Move：为 DeFi 而生的智能合约语言

Starcoin 邓启明

Agenda

Starcoin

01

智能合约 & 安全

Starcoin

智能合约 & 安全

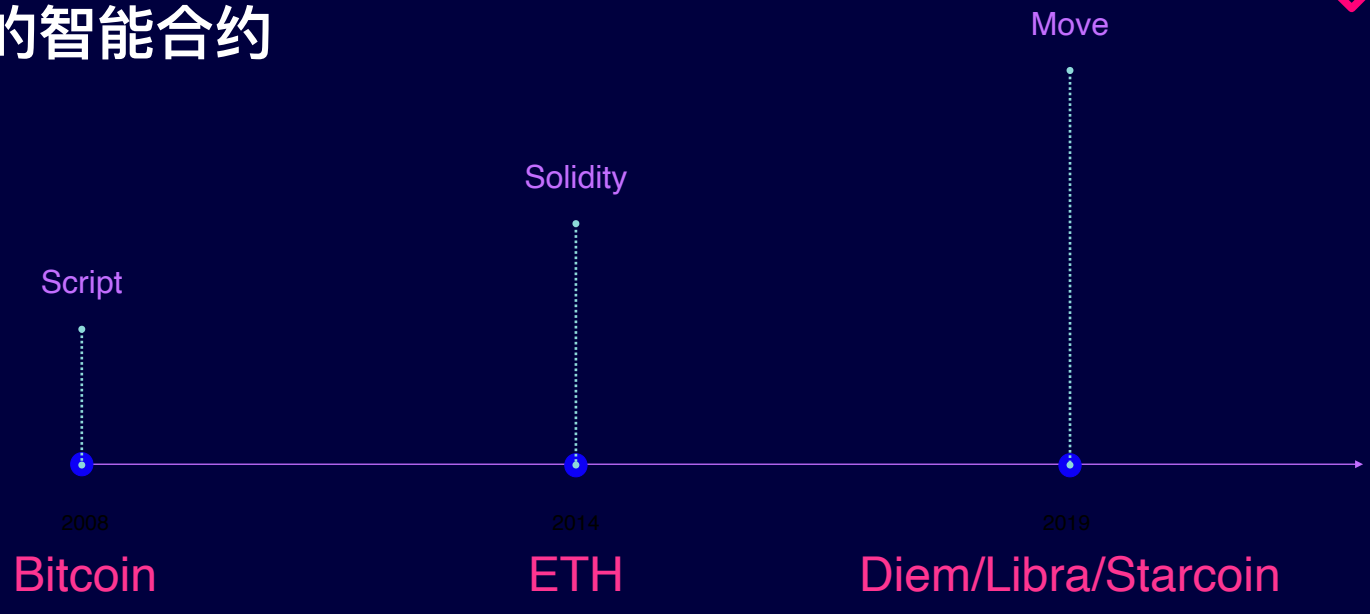
1. 进化的智能合约

2. 真实的安全漏洞

3. 引发的思考

Starcoin

进化的智能合约



Starcoin

真实的安全漏洞

1. ERC20的安全隐患

2. 默认可见性的安全隐患

Starcoin

ERC20的安全隐患

ERC20 Token不要使用transfer传输到一个合约地址，否则会导致代币丢失

Starcoin

ERC20不是一等公民

```
function () payable public { ... } // ETH的合约转账
```

```
function transfer(address _to, uint256 _value) public {
    _transfer(msg.sender, _to, _value);
}    // ERC20的合约转账
```

Starcoin

默认可见性

* Parity钱包

```
contract WalletLibrary is WalletEvents {

  ...

  // METHODS

  ...

  // constructor is given number of sigs required to do protected
"onlymanyowners" transactions
  // as well as the selection of addresses capable of confirming them.
  function initMultiowned(address[] _owners, uint _required) {
    m_numOwners = _owners.length + 1;
    m_owners[1] = uint(msg.sender);
    m_ownerIndex[uint(msg.sender)] = 1;
    for (uint i = 0; i < _owners.length; ++i)
    {
      m_owners[2 + i] = uint(_owners[i]);
      m_ownerIndex[uint(_owners[i])] = 2 + i;
    }
    m_required = _required;
  }

  ...

  // constructor - just pass on the owner array to the multiowned and
  // the limit to daylimit
  function initWallet(address[] _owners, uint _required, uint _daylimit) {
    initDaylimit(_daylimit);
    initMultiowned(_owners, _required);
  }
}
```

