Blockchain Research Progress
-- Smart Contracts Security

Chao Zhang
Tsinghua University

清華大學
Tsinghua University

About

2004-2008-2013 ➔ 2013-2016 ➔ 2016-present

北京大學 PEKING UNIVERSITY | Berkeley UNIVERSITY OF CALIFORNIA | 清華大學 Tsinghua University

Honors

- Thousand Youth Talents Plan
- Qiu Shi Outstanding Young Scholar
- Young Elite Scientists Sponsorship Program by CAST/CCF

Research Groups

- **ChainTrust**: Blockchain Security
- VUL337: Software & System Security

Cyber Grand Challenge (CGC)

Automated Attack & Defense, by DARPA

Defense #1 in 2015 ， Offense #2 in 2016

Contents

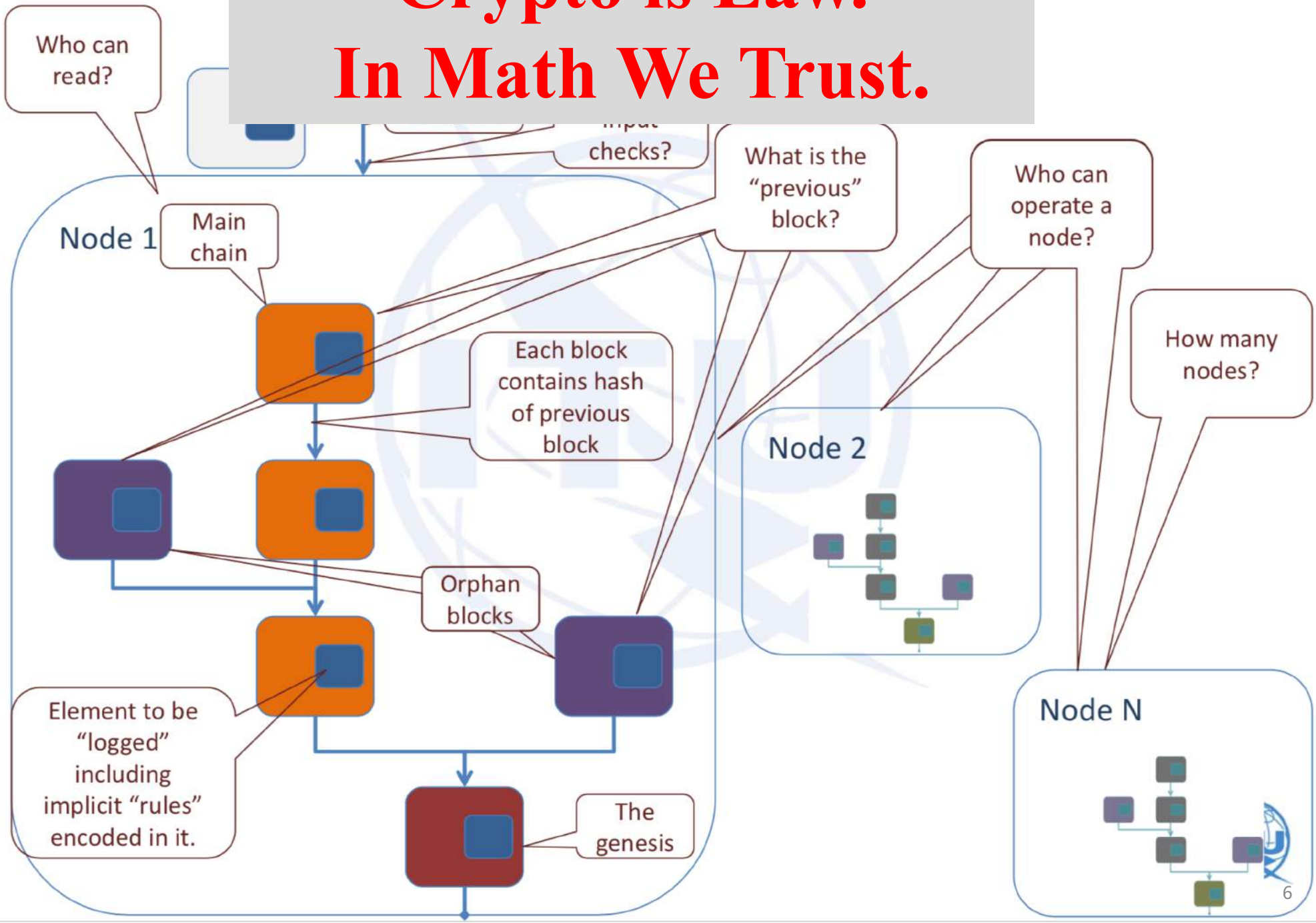
- Blockchain Background
- Smart Contracts Security Incidents
- Smart Contracts Security Solutions
- Our Solution to Integer Overflow
- Conclusion

Blockchain

Bitcoin: a decentralized*, public ledger of transactions

* on a peer-to-peer (p2p) virtual network

Crypto is Law.
In Math We Trust.



