugins:

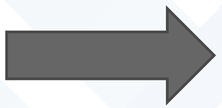
- language-ethereum
- etheratom

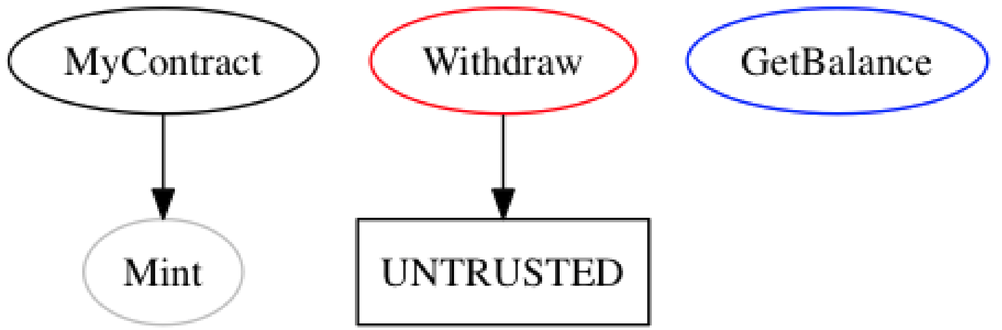
Remix: browser based



# 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;
  }
}
```

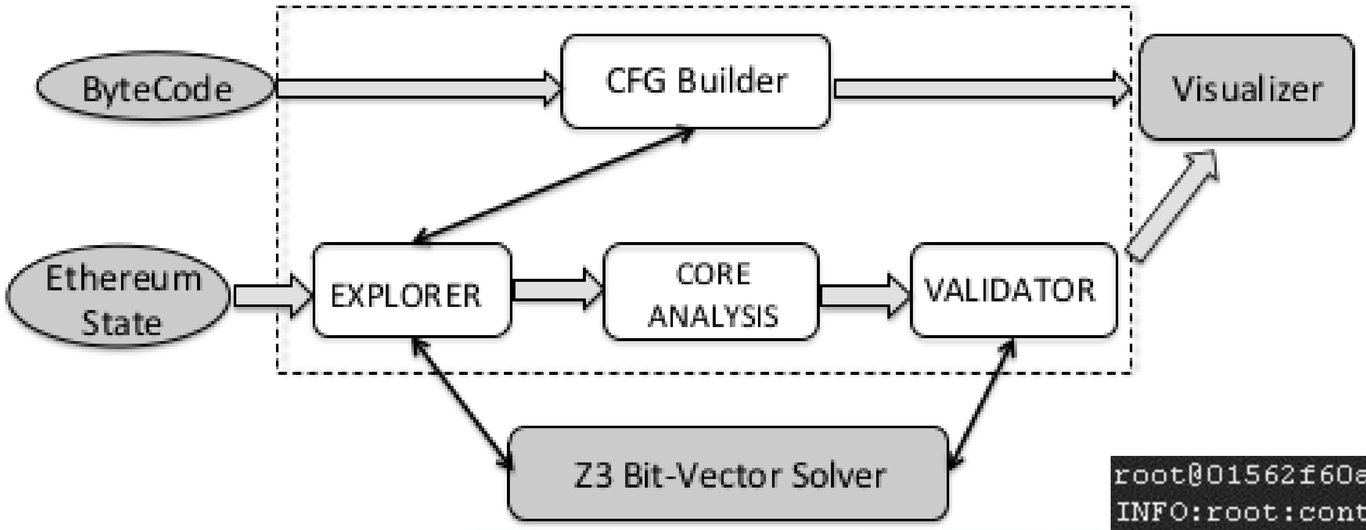





**Legend:**

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

# OYENTE




```
root@01562f60ae20:/oyente/oyente# python oyente.py -s hackertest.sol
INFO:root:contract hackertest.sol:greeter:
INFO:symExec:    ============ Results ===========
INFO:symExec:      EVM Code Coverage:                      99.5%
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 ======
```

# MANTICORE

# BASIC METHODOLOGY

Interview devs
Review .sol file
Try compiling
Dissect code flow—optional solgraph
Run oyente (cross fingers)
Run Manticore
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

```
contract Entrancy {

  mapping (address => uint) private expendableTokens;

  function stealTokens() public {
      uint amountToLose = expendableTokens[msg.sender];
      expendableTokens[msg.sender] = 0;
      if (!(msg.sender.call.value(amountToLose)())) { throw; }
  }
}
```

# REENTRANCY (AND IRONY) IN THE DAO CODE

