ФИНТЕХ АССОЦИАЦИЯ

SecDevOps – как «заставить» программистов делать безопасный код

ПРАКТИКА НАПИСАНИЯ СМАРТ-КОНТРАКТОВ. ПРИМЕРЫ УЯЗВИМОСТЕЙ. ИЗВЕСТНЫЕ ПРЕЦЕДЕНТЫ.

```
// Пример цифрового актива на Мастерчейн
contract Token {

  // Создатель токена (эмитент)
  address public owner;

  // Инициализация токена
  function Token() public {
    owner = msg.sender;
  }

  // Расчётная книга
  mapping(address => uint) public balance;

  // Эмитент может выпустить активы
  function produce(uint amount) public {
    require (msg.sender == owner);
    balance[owner] += amount;
  }

  // Любой владелец актива может сделать перевод
  function transfer(uint amount, address to) public {
    require (balance[msg.sender] >= amount);
    balance[msg.sender] -= amount;
    balance[to] += amount;
  }
}
```

Web3 interface at http://localhost:20000

CONNECT

0: 0xd7eb66f1743f56665e4455f2531dd267e2e005b9 unlocked

1: 0xd2361be81854f49a9c165e890f0b7ed13428cf3d

Loaded compiler soljson-v0.4.21+commit.dfe3193c.js

COMPILE

Token bytecode:

```
60 60 60 40 52 34 15 61 00 0f 57 60 00 80 fd 5b 60 00 80 54 60 01 60
a0 60 02 0a 03 33 16 60 01 60 a0 60 02 0a 03 19 90 91 16 17 90 55 61
01 e0 80 61 00 3b 60 00 39 60 00 f3 00 60 60 60 40 52 ... (539 bytes)
```

DEPLOY

Token instance at 0x9a07d9dc3bbae837f0546a88552551c0aa37ece2

OPEN

Token.owner(): 0xd7eb66f1743f56665e4455f2531dd267e2e005b9

```
// Пример цифрового актива на Мастерчейн
contract Token {

  // Создатель токена (эмитент)
  address public owner;

  // Инициализация токена
  function Token() public {
    owner = msg.sender;
  }

  // Расчётная книга
  mapping(address => uint) public balance;

  // Эмитент может выпустить активы
  function produce(uint amount) public {
    require (msg.sender == owner);
    balance[owner] += amount;
  }

  // Любой владелец актива может сделать перевод
  function transfer(uint amount, address to) public {
    require (balance[msg.sender] >= amount);
    balance[msg.sender] -= amount;
    balance[to] += amount;
  }
}
```

Примеры уязвимостей:

1. Компрометация ключа владельца – что делать?
2. Логические ошибки в коде – как обновить?

Контракт состоит из двух частей:

из состояния и логики выполнения.

Традиционные механизмы предотвращения атак предназначены для обновления состояния либо логики.

Оператор сети Мастерчейн требует от разработчиков приложений поддерживать отключение и обновление и состояния и логики выполнения контрактов.

ФИНТЕХ
АССОЦИАЦИЯ

SecDevOps – как «заставить» программистов делать безопасный код

ПРИМЕНЕНИЕ ИЗМЕНЕНИЙ В РАСПРЕДЕЛЕННЫХ РЕЕСТРАХ. ПОДХОДЫ К ОБНОВЛЕНИЯМ, МОДЕЛЬ КОНСИСТЕНТНОСТИ.

```
// Контракт с возможностью блокировки
contract Suspendable {

    // Флаг отключения
    bool public disabled;

    // Проверка: отключен ли контракт
    modifier enabled() {
        require(disabled == false);
        _;
    }

    // Проверка прав администратора
    function canDisable()
    public view returns(bool);

    // Заблокировать
    function disable() public
    enabled {
        require(canDisable());
        disabled = true;
    }

    // Разблокировать
    function enable() public {
        require(disabled == true)
        require(canDisable());
        disabled = false;
    }
}
```

```
// Контракт с возможностью обновления
contract Upgradable is Suspendable {

    // Адрес предыдущей версии
    Upgradable public precursor;

    // Адрес следующей версии
    Upgradable public successor;

    // Инициализировать с указанием
    // адреса предыдущей версии
    function Upgradable(Upgradable _precursor) public {
        if (_precursor != address(0))
            _precursor.upgradeWith(this);
        precursor = _precursor;
    }

    /// @notice Указать заместителя и заблокировать
    function upgradeWith(Upgradable _successor) public {
        disable();
        successor = _successor;
    }
```