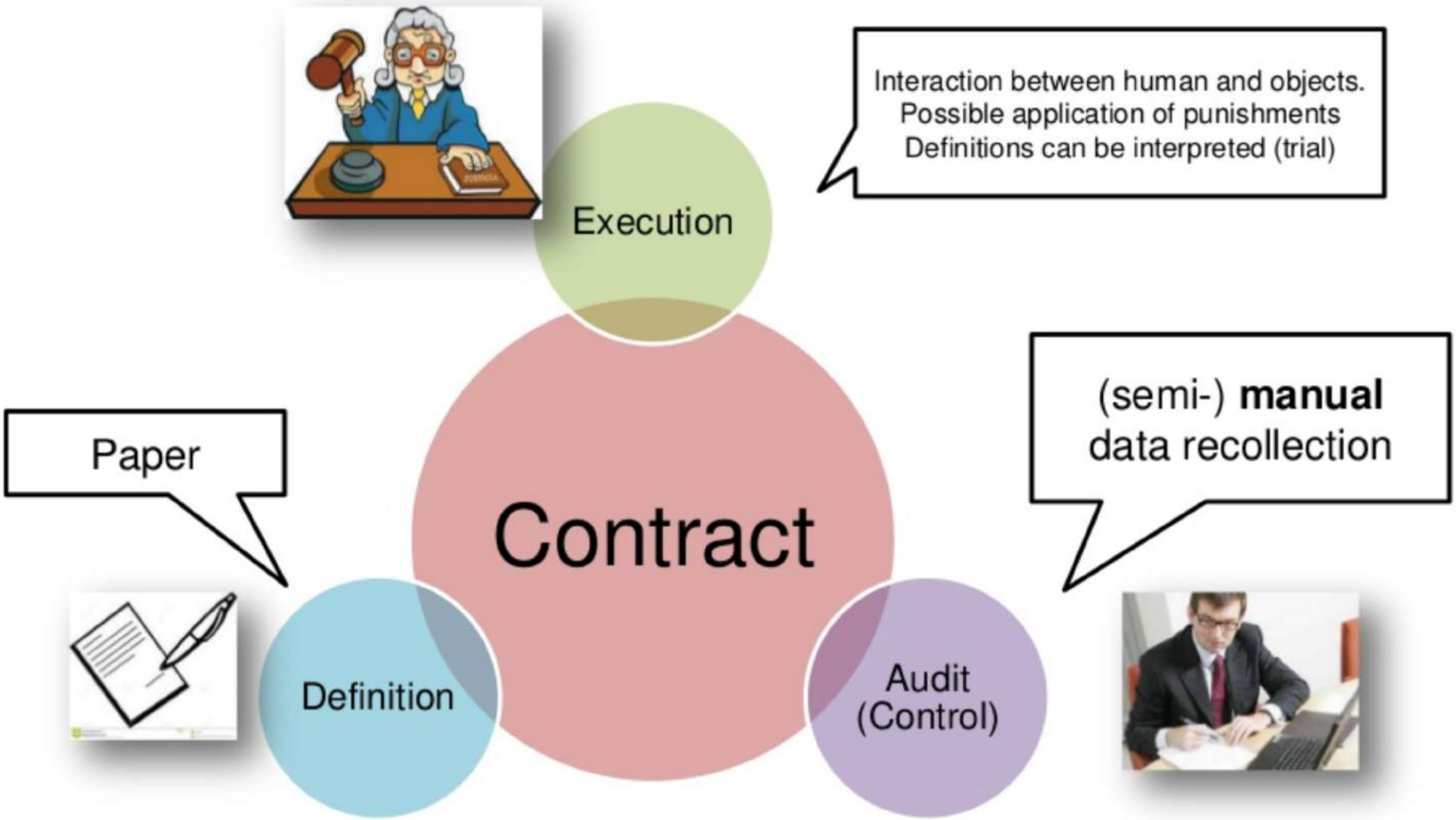
Smart Contract

- What do miners do when mining a block?
- Take Bitcoin as an example
  - place transactions in a block data structure
  - search for solutions of a hard math problem (PoW)
  - broadcast the solution to neighbors

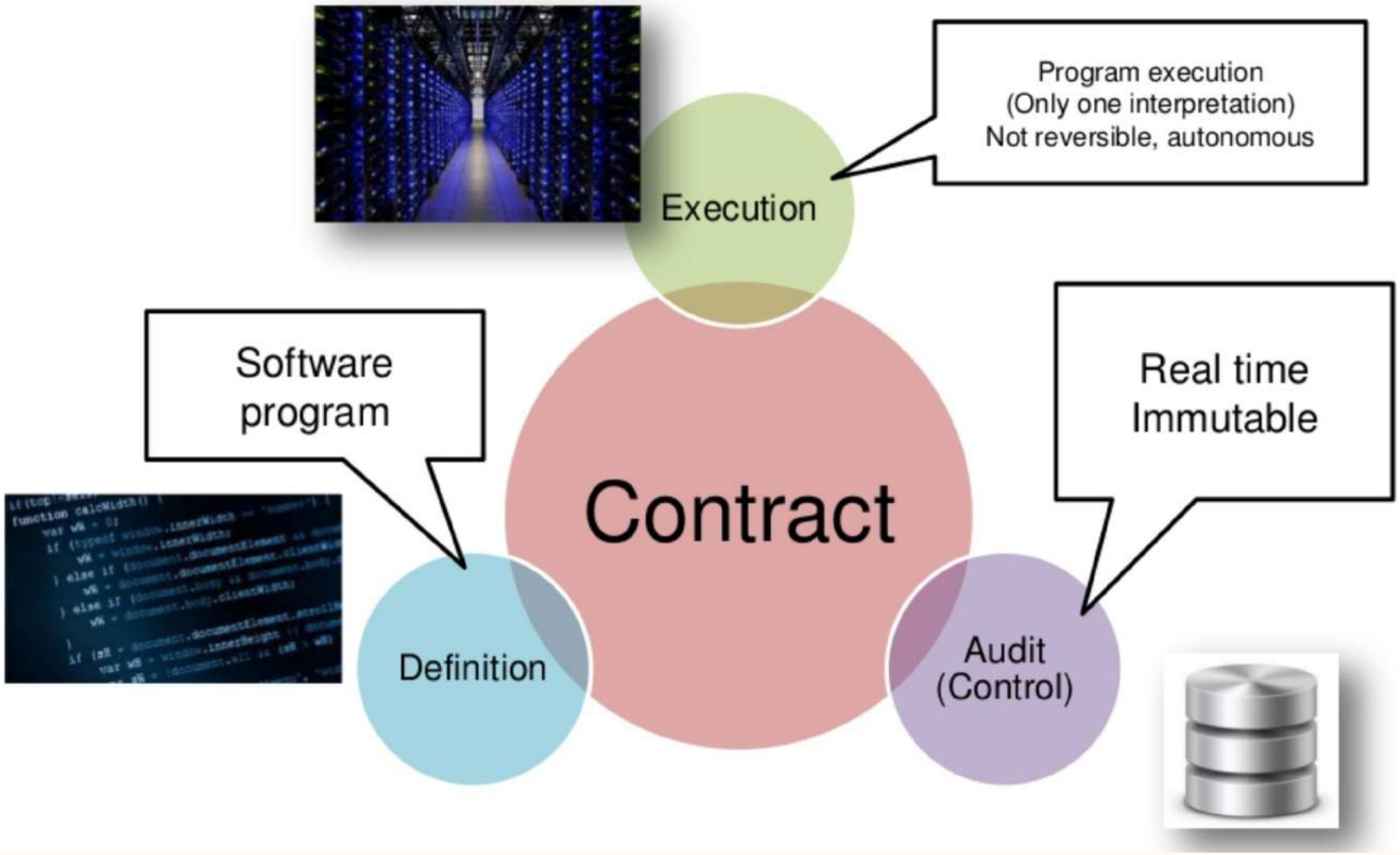
- What are transactions?
  - basically, add/sub operations to update user balances

- Can we do more within transactions?
  - Yes. Use smart contracts to execute arbitrary code.

Traditional Contract



Code is Law.

