Безопасность ICO контрактов (4)

Александр Половьян
alex@ledgers.world

Замечание о модификаторах функций

	пишут в чейн	читают из чейна	deprecated	нужен gas
-	yes	yes	no	yes
pure	no	no	no	no
view	no	yes	no	no
constant	no	yes	yes	no

Замечание о модификаторах функций

If invalid explicit type conversions are used, **state modifications are possible even though a pure function was called**.

	пишут в чейн	читают из чейна	deprecated	нужен gas
-	yes	yes	no	yes
pure	no	no	no	no
view	no	yes	no	no
constant	no	yes	yes	no

Хранение данных

- storage
  Структура данных которая задается под контракт в момент вызова конструктора. Структура не меняется, значения в ней могут меняться во время транзакций.

- memory
  На время выполнения кода.

- event log

- Список транзакций

- *constant* переменные — использование по значению во время компиляции

Хранение данных

- storage – разреженный mapping
- memory – массив с ячейками по 32 байта, не чистится
- event log – отдельный индекс
- Список транзакций – блок
- *constant* переменные – runtime bytecode

Использовать короткие типы данных?

- **Спойлер: скорее всего, нет**

- Упаковка для примитивных типов может быть компактной (меньше gas за хранение данных)

- Стоимость операций с компактно упакованными данными будет выше из-за того что EVM приходится совершать больше действий для работы со значениями

- Порядок объявления переменных помогает компилятору оптимизировать память

Events

- Индексированные записи в *transaction log*
- Минимальный индекс — адрес смарт-контракта создавшего запись
- Смарт-контракты только пишут, но не читают
- Зачем?
    1. Облегчают интеграцию сторонних приложений
    2. Дешевый способ записи информации

Стоимость

Запись в лог	375 gas	
Отправитель	free (?)	Адрес смарт-контракта
Топик (log topic, индекс)	375 gas за каждый топик	4 макс (ограничение EVM)
Данные (последовательность байт)	8 gas за 1 байт	**YellowPaper:** не ограничено **Solidity:** 32 байта макс
1 ячейка памяти смарт-конракта	~20 000 gas за 1 ячейку	~5 000 gas за перезапись
Параметр транзакции	68 gas за 1 байт	

Events

```
contract SocialConnection {
    SocialAccount public from;
    SocialAccount public to;
    event FriendAdded (address indexed from, address to);

    function SocialConnection(SocialAccount _from, SocialAccount _to) public {
        from = _from;
        to = _to;
        emit FriendAdded(from, to);
    }

    function Unfriend () public {
        require(msg.sender == address(from) || msg.sender == address(to));
        selfdestruct(msg.sender);
    }
}
```

Ограничение вызова

- require — валидация входных параметров
- assert — не должен происходить никогда (обращение к несуществующему индексу массива)
- throw – deprecated

	revert chain state	refund unused gas	deprecated
require	yes	yes	no
assert	yes	no	no
throw	yes	yes	yes

Модификаторы функций

```
contract SocialAccount () {
    address owner;
    …
    function addFriend(SocialAccount _a) {
        require(msg.sender == owner);
        …
    }

    function removeFriend(SocialAccount _a) {
        require(msg.sender == owner);
        …
    }
}
```

```
contract SocialAccount () {
    address owner;
    …
    function addFriend(SocialAccount _a) onlyOwner {

        …
    }

    function removeFriend(SocialAccount _a) onlyOwner {

        …
    }

    modifier onlyOwner {
        require(msg.sender == owner);
        _;
    }
}
```

Возвращение нескольких значений

```
 function complexAdd(uint256 _re, uint256 _im)
   public pure returns (uint256 re, uint256 im)
{
     re = _re + 1;
     im = _im + 1;
 }

 function complexAdd2(uint256 _re, uint256 _im)
   public pure returns (uint256 re, uint256 im)
{
     return (_re + 1,  _im + 1);
 }

 function complexAdd3(uint256 _re, uint256 _im)
   public pure returns (uint256, uint256)
{
     return (_re + 1,  _im + 1);
 }
```

Возвращение нескольких значений

```
function combine2(uint256 _a, uint256 _b) public pure returns(uint256, uint256) {
    uint256 a; uint256 b;
    (a, b) = complexAdd(_a, _b);
    (a, b) = complexAdd2(a, b);
    (a, b) = complexAdd3(a, b);
    return (a, b);
}

function combine(uint256 _a, uint256 _b) public pure returns(uint256, uint256) {
    return complexAdd2(complexAdd3(_a, _b)); // WILL NOT COMPILE
}
```

Почти ООП

Наследование

```
pragma solidity ^0.4.21;

contract SocialEntity {
    address public owner;

    function SocialEntity () public {
        owner = msg.sender;
    }
}

contract SocialAccount is SocialEntity {

}

contract PartyEvent is SocialEntity {

}
```

Чуть более полезное наследование

```
pragma solidity ^0.4.21;

contract SocialEntity {
    enum EntityType { None, Account, Party }
    address public owner;
    EntityType public myType;

    function SocialEntity (EntityType _t) public {
        owner = msg.sender;
        myType = _t;
    }
}

contract SocialAccount is SocialEntity(SocialEntity.EntityType.Account) {

}

contract PartyEvent is SocialEntity(SocialEntity.EntityType.Party) {

}
```

Абстрактность

```
pragma solidity ^0.4.21;

contract SocialEntity {
    address public owner;

    function SocialEntity () public {
        owner = msg.sender;
    }

    function ThisMethodMakesMeAbstract () public;
}
```

Интерфейсы

```
pragma solidity ^0.4.11;

interface Token {
    function transfer(address recipient, uint amount) public;
}
```

Библиотеки

- Паттерны кода которые доступны внутри контракта
- Нет состояния
- Не наследуют и не наследуют
- Не получают ETH
- Частый паттерн: `using SafeMath for uint256`

Тип данных: функция

```
function map(
    uint[] memory self,
    function (uint) pure returns (uint) f
) internal pure returns (uint[] memory r) {

  r = new uint[](self.length);
  for (uint i = 0; i < self.length; i++) {
    r[i] = f(self[i]);
  }
}
```

Вызов сторонних методов

```
pragma solidity ^0.4.21;

contract Ping {

    function a() public view returns (address) {
        return this;
    }

    function b(Ping _p) public view returns (address) {
        return _p.a();
    }

}
```

Использование ETH

- Модификатор `payable`
  разрешает функции принимать ETH
- Добавляем value в любую транзакцию
- Добавляем value в вызов конструктора
- Fallback function
  код запускается при получении ETH без указания функции

Использование ETH

```
pragma solidity ^0.4.21;

contract MyWallet {
    address public owner;

    function MyWallet() payable public {
        owner = msg.sender;
    }

    function AmIRich() public view returns (uint256) {
        return address(this).balance;
    }

    function () public payable {

    }
}
```

Обращение ETH

```
pragma solidity ^0.4.21;

contract MyWallet {
    ...
    function Withdraw () public {
        require(msg.sender == owner);
        msg.sender.transfer(address(this).balance);
    }

}
```

Обращение ETH (2)

```
pragma solidity ^0.4.21;

contract MyWallet {
    ...
    function Withdraw (address _payee) public {
        require(msg.sender == owner);
        if (_payee == 0x0) {
            _payee = owner;
        }
        _payee.transfer(address(this).balance);
    }
}
```

Несколько вариантов

add.call.value(x)()			
addr.transfer(x)			
add.send(x)			
майнинг (beneficiary)			
selfdestruct(addr)			