Безопасность ICO контрактов (3)

Александр Половьян
alex@ledgers.world

Смарт-контракт

- Разработка социальной сети

- Один аккаунт — один смарт-контракт

Пустой контракт

```
pragma solidity ^0.4.21;

contract SocialAccount {

}
```

Добавляем данные

```
pragma solidity ^0.4.21;

contract SocialAccount {
    string public name;
    string public phone;

}
```

- state variables

Типы данных

```
pragma solidity ^0.4.21;

contract SocialAccount {
    string public name;
    string public phone;
    bytes public photo;

}
```

- bool
- int / uint / uint256
- address — 0x20BYTES
- byte / bytes / bytes32 /
- string

Структуры данных

- enum Direction { Up, Down }
- struct Elevator {
    address caller;
    Direction moving;
  }
- array address[]
  - push
  - length
- mapping (address => Elevator)
  - Ø default value
  - stores only hash of data for lookup
- mapping (address => mapping (address => uint256))

- `delete` — assigns default

Модификаторы видимости данных

```
pragma solidity ^0.4.21;

contract SocialAccount {
    string public name;
    string public phone;
    bytes public photo;

}
```

- private
- public — default getter function

Конструктор

```
pragma solidity ^0.4.21;

contract SocialAccount {
    string public name;

    function SocialAccount (string _name) public {
        name = _name;
    }

}
```

Сеттеры

```
pragma solidity ^0.4.21;

contract SocialAccount {
    string public name;

    function SocialAccount (string _name) public {
        name = _name;
    }

    function SetName (string _name) public {
        name = _name;
    }

}
```

Видимость функций

```
pragma solidity ^0.4.21;

contract SocialAccount {
    string public name;

    function SocialAccount (string _name) public {
        name = _name;
    }

    function SetName (string _name) public {
        name = _name;
    }

}
```

	доступна снаружи контракта	доступна внутри контракта	наследуется
external	yes	no	yes
public	yes	yes	yes
internal	no	yes	yes
private	no	yes	no

Ограничение вызова

```
pragma solidity ^0.4.21;

contract SocialAccount {
    string public name;
   address public owner;

    function SocialAccount (string _name) public {
        name = _name;
        owner = msg.sender;
    }

    function SetName (string _name) public {
        require(msg.sender == owner);
        name = _name;
    }

}
```

Друзья

```
pragma solidity ^0.4.21;

contract SocialAccount {
    address[] public friends;

    ....

    function AddFriend (SocialAccount _address) public {
        require(msg.sender == owner);
        friends.push(_address);
    }
}
```

Getter для массива

```
pragma solidity ^0.4.21;

contract SocialAccount {
    address[] public friends;

    ....

    function AddFriend (SocialAccount _address) public {
        require(msg.sender == owner);
        friends.push(_address);
    }
}
```

Все друзья

```
pragma solidity ^0.4.21;

contract SocialAccount {

    address[] public friends;

    ....

    function AddFriend (SocialAccount _address) public {
        require(msg.sender == owner);
        friends.push(_address);
    }

    function AllFriends () public returns (address[]) {
        return friends;
    }

}
```

Как должны работать друзья?

- Связь между двумя аккаунтами
- Возможность разорвать связь по инициативе одного из контрагентов
- Максимум одна связь в паре
- Связь с собой невозможна

Критика технической реализации

- Удаление неочевидно
- Поиск линейно усложняется
- Проверить, является ли кто-нибудь другом?

Друзья 2

```
pragma solidity ^0.4.21;

contract SocialAccount {

    mapping(SocialAccount => bool) public friends;

    function AddFriend (SocialAccount _address) public {
        friends[_address] = true;
    }

   function RemoveFriend (SocialAccount _address) public {
        delete friends[_address];
    }

}
```

Друзья 2

```
pragma solidity ^0.4.21;

contract SocialAccount {

    mapping(SocialAccount => bool) public friends;

    function AddFriend (SocialAccount _address) public {
        friends[_address] = true;
    }

   function RemoveFriend (SocialAccount _address) public {
        delete friends[_address];
    }

    function MutualFriend (SocialAccount _address) public returns (bool) {
        return friends[_address] && _address.friends(this);
    }

}
```

Друзья 2

```
pragma solidity ^0.4.21;

contract SocialAccount {

    ...

    function MutualFriend
        (SocialAccount _address)
        public view returns (bool)
    {
        return friends[_address] &&
            _address.friends(this);
    }

}
```

	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

Друзья 2

```
pragma solidity ^0.4.21;

contract SocialAccount {

    mapping(SocialAccount => bool) public friends;

    function AddFriend (SocialAccount _address) public {
        friends[_address] = true;
    }

   function RemoveFriend (SocialAccount _address) public {
        delete friends[_address];
    }

    function MutualFriend (SocialAccount _address) public returns
(bool) {
        return friends[_address] && _address.friends(this);
    }

}
```

- Как сделать список всех друзей?

Друзья 3

```
pragma solidity ^0.4.21;

contract SocialAccount {
    ...
}

contract SocialConnection {
    SocialAccount from;
    SocialAccount to;

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

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