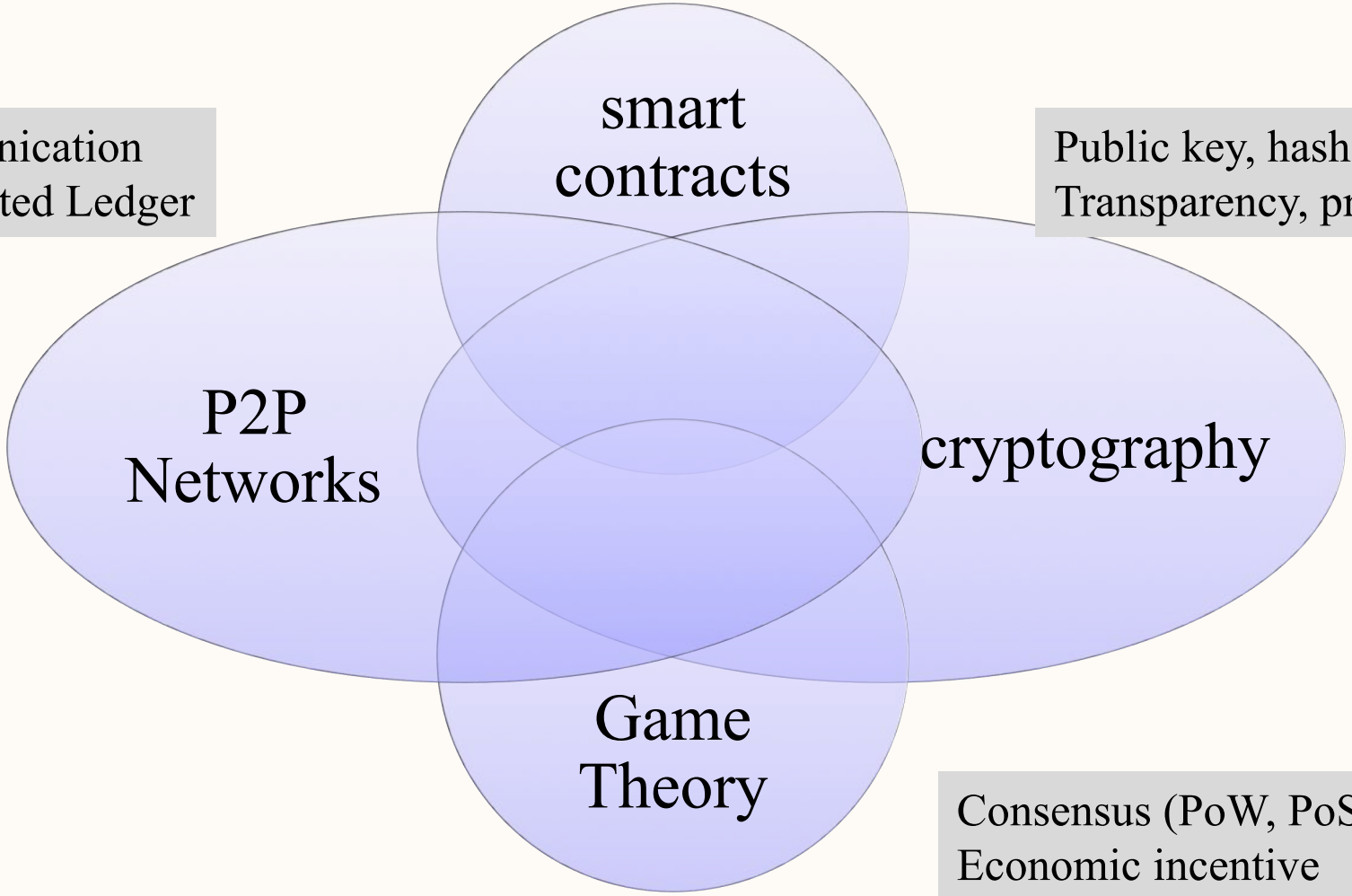

Program execution
(Only one interpretation)
Not reversible, autonomous
Execution
Software
program
Real time
Immutable
Contract
Definition
Audit
(Control)

Technologies

Distributed applications
Ecosystem

Communication
Distributed Ledger

Public key, hash, signature..
Transparency, privacy…



Consensus (PoW, PoS, DPoS…)
Economic incentive

Blockchain Build Blocks

- Smart Contracts (Relations)
  - define participants' relationships
- Transactions (Assets)
  - track asset movements
- Consensus Rules (Governance)
  - define game theory of actors
- P2P Network (Physical)
  - define communication protocol
- TCP/IP Infrastructure
- Cryptography Algorithms

Security Risks

- Smart Contracts (Relations)
  - define participants' relationships
- Transactions (Assets)
  - track asset movements
- Consensus Rules (Governance)
  - define game theory of actors
- P2P Network (Physical)
  - define communication protocol
- TCP/IP Infrastructure
- Cryptography Algorithms

- Smart Contract Security
  - Contracts, virtual machine
- Transactions Risks
  - Private key protection/retrieval
  - Replay attack
    - GETH client incidents
  - Exchange risks
  - Wallet risks
- Consensus risks:
  - Centralized: 51% power
  - Forking
- P2P Network risks:
  - Network availability
  - Eclipse attacks

Contents

- Blockchain Background
- Smart Contracts Security Incidents
- Smart Contracts Security Solutions
- Our Solution to Integer Overflow
- Conclusion

Incident #1: The DAO Hack

- 2016/4/30, 28-day funding window
- 11.6M ETH, 15%
- 2016/6/12, developers found the recursive call bug
- 2016/6/18, The DAO Hack, 3.6M ETH
- 2016/6/17, Soft-fork (freeze hacked ETH)
- 2016/7/20, hard-fork (reverse DAO), ETH/ETC

Excerpt from DAO contract (simplified)

Contract DAO:

```
mapping (address => int64) balances;

function withdraw(uint x) {
 if (balances[msg.sender] >= x) {
   msg.sender.call.value(x)();
   balances[msg.sender] -= x;
 }
}
```

Re-entrancy hazards in Ethereum

Balance 100

Wallet Contract

```
function doWithdraw() {
    A.withdraw(100);
}
function() {
    EventMoneyReceived(msg.value);
}
```

Ether Balance 200
balances[user] 0

Contract DAO:

```
mapping (address => int64) balances;

function withdraw(uint x) {
 if (balances[msg.sender] >= x) {
   callee.call.value(x)();
   balances[msg.sender] -= x;
 }
}
```

Re-entrancy hazards in Ethereum

Balance 300

Attacker Contract

```
function startAttack() {
    A.withdraw(100);
}
function() {
    A.withdraw(100);
}
```

Ether Balance 0

balances[attacker] -300

Contract DAO:

```
mapping (address => int64) balances;

function withdraw(uint x) {
 if (balances[msg.sender] >= x) {
   callee.call.value(x)();
   balances[msg.sender] -= x;
 }
}
```

Incident #2: Parity Wallet

- In July 2017, an exploitable vulnerability was exploited
  - 153,037 Ether (worth $30+ million) was stolen from three wallet contracts
  - A remaining $78 million worth of tokens plus 377,105+ ETH (around $72 million) were recovered by a daring whitehat rescue, and returned to their *rightful* owners
- In November 2017, a second bug was triggered, leading to $150M frozen funds.