```
// Burn DAO Tokens
        Transfer(msg.sender, 0, balances[msg.sender]);
        withdrawRewardFor(msg.sender); // be nice, and get his rewards
        totalSupply -= balances[msg.sender];
        balances[msg.sender] = 0;
        paidOut[msg.sender] = 0;
        return true;
    }
```

# DEFAULT PUBLIC – PARITY WALLET HACK

```
@@ -104,7 +104,7 @@ contract WalletLibrary is WalletEvents {
104 104
105 105     // constructor is given number of sigs required to do protected "onlymanyowners" transactions
106 106     // as well as the selection of addresses capable of confirming them.
107     -   function initMultiowned(address[] _owners, uint _required) {
    107 +   function initMultiowned(address[] _owners, uint _required) internal {
108 108       m_numOwners = _owners.length + 1;
109 109       m_owners[1] = uint(msg.sender);
110 110       m_ownerIndex[uint(msg.sender)] = 1;
@@ -198,7 +198,7 @@ contract WalletLibrary is WalletEvents {
198 198     }
199 199
200 200     // constructor - stores initial daily limit and records the present day's index.
201     -   function initDaylimit(uint _limit) {
    201 +   function initDaylimit(uint _limit) internal {

    216 +
214 217     // constructor - just pass on the owner array to the multiowned and
215 218     // the limit to daylimit
216     -   function initWallet(address[] _owners, uint _required, uint _daylimit) {
    219 +   function initWallet(address[] _owners, uint _required, uint _daylimit) only_uninitialized {
```

# INITWALLET

Overview | Comments

## Transaction Information

Tools & Utilities

| | |
|---|---|
| TxHash: | 0x9dbf0326a03a2a3719c27be4fa69aacc9857fd231a8d9dcaede4bb083def75ec |
| Block Height: | 4043800 (28739 block confirmations) |
| TimeStamp: | 6 days 5 hrs ago (Jul-19-2017 12:18:15 PM +UTC) |
| From: | 0xb3764761e297d6f121e79c32a65829cd1ddb4d32 (MultisigExploit-Hacker) |
| To: | Contract 0xbec591de75b8699a3ba52f073428822d0bfc0d7e |
| Value: | 0 Ether ($0.00) |
| Gas Limit: | 82703 |
| Gas Price: | 0.000000021 Ether (21 Gwei) |
| Gas Used By Txn: | 66839 |
| Actual Tx Cost/Fee: | 0.001403619 Ether ($0.29) |
| Cumulative Gas Used: | 1283734 |
| Nonce: | 5 |
