智能合约安全编程指南

@施懿民

智能合约 – 对于新手

- 使用remix进行智能合约开发：https://remix.ethereum.org/
- 准备以太私有网络测试智能合约开发
  - geth init genesis.json --datadir blocks
  - geth --datadir blocks –nodiscover
- 使用CLI挖矿：
  - geth attach ipc:\\.\pipe\geth.ipc
- 使用钱包链接测试网络：
  - "Ethereum Wallet.exe" --node geth --gethpath D:\research\eth\cui\Geth\geth.exe --node-attach ipc:\\.\pipe\geth.ipc

智能合约简单体验

address _

```
1  pragma solidity ^0.4.0;
2
3  contract MyContract {
4      uint myVariable;
5
6      constructor() public payable {
7          myVariable = 5;
8      }
9
10     function setMyVariable(uint value) public {
11         myVariable = value;
12     }
13
14     function getMyVariable() constant public returns(uint) {
15         return myVariable;
16     }
17
18     function getMyContractBalance() constant public returns(uint) {
19         return this.balance;
20     }
21
22     function () payable public {
23     }
24 }
```

智能合约安全不容忽视

- 智能合约可能有经济价值，自带金融属性。
- 智能合约的bug无法打补丁
  - 一旦智能合约被部署，代码就无法变更
  - 除非硬分叉
- 区块链交易无法回滚
  - 中心化数据库很容易回滚
  - 但区块链上，即使是一个恶意交易也无法回滚

智能合约安全举例 – 整数溢出

- Uint256也是有取值范围的，取值范围是：0 ~ 115792089237316195423570985008687907853269984665640564039457584007913129639935
- 溢出的话值就反转
- 解决方案：使用SafeMath

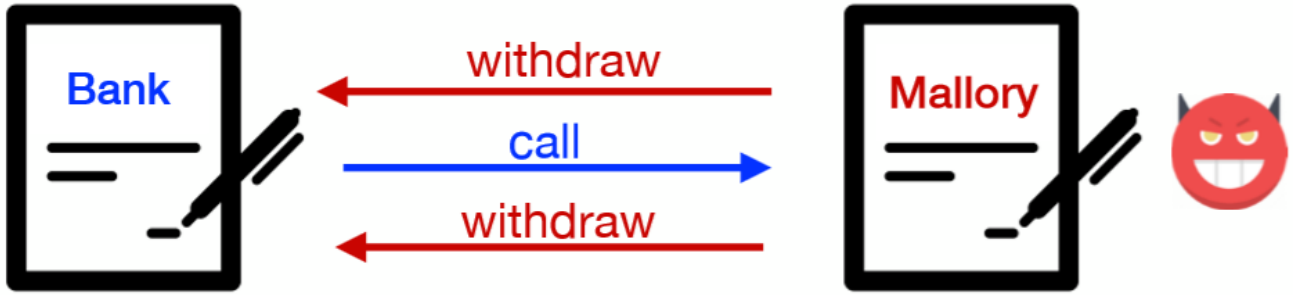
智能合约安全举例 – 编程疏忽

- 下面的代码有什么问题？

```
1  contract Rubixi {
2    address private owner;
3    function DynamicPyramid() { owner = msg.sender; }
4    function collectAllFees() { owner.send(collectedFees); }
5    ...
```

智能合约安全举例 – 重入攻击

- 在以太坊，当有函数调用时：
  - 调用者（Caller）需要等待被调用的方法（Callee）执行完毕
  - 一个恶意的被执行方法可以利用这点




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

```
function() {
 bank.withdraw(bank.queryCredit(this));
}
```

DEMO：攻击演示

- 重放BEC溢出攻击
- 重入攻击