Parity Wallet uses the “Prototype Inheritance” pattern

* Toy version of code

```
contract Wallet {
  address  walletLibrary;
  address owner;

  function Wallet(address _owner) {
      _walletLibrary = ${hardcoded address};
       _walletLibrary.delegatecall(
         "initWallet(address)",
         _owner);
  }
  function withdraw(uint amount)
     returns (bool success) {
       return _walletLibrary.delegatecall(
          "withdraw(uint)",
          amount);
  }
   // fallback function gets called if no
   // other function matches
  function () payable {
       _walletLibrary.delegatecall(msg.data);
  }
}
```

Wallet Library contract:

```
contract WalletLibrary {
     address owner;
     // called by constructor
     function initWallet(address _owner) {
         owner = _owner;
         // ... more setup ...
     }
     function changeOwner(address _new_owner)
     function () payable {
         // ... receive money, log events, ...
     }
     function withdraw(uint amount);
}
```

0xa657491c1e7f16adb39b9b60e87bbb8d93988bc3

Multiple Wallet contracts, all refer to Wallet Library

Why? Reduces fees from duplicate data!

1. Constructor is called, invokes **`initWallet`**

```
contract Wallet {
  address _walletLibrary;
  address owner;

  function Wallet(address _owner) {
      _walletLibrary = ${hardcoded address};
       _walletLibrary.delegatecall(
         "initWallet(address)",
         _owner);
  }
  function withdraw(uint amount)
     returns (bool success) {
       return _walletLibrary.delegatecall(
          "withdraw(uint)",
          amount);
  }
   // fallback function gets called if no
   // other function matches
  function () payable {
       _walletLibrary.delegatecall(msg.data);
  }
}
```

Wallet Library contract:

```
contract WalletLibrary {
     address owner;
     // called by constructor
     function initWallet(address _owner) {
         owner = _owner;
         // ... more setup ...
     }
     function changeOwner(address _new_owner)
     function () payable {
         // ... receive money, log events, ...
     }
     function withdraw(uint amount);
}
```

0xa657491c1e7f16adb39b9b60e87bbb8d93988bc3

2. The withdraw function invokes **`withdraw`**

```
contract Wallet {
   address _walletLibrary;
   address owner;

   function Wallet(address _owner) {
       _walletLibrary = ${hardcoded address};
        _walletLibrary.delegatecall(
          "initWallet(address)",
          _owner);
   }
   function withdraw(uint amount)
      returns (bool success) {
        return _walletLibrary.delegatecall(
           "withdraw(uint)",
           amount);
   }
    // fallback function gets called if no
    // other function matches
   function () payable {
        _walletLibrary.delegatecall(msg.data);
   }
}
```

Wallet Library contract:

```
contract WalletLibrary {
     address owner;
     // called by constructor
     function initWallet(address _owner) {
         owner = _owner;
         // ... more setup ...
     }
     function changeOwner(address _new_owner)
     function () payable {
         // ... receive money, log events, ...
     }
     function withdraw(uint amount);
}
```

0xa657491c1e7f16adb39b9b60e87bbb8d93988bc3

3. The fallback function can invoke ***any method***

```
contract Wallet {
  address _walletLibrary;
  address owner;

  function Wallet(address _owner) {
      _walletLibrary = ${hardcoded address};
       _walletLibrary.delegatecall(
         "initWallet(address)",
         _owner);
  }
  function withdraw(uint amount)
     returns (bool success) {
       return _walletLibrary.delegatecall(
          "withdraw(uint)",
          amount);
  }
   // fallback function gets called if no
   // other function matches
  function () payable {
       _walletLibrary.delegatecall(msg.data);
  }
}
```

Wallet Library contract:

```
contract WalletLibrary {
     address owner;
     // called by constructor
     function initWallet(address _owner) {
         owner = _owner;
         // ... more setup ...
     }
     function changeOwner(address _new_owner)
     function () payable {
         // ... receive money, log events, ...
     }
     function withdraw(uint amount);
}
```

Attacker calls:

`victim.call("initWallet(address)", attacker)`

Fixing the July 2017 Parity wallet bug

Old Wallet Library contract:

```
contract WalletLibrary {
     address owner;
     // called by constructor
     function initWallet(address _owner) {
         owner = _owner;
         // ... more setup ...
     }
     function changeOwner(address _new_owner)
     function () payable {
         // ... receive money, log events, ...
     }
     function withdraw(uint amount);
}
```

0xa657491c1e7f16adb39b9b60e87bbb8d93988bc3

New Library contract:

```
contract WalletLibrary {
     address owner;
     // called by constructor
     function initWallet(address _owner) {
         require(initialized == false);
         initialized = true;
         owner = _owner;
         // ... more setup ...
     }
     function changeOwner(address _new_owner)
     function () payable {
         // ... receive money, log events, ...
     }
     function withdraw(uint amount);
}
```

0x863df6bfa4469f3ead0be8f9f2aae51c91a907b4

anyone can kill your contract #6995

Open **devops199** opened this issue 22 hours ago · 12 comments

The WalletLibrary contract

devops199 commented 22 hours ago • edited

I accidentally killed it.

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

Hello, first of all i'm not the owner of that contract. I was able to make myself the owner of that contract because its uninitialized.

These (https://pastebin.com/ejakDR1f) multi_sig wallets deployed using Parity were using the library located at "0x863df6bfa4469f3ead0be8f9f2aae51c91a907b4" address. I made myself the owner of "0x863df6bfa4469f3ead0be8f9f2aae51c91a907b4" contract and killed it and now when i query the dependent contracts "isowner(<any_addr>)" they all return TRUE because the delegate call made to a died contract.

I believe some one might exploit.

anyone can kill your contract #6995

Parity wallet uses “Prototype Inheritance”
The WalletLibrary should only act as a template.
- It should not have any “state”
- Its constructor has never been called

But, the WalletLibrary is actually just a contract!

Attacker calls:

```
library.call("initWallet(address)",
             attacker);
library.kill();
```

Thereafter, any method call to any instance “Wallet” will fail, since the target of delegatecall is destroyed