Starcoin

默认可见性

1. 默认Public可见性

2. 攻击者可以调用函数将合约的owner重置为自己

引发的思考

* 语言表达能力很强，缺乏对金融场景的支持

* 对开发者有很多隐性要求

Starcoin

02

为 DeFi 而生的 Move

为 DeFi 而生的 Move

1. Move基础
2. 安全的Rust
3. 从安全的Rust到Move
4. 定义更安全的Token
5. Move的其他安全特性

Starcoin

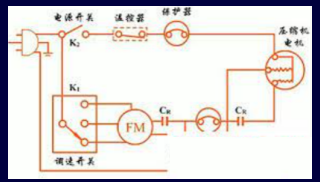
① Move基础

- **Module & Struct**
- **Ability**
- **Visibility**
- Ref & Mut Ref
- 泛型
- 形式化验证
- Unit test

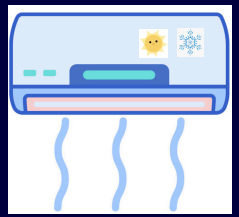
Starcoin

Module & Struct & Instance

Module



Struct



Instance

Starcoin

Example

```
module Factory {  // 1. Module
    struct AirConditioner { // 2. Struct
        start_button: bool,
    }

    public fun new() : AirConditioner{  // 3. instance
        AirConditioner {
            start_button: false,
        }
    }

    public fun drop(instance: AirConditioner) {  // 4. instance
        let AirConditioner{start_button: _} = instance;
    }
}

module ModuleTest {
    use {{alice}}::Factory;

    public fun foo() {
        let instance = Factory::new();   // 5. instance
        Factory::drop(instance);
    }
}
```

Starcoin

② 安全的Rust

* 所有权
* 引用

Starcoin

安全的Rust

* Instance所有权
* Immut Ref & Mut Ref

Starcoin

Rust的所有权

```
fn do_test(_data: Vec<u64>) {// 1 要求所有权
    // data，必须_，意味着data立即被回收
}

fn test_move() { // 2
    let data: Vec<u64> = Vec::new();
    do_test(data);// move 语义，所有权转移到1
}

fn test_copy() { // 3
    let data: Vec<u64> = Vec::new();
    do_test(data.clone());// u64自带copy能力
    drop(data) // 所有权还在，明确销毁
}
```

Starcoin

引用

* Immut Ref： 只读，同时可以有多个

* Mut Ref：可以修改，同时只能存在1个

Rust的资源操作

* move or copy

* Immut ref or Mut ref

Starcoin

③从安全的Rust到Move

* Rust到Move
* Ability
* Struct & has
* badcase

Starcoin

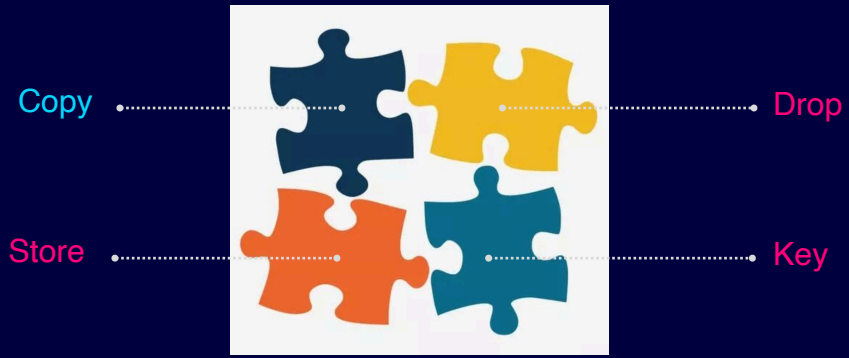
从安全的Rust到Move

* Ability

* Immut ref or Mut ref

Starcoin

Ability



Starcoin

Ability

* has

```
struct Test has key, copy, drop, store { }  // has
```

Starcoin

badcase 1

```
MyToken1 has drop { //1
    amount: u128
}

let _ = MyToken1::new(10);   //凭空消失
```