| Input Data: | Function: initWallet(address[] _owners, uint256 _required, uint256 _daylimit) ***<br><br>MethodID: 0xe46dcfeb<br>[0]: 0000000000000000000000000000000000000000000000000000000000000060<br>[1]: 0000000000000000000000000000000000000000000000000000000000000000 |

Convert To Ascii

# EXECUTE

Overview | Internal Transactions | Event Logs | Comments

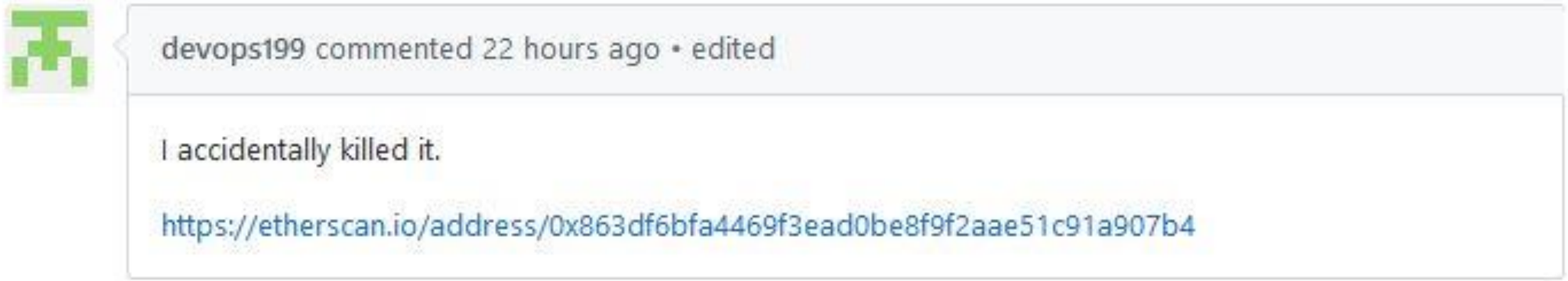
## Transaction Information

Tools & Utilities

| | |
|---|---|
| TxHash: | 0xeef10fc5170f669b86c4cd0444882a960872213251f8bf2f55d6188633aa7be7c |
| Block Height: | 4043802 (28738 block confirmations) |
| TimeStamp: | 6 days 5 hrs ago (Jul-19-2017 12:19:36 PM +UTC) |
| From: | 0xb3764761e297d6f121e79c32a65829cd1ddb4d32 (MultisigExploit-Hacker) |
| To: | Contract 0xbec591de75b8699a3ba52f073428822d0bfc0d7e<br>TRANSFER 82,189 Ether to → 0xb3764761e297d6f121... |
| Value: | 0 Ether ($0.00) |
| Gas Limit: | 78926 |
| Gas Price: | 0.000000021 Ether (21 Gwei) |
| Gas Used By Txn: | 58433 |
| Actual Tx Cost/Fee: | 0.001227093 Ether ($0.25) |
| Cumulative Gas Used: | 1821881 |
| Nonce: | 6 |
| Input Data: | Function: execute(address _to, uint256 _value, bytes _data)<br>***<br><br>MethodID: 0xb61d27f6<br>[0]:000000000000000000000000b3764761e297d6f121e79c32a65829cd1ddb4d32<br>[1]:0000000000000000000000000000000000000000000000000116779808c03e4140000<br>[2]:0000000000000000000000000000000000000000000000000000000000000000 |

Convert To Ascii

# PARITY MULTISIG WALLET HACK 2



# UNCHECKED SEND IN KING OF THE ETHER

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

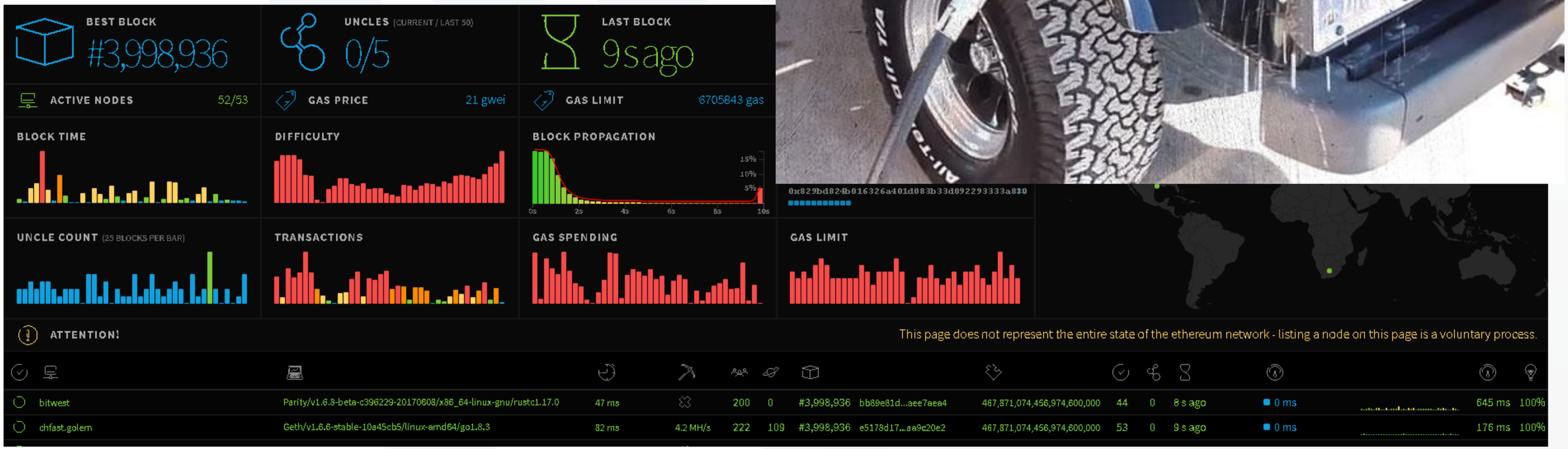
```
117         uint compensation = valuePaid - wizardCommission;
118
119         if (currentMonarch.etherAddress != wizardAddress) {
120             currentMonarch.etherAddress.send(compensation);
121         } else {
122             // When the throne is vacant, the fee accumulates for the wizard.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



# 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

```
INFO:symExec:    ============ Results ===========
INFO:symExec:      EVM Code Coverage:                      99.5%
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
INFO:symExec:    ====== Analysis Completed ==
```