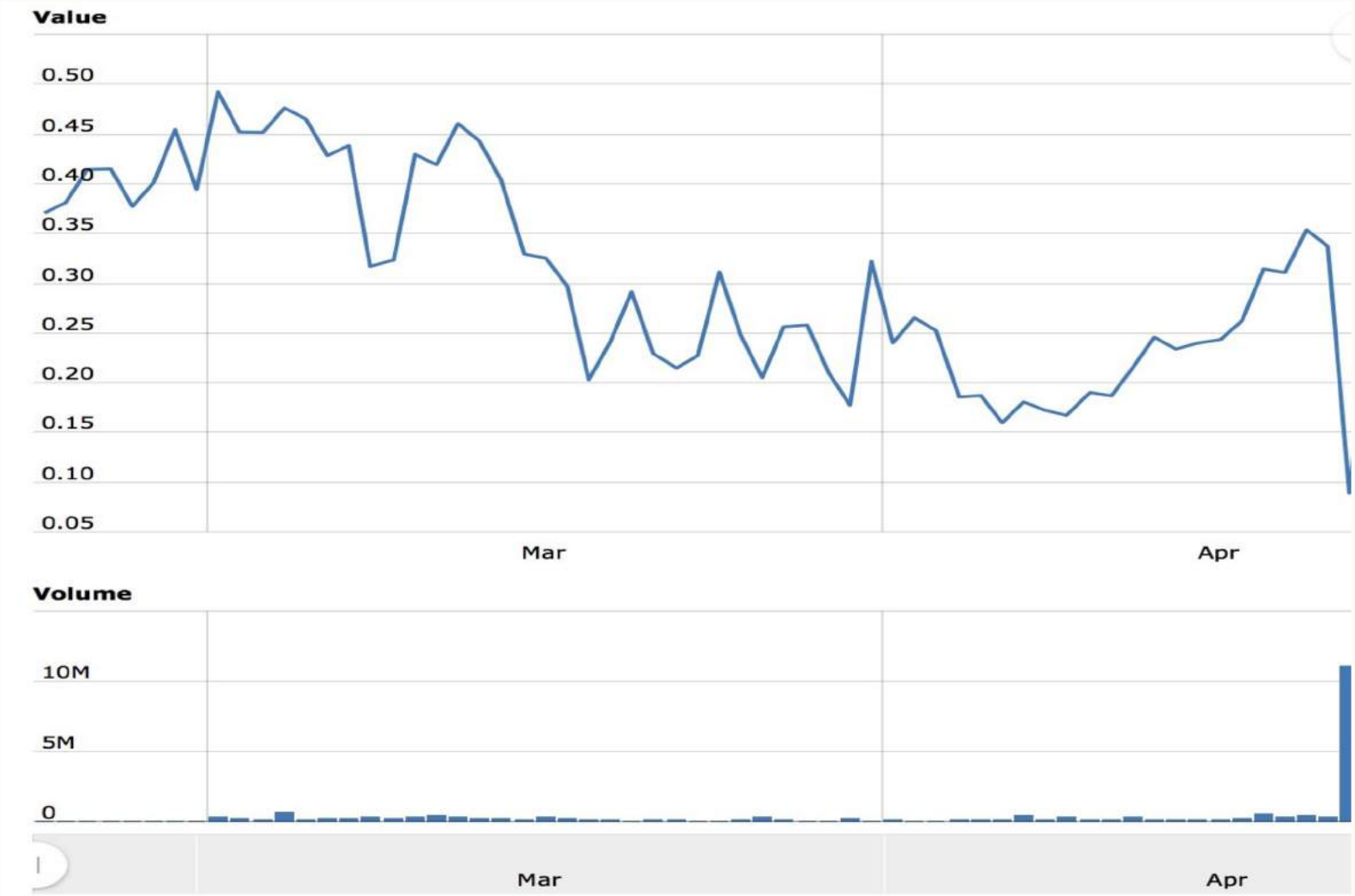
Wallet Library contract:

```
contract WalletLibrary {
    address owner;
    // called by constructor
    function initWallet(address _owner) {
        require(initialized == false);
        initialized = true;
        owner = _owner;
        // ... more setup ...
    }
    function changeOwner(address _new_owner)
    function () payable {
        // ... receive money, log events, ...
    }
    function withdraw(uint amount);
}
```

0x863df6bfa4469f3ead0be8f9f2aae51c91a907b4

Incident #3: BEC integer overflow

2018/4/22, the trading value of BEC (cap: $0.3B) drops to 0



batchOverflow (CVE-2018-10299)

```
255   function batchTransfer(address[] _receivers, uint256 _value) public whenNotPaused returns (bool) {
256     uint cnt = _receivers.length;
257     uint256 amount = uint256(cnt) * _value;
258     require(cnt > 0 && cnt <= 20);
259     require(_value > 0 && balances[msg.sender] >= amount);
260
261     balances[msg.sender] = balances[msg.sender].sub(amount);
262     for (uint i = 0; i < cnt; i++) {
263         balances[_receivers[i]] = balances[_receivers[i]].add(_value);
264         Transfer(msg.sender, _receivers[i], _value);
265     }
266     return true;
267   }
268 }
```

Transactions | Token Transfers | Comments

Latest 4 Erc20 Token Transfer Events

TxHash	Age	From		To	Value	Token
0xef373ec45d431ed...	2 hrs 8 mins ago	0x3f2dd0cb25bbf89...	OUT	0x521c526d5b50de...	57,896,044,618,658,100,000,000,000...	Erc20
0xef373ec45d431ed...	2 hrs 8 mins ago	0x3f2dd0cb25bbf89...	OUT	0x4473c6396eba3d...	57,896,044,618,658,100,000,000,000...	Erc20
0xdd8e427ecb6926...	2 hrs 54 mins ago	0x3f2dd0cb25bbf89...	OUT	0x66f471fd1c471bb...	57,896,044,618,658,100,000,000,000...	Erc20
0xdd8e427ecb6926...	2 hrs 54 mins ago	0x3f2dd0cb25bbf89...	OUT	0x4473c6396eba3d...	57,896,044,618,658,100,000,000,000...	Erc20

proxyOverflow (CVE-2018-10376)

```
function transferProxy(address _from, address _to, uint256 _value, uint256 _fee,
    uint8 _v,bytes32 _r, bytes32 _s) public transferAllowed(_from) returns (bool){

    if(balances[_from] < _fee + _value) revert();

    uint256 nonce = nonces[_from];
    bytes32 h = keccak256(_from,_to,_value,_fee,nonce);
    if(_from != ecrecover(h,_v,_r,_s)) revert();

    if(balances[_to] + _value < balances[_to]
        || balances[msg.sender] + _fee < balances[msg.sender]) revert();
    balances[_to] += _value;
    Transfer(_from, _to, _value);

    balances[msg.sender] += _fee;
    Transfer(_from, msg.sender, _fee);

    balances[_from] -= _value + _fee;
    nonces[_from] = nonce + 1;
    return true;
}
```

TxHash:	0xa850ce29dac8cefd0cf3250486a8080ca32b0a94308b9ac3ac6639140e601344
TxReceipt Status:	Success
Block Height:	5497602 (3120 block confirmations)
TimeStamp:	12 hrs 50 mins ago (Apr-24-2018 01:17:50 PM +UTC)
From:	0x0e64779930d9a5165dd712c06d11632c58ff8a4d
To:	Contract 0x3ac6cb00f5a44712022a51fbace4c7497f56ee31
Token Transfer:	65,133,050,195,990,400,000,000,000,000,000,000,000,000,000,000,000,000,000,000.891004451135422463 ERC20 (M2C Mesh Network Token) from 0x24e62761adad4e... to → 0x24e62761adad4e... 50,659,039,041,325,800,000,000,000,000,000,000,000,000,000,000,000,000,000,000.693003461994217473 ERC20 (M2C Mesh Network Token) from 0x24e62761adad4e... to → 0x0e64779930d9a5...

