when transactions aren’t enough

• “The key component is this idea of a Turing-complete blockchain”
• --Vitalik Buterin
meow—putting that computing to use
smart contracts
billions, or just millions, of reasons

No DAO funds at risk following the Ethereum smart contract ‘recursive call’ bug discovery

Our team is blessed to have Advisor. During the early days, he guided us in Ethereum smart contracts. “recursive call vulnerability can be seen on line 580:
problem isn’t going away

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

<table>
<thead>
<tr>
<th>Root</th>
<th>01562f60a20</th>
<th>oyente/oyente# python oyente.py -s hackertest.sol</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFO</td>
<td>root:contract hackertest.sol:greeter:</td>
<td></td>
</tr>
<tr>
<td>INFO</td>
<td>symExec: Results --------</td>
<td></td>
</tr>
<tr>
<td>INFO</td>
<td>symExec: EVM Code Coverage: 99.5%</td>
<td></td>
</tr>
<tr>
<td>INFO</td>
<td>symExec: Parity Multisig Bug 2: False</td>
<td></td>
</tr>
<tr>
<td>INFO</td>
<td>symExec: Callstack Depth Attack Vulnerability: False</td>
<td></td>
</tr>
<tr>
<td>INFO</td>
<td>symExec: Transaction-Ordering Dependence (TOD): False</td>
<td></td>
</tr>
<tr>
<td>INFO</td>
<td>symExec: Timestamp Dependency: False</td>
<td></td>
</tr>
<tr>
<td>INFO</td>
<td>symExec: Re-Entrancy Vulnerability: False</td>
<td></td>
</tr>
<tr>
<td>INFO</td>
<td>symExec: Analysis Completed ======</td>
<td></td>
</tr>
</tbody>
</table>
MAIAN

[Check on PRODIGAL]
Vulnerability found
Vulnerability confirmed (see the log below)

[Check on SUICIDAL]
Vulnerability found
Vulnerability confirmed (see the log below)

[Check on GREEDY]
Not vulnerable

To keep MAIAN free and up to date, consider donating Ether to our account: 0xe040c29e1326e7836a3c852f718351a6d3f4a96
basic methodology

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

```solidity
contract Entrance {

    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
default public – Parity wallet hack

```solidity
contract WalletLibrary is WalletEvents {

// constructor is given number of sigs required to do protected "onlyManynowners" transactions
// as well as the selection of addresses capable of confirming them.

function initMultiowed(address[] _owners, uint _required) {
    m_numOwners = _owners.length + 1;
    m_owners[1] = uint(msg.sender);
    m_ownerIndex[uint(msg.sender)] = 1;
}

// constructor - stores initial daily limit and records the present day's index.

function initDaylimit(uint _limit) {
    m_daylimit = _limit;
}

// constructor - just pass on the owner array to the multiowned and
// the limit to daylimit

function initWallet(address[] _owners, uint _required, uint _daylimit) {
    // ...
<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TxHash</td>
<td>0xeaf0f6c170d652b68c4d044362a66097822132596296de1883333a78ae7c</td>
</tr>
<tr>
<td>Block Height</td>
<td>404302 (2873 block confirmations)</td>
</tr>
<tr>
<td>TimeStamp</td>
<td>8 days 5 hrs ago (Jul-19-2017 12:19:38 PM +UTC)</td>
</tr>
<tr>
<td>From</td>
<td>0x37c8781e21906f121e73c32e6829cd16bb4d32 MultisigExploit-Hacker</td>
</tr>
<tr>
<td>To</td>
<td>0x Contract Debugger691d75b88920a6207342982cd3be667a7e TRANSFER: 0.18 Ether to 0x37c8781e21906f121..</td>
</tr>
<tr>
<td>Value</td>
<td>0 Ether (0.00)</td>
</tr>
<tr>
<td>Gas Limit</td>
<td>75628</td>
</tr>
<tr>
<td>Gas Price</td>
<td>0.000000000021 Ether (21 Gwei)</td>
</tr>
<tr>
<td>Gas Used By Tx</td>
<td>56433</td>
</tr>
<tr>
<td>Actual Tx Cost/Fee</td>
<td>0.001227863 Ether (0.26)</td>
</tr>
<tr>
<td>Cumulative Gas Used</td>
<td>1821881</td>
</tr>
<tr>
<td>Nonce</td>
<td>6</td>
</tr>
<tr>
<td>Input Data</td>
<td>function execute(address to, uint256 value, bytes data)</td>
</tr>
</tbody>
</table>

```
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: 0xe6dcfeb
[0]:000000000000000000000000000000000000000000000000000000000000060
[1]:000000000000000000000000000000000000000000000000000000000000000
[2]:000000000000000000000000000000000000000000000000000000000000000
[3]:000000000000000000000000000000000000000000000000000000000000001
[4]:000000000000000000000000000000000000000000000000000000000000000

Function: kill(address _to)
MethodID: 0xcbf0b0c0
[0]:000000000000000000000000000000000000000000000000000000000000060

not going with the (over)flow

\[ 2^{256} - 1 \]
unchecked send in king of the ether

```solidity
    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

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

```java
1 if (kingOfLosingDone && !(compensationSent)) {
2    if (monarch.send(500))
3        compensationSent = True;
4    else throw;
5 }
```
gas limits
withdraw don’t send

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

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

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

Wrong guess 1.

Thanks, I win.

Wrong guess 2.
transaction-ordering dependence

```solidity
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

```solidity
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 ==========

Announcement of imminent hard fork for EIP150 gas cost changes:

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...
variable or function ambiguity

```solidity
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
- Separating public/private data
things might be getting better?
keep in touch

@KonstantHacker