# 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_++;
```

# INPUT VALIDATION

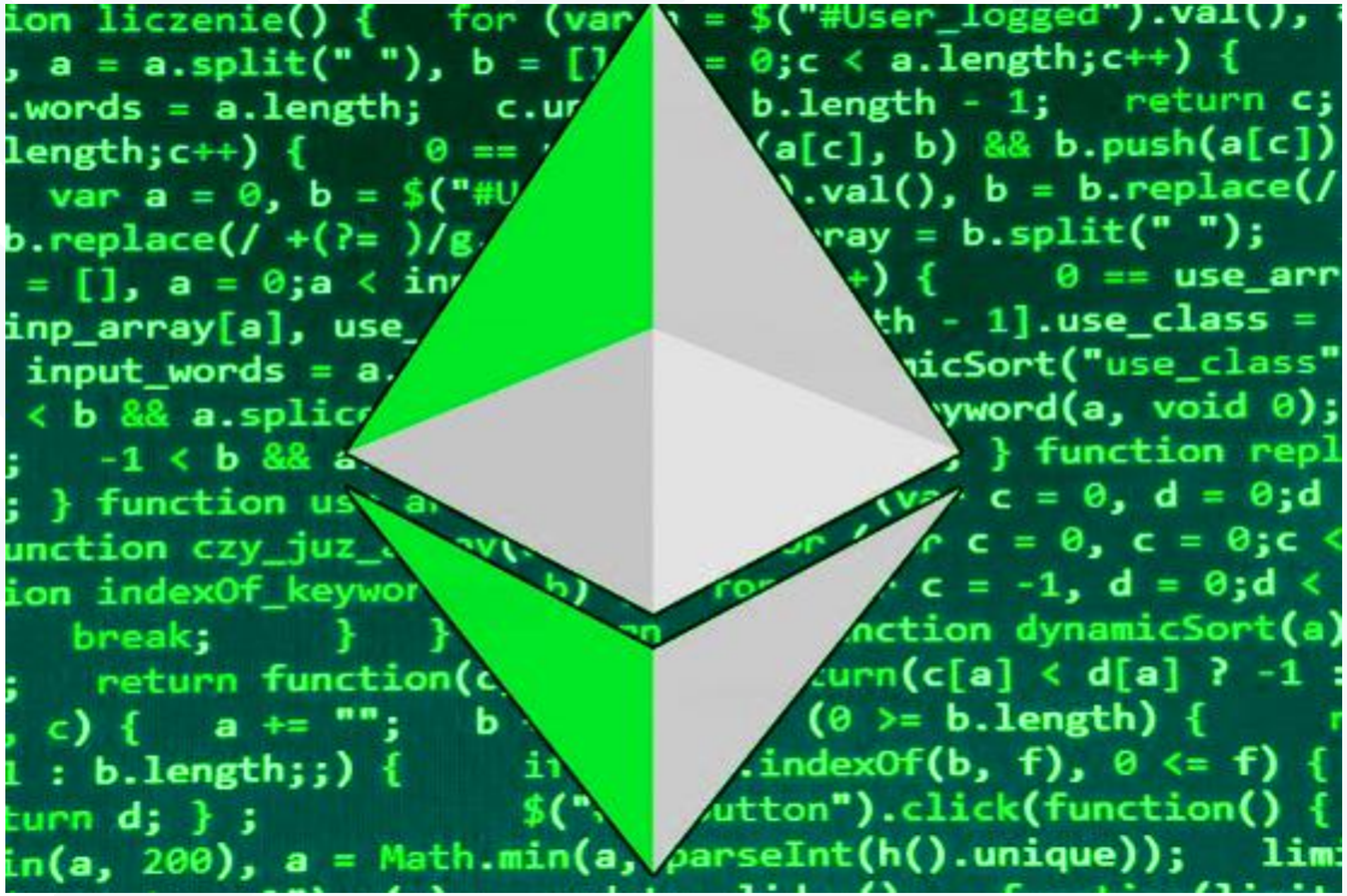
throw -- being deprecated

require (condition) – check external conditions (invalid inputs or errors in external components)

assert (condition) – internal errors

# ODDS AND ENDS

Timestamp dependence
Business logic flaws
Separating public/private data



# GET INVOLVED

MIS|TI™ PRESENTS

InfoSec World

Conference & Expo 2018

# THANK YOU

## PLEASE FILL OUT YOUR EVALUATIONS!

*Konstantinos Karagiannis*

*Chief Technology Officer, Security Consulting*

***@KonstantHacker***