multiOverflow **(CVE-2018-10706)**

```
function transferMulti(address[] _to, uint256[] _value) public returns (uint256 amount){
    require(_to.length == _value.length);
    uint8 len = uint8(_to.length);
    for(uint8 j; j<len; j++){
        amount += _value[j]*10**uint256(decimals);
    }
    require(balanceOf[msg.sender] >= amount);
    for(uint8 i; i<len; i++){
        address _toI = _to[i];
        uint256 _valueI = _value[i]*10**uint256(decimals);
        balanceOf[_toI] += _valueI;
        balanceOf[msg.sender] -= _valueI;
        emit Transfer(msg.sender, _toI, _valueI);
    }
}
```

TxHash:	0x180dfd68ed78858679173819258ef0f582cb6d140f97342a4a38d9272c97d4c8
TxReceipt Status:	Success
Block Height:	5549398 (3592 block confirmations)
TimeStamp:	14 hrs 53 mins ago (May-03-2018 01:44:03 PM +UTC)
From:	0x7634dc2c0d3a48b00ee67789db40d9a365ff564c
To:	Contract 0xb75a5e36cc668bc8fe468e8f272cd4a0fd0fd773
Token Transfer:	578,960,446,186,581,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000 ERC20 (Social Chain Token) from 0x7634dc2c0d3a48... to → 0x93995bc9db9ae7... 578,960,446,186,581,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000 ERC20 (Social Chain Token) from 0x7634dc2c0d3a48... to → 0x0c2a5f9a88bf246...

burnOverflow **(CVE-2018-11239)**

```
function _transfer(address _from, address _to, uint _value) internal {
  /* Prevent transfer to 0x0 address. Use burn() instead  */
  require (_to != 0x0);
  /* Check if the sender has enough */
  require (balanceOf[_from] >= _value + burnPerTransaction);
  /* Check for overflows */
  require (balanceOf[_to] + _value > balanceOf[_to]);
  /* Subtract from the sender */
  balanceOf[_from] -= _value + burnPerTransaction;
  /* Add the same to the recipient */
  balanceOf[_to] += _value;
  /* Apply transaction fee */
  balanceOf[0x0] += burnPerTransaction;
  /* Update current supply */
  currentSupply -= burnPerTransaction;
  /* Notify network */
  Burn(_from, burnPerTransaction);
  /* Notify network */
  Transfer(_from, _to, _value);
}
```

TimeStamp:	1 hr 19 mins ago (May-18-2018 12:55:06 PM +UTC)
From:	0xaabc9616535c60a327c7f3531e219f9069c88b15
To:	Contract 0xb5335e24d0ab29c190ab8c2b459238da1153ceba (HexagonToken)
Token Transfer:	11,579,208,923,731,600,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000.9934 Hexagon Token from 0xaabc9616535c60... to → 0x3331270881cf775...

Contents

- Blockchain Background
- Smart Contracts Security Incidents
- Smart Contracts Security Solutions
- Our Solution to Integer Overflow
- Conclusion

Vulnerability Discovery

- Code Review *(10%?)*
- Fuzzing *(80%?)*
- Static Analysis
- Dynamic Analysis
- Taint Analysis
- Symbolic Execution
- Formal Verification

Code Review

EOS Out-of-Bound (by Qihoo 360)

```
72      call_indirect_table_type table;
73      table.resize(module->table.initial);
74      for (auto& segment : module->table.segments) {
75         Address offset = ConstantExpressionRunner<TrivialGlobalManager>(globals).visit(segment.offset).value.geti32();
76         assert(offset + segment.data.size() <= module->table.initial);
77         for (size_t i = 0; i != segment.data.size(); ++i) {
78            table[offset + i] = segment.data[i];
79         }
80      }
```

***assert()* works in debug mode only, NOT in release mode.**

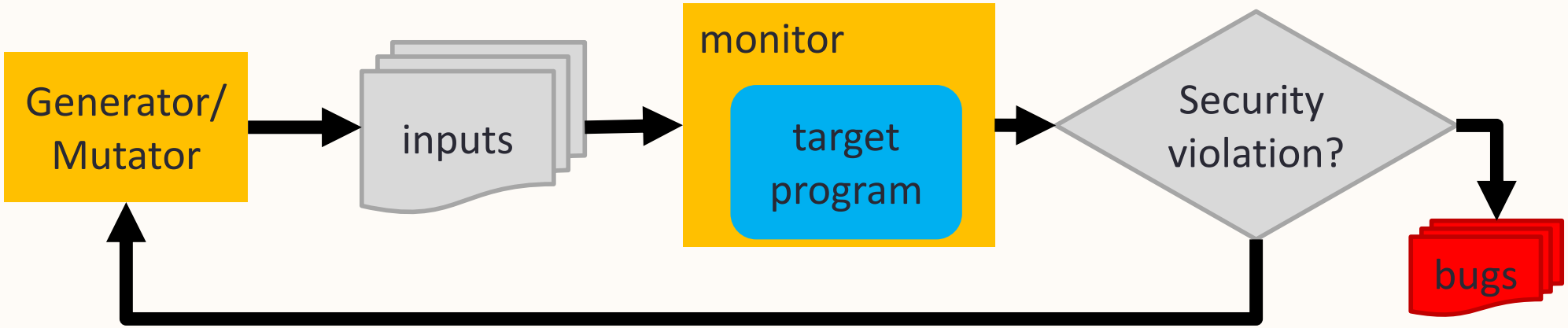
```
commit ea89dce21d13d41a22b3512a27be97b4be9df755
Author: Daniel Larimer <dan.larimer@block.one>
Date:   Mon May 28 13:21:33 2018 -0400

    fix asserts that get compiled out in debug

diff --git a/libraries/chain/webassembly/binaryen.cpp b/libraries/chain/webassembly/binaryen.cpp
index 60a590789..05765f13a 100644
--- a/libraries/chain/webassembly/binaryen.cpp
+++ b/libraries/chain/webassembly/binaryen.cpp
@@ -73,7 +73,7 @@ std::unique_ptr<wasm_instantiated_module_interface> binaryen_runtime::instantiat
       table.resize(module->table.initial);
       for (auto& segment : module->table.segments) {
          Address offset = ConstantExpressionRunner<TrivialGlobalManager>(globals).visit(segment.offset).value.geti32();
-         assert(offset + segment.data.size() <= module->table.initial);
+         FC_ASSERT(offset + segment.data.size() <= module->table.initial);
          for (size_t i = 0; i != segment.data.size(); ++i) {
             table[offset + i] = segment.data[i];
          }
```