Starcoin

badcase 2

```
MyToken2 has copy { // 1
    amount: u128
}

let coins1 = MyToken2::new(10);  // 2
let coins2 = copy coins1; // 3 无限增发
```

Starcoin

④ 定义更安全的Token

* MyToken
* Account
* 一等公民

Starcoin

MyToken

```
module Token {
    Token has store {   // 1. 定义 Token，不能 copy\drop
        amount: u128
    }

    public fun withdraw(token:&mut Token, amount: u128): Token {  // 2. 取款
        token.amount -= amount;
        Token {
          amount
        }
    }

    public fun deposit(to_token:&mut Token, check: Token) {   // 3. 存款
        let Token { amount } = check;
        token.amount = token.amount + amount;
    }
}
```

1. Token只能store

Starcoin

Account

```
module Account {
    use 0x1::Token;
    use 0x1::Singer;

    struct Account has key {   // 4. 账号
        balance: Token,
    }

    public fun transfer(account: &singer, receiver: address, amount: u128) {  // 5. 转账
        let myself = Singer::address_of(account);
        let balance = borrow_global_mut<Account>(myself).balance;  // 6. 获取账号 mut instance
        let coins = Token::withdraw(balance, amount); // 7. 取款
        let receiver_balance = borrow_global_mut<Account>(receiver).balance; // 8. 获取 mut
        Token::deposit(receiver_balance, coins);  // 9. 存款
    }
}
```

1. 操作别人账号的Token时，只能增加，不能减少
2. 转移Token所有权，不能修改(增发)，也不能丢弃

Starcoin

一等公民MyToken

1. 不能丢弃
2. 不能增发
3. 只能存储
4. 一视同仁

Starcoin

⑤ Move的其他安全特性

- 函数可见性
- 没有delegatecall
- 形式化验证spec
- 泛型
- Unit test

Starcoin

Function Visibility

- Public
- Private
- **Friend**
- Script

Visibility Example

```
address 0x2 {
  module A {
    // friend declaration via fully qualified module name
    friend 0x2::B;  // 6

    // friend declaration via module alias
    use 0x2::C;
    friend C;  // 7

    fun i_am_private() {    // 1
      // other functions in the same module can also call friend functions
      bar();
    }

    public fun i_am_public() {   // 2
      // other functions in the same module can also call friend functions
      bar();
    }

    public(script) fun i_am_script() {}   // 3

    public(friend) fun bar() {}    // 4

    public(friend) fun foo() {   // 5
      // a friend function can call other non-script functions in the same module
      i_am_private();
      i_am_public();
      bar();
    }
  }
}
```

```
address 0x2 {
  module B {
    use 0x2::A;

    public fun foo() {
      // as a friend of 0x2::A, functions in B can call friend functions in A
      A::foo();    // 8
    }

    public fun bar() {
      0x2::A::bar();   // 9
    }
  }
}
```

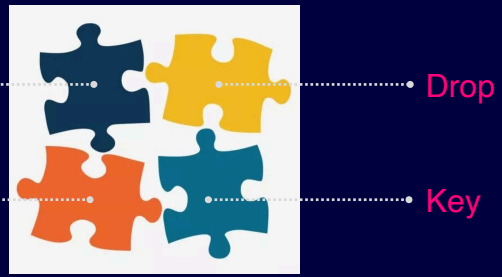
Starcoin

默认可见性

* Private（默认可见性）
* Friend（授权访问）

Starcoin

总结



\+ 可见性

* 简单易学安全

Starcoin

Move线上黑客松预告

* Starcoin介绍
* Move线上黑客松
* 更多Move知识讲座

Starcoin

Starcoin：新一代Libra

- PoW
- 分层
- 状态计费
- Move
- Stdlib
- 完备的 DAO 链上治理能力

Starcoin

Move黑客松

* 线上
* 2021年6月23日0点 —— 2021年7月31日24点
* Move-cli

Starcoin

更多Move直播

* Move基础语法，直播时间：2021年6月30日
* Move进阶，直播时间：2021年7月7日
* Move开发实战，直播时间：2021年7月14日
* Move & DeFi实战，直播时间:2021年7月21日

Starcoin

招聘

* 研究源码、了解技术原理

* 写技术文章、参加技术会议，做技术科普

Starcoin

Q & A

Starcoin