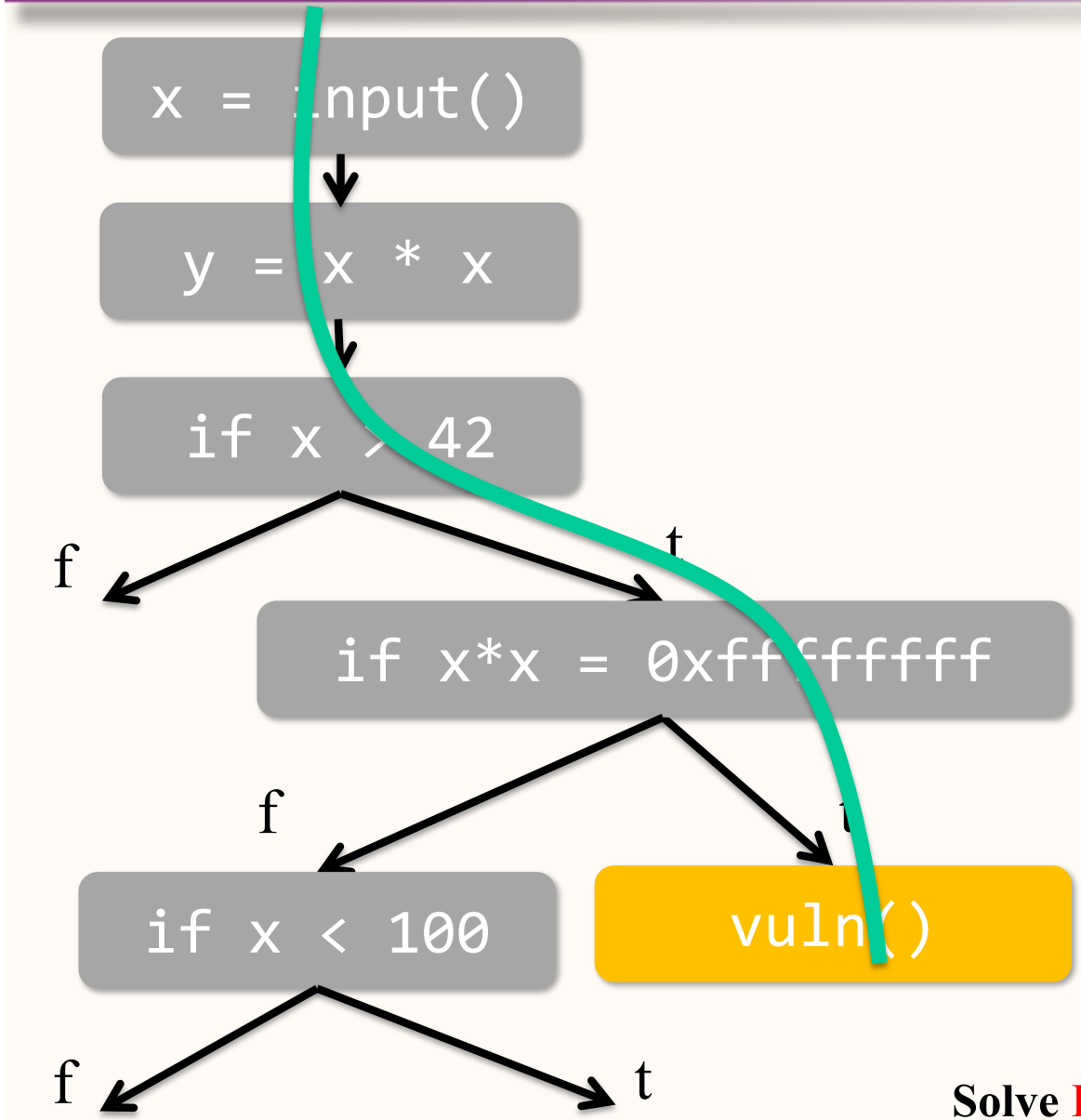
Fuzzing

- Testing and Monitoring



- Effective for traditional applications, but
- Ineffective for smart contracts
  - hard to monitor smart contracts
  - hard to define security violations

Symbolic Execution



x takes arbitrary value

y has a symbolic expression

Path constraints

$x > 42$

Path constraints

$(x > 42) \wedge (x*x == 0xffffffff)$

Vulnerability Constraints

Solve Path_Constraints ^ Vul_Constraints

Formal Methods

What security properties should hold?

Formal Specification

- Propositional Logic
- Predicate Logic
- Temporal Logic

How to prove the property holds?

Formal Verification

- Theorem Proving
- Abstract Interpretation
- Model Checking

Security Analysis Tools

Tools which are available. There can be paid services to use full features.

Name	Available Path	Features
Oyente	http://oyente.melon.fund	symbolic execution
Securify	http://securify.ch	formal verification
Remix	http://remix.ethereum.org	solidity compiler, debugger
SmartCheck	http://tool.smartdec.net	static code analysis
Mythril	https://github.com/ConsenSys/mythril	concolic and taint analysis
why3	http://why3.lri.fr/try/	formal verification, general tool

Contents

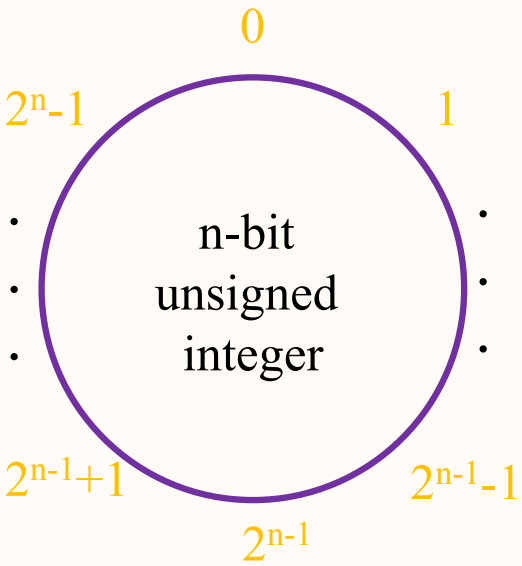
- Blockchain Background
- Smart Contracts Security Incidents
- Smart Contracts Security Solutions
- Our Solution to Integer Overflow
- Conclusion

What is Integer Overflow?

Root Cause of Integer Overflow

In Math

- Integer has unlimited range
  - $(-\infty, +\infty)$
- Never overflow



In Computer

- Integer has limited range
  - uint32_t: [0, 1, …, 2^{32}-1]
  - uint256_t: [0, 1, …, 2^{256}-1]
- Wrap if exceeds bounds
  - uint32_t i= 2^{32}-1;
  - uint32_t j = 1;
  - i + j = 0

Solution to Integer Overflow

- Ideally: validate all arithmetic operations
  - Automatically enforced by tools
  - Challenges: benign overflows
    - TCP sequence
    - Hash/Crypto computation
    - Modulo operation

- In practice, integer overflow are validated by developers.

Fixing Integer Overflow: confusing

A wrong patch released by Dillo v2.1, against a vulnerability in v2.0 found by IntScope, CVE-2009-2294

```
Png_datainfo_callback(png_structp png_ptr, png_infop info_ptr)
{
    png_get_IHDR(png_ptr, info_ptr,
                 &bit_depth, &colo
```

Intention: set the upper bound of width*height
Reality: width*height might overflow
This check is useless

```
+   /* check max image size */
+   if ( abs( png->width*png->height ) > IMAGE_MAX_W * IMAGE_MAX_H) {
+      Error();
+   }

    row_pointers = (uchar_t **) dMalloc(png->height * sizeof(uchar_t *));
    png->linebuf = dMalloc(3 * png->width);
```

Fixing Integer Overflow: error-prone

A wrong patch released by CUPS v1.3.9

CVE-2008-1722

```
    png_get_IHDR(pp, info, &width, &height, …);   // read from file

    img->xsize = width;                            // propagate
    img->ysize = height;

-   in = malloc(img->xsize * img->ysize * 3);

+   {
+     bufsize = img->xsize * img->ysize * 3;
+     if ((bufsize / (img->ysize * 3)) != img->xsize)
+         fprintf(stderr, "…" );
+   }
+   in = malloc(bufsize);
```

Original version
xsize*ysize*3 might overflow

Patched version
ysize*3 might overflow, leading to xsize*ysize*3 still overflows
This patch is not sound.

Fixing Integer Overflow: tedious

A correct patch released by CUPS v1.4.0

To check whether xsize*ysize*3 overflows correctly, 2 steps are needed.

```
    png_get_IHDR(pp, info, &width, &height, ...);   // read from file
+   if (width > CUPS_IMAGE_MAX_WIDTH )   {fprintf(stderr, "..."); }
    img->xsize = width;
    img->ysize = height;

    in = malloc(img->xsize * img->ysize * 3);       // overflow occurs

    {
      bufsize = img->xsize * img->ysize * 3;
-     if ((bufsize / (img->ysize * 3)) != img->xsize)   // incorrect patch
+     if ((bufsize / (img->xsize * 3)) !=
          fprintf(stderr, "...");
    }
    in = malloc(bufsize);
```

Check xsize first to make sure that X=xsize*3 won't overflow

Then, check whether ysize*X overflows

Fixing Integer Overflow: compiler-sensitive

```
#include <sys/types.h>
#include <stdio.h>

void foo(ssize_t x) {
    if (x>=0 && x+x<0)
        printf("Overflow\n");
}

main() {
    ssize_t x=0x7FFFFFFF;
    foo(x);
}
```

This intention of this check:
To capture integer overflow in expression x+x.
(x+x overflows when x is in [0x40000000, 0x7FFFFFFF])

The reality is quite different:
positive + positive = negative? So, this statement is discarded by some compiler optimizations, e.g. GCC.
Thus, it fails to capture overflows in expression x+x.

python™ Python is a victim of this GCC feature.

Our Solution to Integer Overflow

Symbolic Execution + Taint Analysis

- Symbolic Execution:
  - Path constraints + Vulnerability Constraints
  - constraint solving

- Vulnerability Constraints: integer overflow
  - e.g., x + y <= x && y > 0

- Problem:
  - too many false positives, due to benign overflows

Symbolic Execution + Taint Analysis

- Taint Analysis
  - taint source: inputs
  - taint propagation: how inputs affect internal variables
  - taint sink: sensitive locations

- Symbolic Execution + Taint Analysis
  - Path constraints + Vulnerability Constraints
  - (overflow) taint value flows into taint sinks
    - e.g., storage variables, external function calls
  - constraint solving
  - Low false positives

Automated Scanning

- Etherscan.io 390 contracts
  - 25 tokens have integer overflow vulnerabilities

overBuy (CVE-2018-11809)

```
function buyTokensICO() public payable onlyInState(State.ICORunning)
{
    // min - 0.01 ETH
    require(msg.value >= ((1 ether / 1 wei) / 100));
    uint newTokens = msg.value * getPrice();

    require(totalSoldTokens + newTokens <= TOTAL_SOLD_TOKEN_SUPPLY_LIMIT);

    balances[msg.sender] += newTokens;
    supply+= newTokens;
    icoSoldTokens+= newTokens;
    totalSoldTokens+= newTokens;

    LogBuy(msg.sender, newTokens);
}
```

overMint (CVE-2018-11809)

```
// Mint few tokens and transfer them to some address.
function mint(address _holder, uint256 _value) external icoOnly {
  require(_holder != address(0));
  require(_value != 0);
  require(totalSupply + _value <= tokenLimit);

  balances[_holder] += _value;
  totalSupply += _value;
  Transfer(0x0, _holder, _value);
}
```

mintAny (CVE-2018-11812)

```
function mintToken(address target, uint256 mintedAmount) onlyOwner {
    balanceOf[target] += mintedAmount;
    totalSupply += mintedAmount;
    Transfer(0, this, mintedAmount);
    Transfer(this, target, mintedAmount);
}
```

allocateAny (CVE-2018-11810)

```
/// @param _address The address of the recipient
/// @param _amount Amount of the allocation
/// @param _type Type of the recipient. 0 for advisor, 1 for founders.
/// @return Whether the allocation was successful or not
function allocate(address _address, uint256 _amount, uint8 _type) public onlyOwner returns (bool success) {
    // one allocations by address
    require(allocations[_address] == 0);

    if (_type == 0) { // advisor…
    } else if (_type == 1) { // founder…
    } else {
        // check allocated amount
        require(holdersAllocatedAmount + _amount <= HOLDERS_AMOUNT + RESERVE_AMOUNT);
        // increase allocated amount
        holdersAllocatedAmount += _amount;
    }
    // set allocation
    allocations[_address] = _amount;
    initialAllocations[_address] = _amount;

    // increase balance
    balances[_address] += _amount;
```

underSell (CVE-2018-11811)

```
/// @notice Sell `amount` tokens to contract
/// @param amount amount of tokens to be sold
function sell(uint256 amount) {
    require(this.balance >= amount * sellPrice);      // checks if the contract has enough ether to buy
    _transfer(msg.sender, this, amount);              // makes the transfers
    msg.sender.transfer(amount * sellPrice);          // sends ether to the seller.
                                                      // It's important to do this last to avoid recursion attacks
}
```

ownerUnderflow (CVE -2018-11687)

```
function distribute(address[] addresses) onlyOwner {
    for (uint i = 0; i < addresses.length; i++) {
        balances[owner] -= 2000 * 10**8;
        balances[addresses[i]] += 2000 * 10**8;
        Transfer(owner, addresses[i], 2000 * 10**8);
    }
}
```

Contents

- Blockchain Background
- Smart Contracts Security Incidents
- Smart Contracts Security Solutions
- Our Solution to Integer Overflow
- Conclusion

Conclusion

- Blockchain is an emerging technique
- Blockchain has a large attack surface, and smart contract is one of the most vulnerable targets.
- There are many smart contract disasters. Researchers have proposed several solutions to detect vulnerabilities.
- We have proposed a new solution to discover integer overflow with low false positives.

Thanks!

Q&A

Hack for Fun

Program Analysis

* source/binary analysis
* AI-based analysis

Vulnerability Discovery

* 100+ CVE
* 4 prototype system

Security

Proactive Defense

* ~10 defense solutions
* Microsoft BlueHat Prize

Vulnerability Exploit

* Automated exploit generation
* DEFCON CTF #2 (2016)