Security Audit Report of IFT Token Smart Contract

D-D-S
Moscow
2018

Table of contents:

1. Limitations of usage

This report contains information on discovered vulnerabilities, their severity and methods of exploiting those vulnerabilities, discovered in the process of the audit.

This report is given «AS IS» with no legal guarantees.

2. Initial conditions

Contract implements a kind of joint-stock type company with ability of shareholders to vote and distribute Ethereum coins based on their IFT token share.

We reviewed next files:

- **`arbitrage.sol`** - main contract
- **`Erc20.sol`** - ERC-20 compatibility contract
- **`Ownable.sol`** - contract, which provides basic authorization control functions
- **`SafeMath.sol`** - contract implementing math operations with safety checks that revert on error

3. **Overall recommendations**

`arbitrage.sol`

Overall, the contract well written and easy readable. But we have spotted code in comment sections - seems that it has been used in test purposes. We recommend to delete this code before deploying to avoid any possible errors.

Also we recommend to use the latest solc compiler version (this contract has 0.4.24, and latest actual is 0.5.1). Using SafeMath with for all uint256 operations in the code could also improve its robustness.

We also recommend to use built-in Ethereum time aliases to improve readability and reduce memory usage, e.g. writing **`k*days`**, where k is day count.

4.1 Vulnerabilities review

`arbitrage.sol`
1:

Code of main contract doesn't meet the customer's requirements of administrator not owning the tokens: it is possible to add contract address as shareholder, and the contract already has an owner, which is administrator.

`addArbiter` function checks only for owner's address, but doesn't perform check on contract address itself (`account != owner`).

```
264     function addArbiter(address account,uint256 tokens)
265         require(currentState != VOTING);
266         require(currentState != OKVOTING);
267         require(account != owner);   //restrict owner fr
268         require(_voted[account] == 0);  //check if arbit
```

`addArbiter` doesn't have any other constrains that could limit this vulnerability.

Same problems encounter `issueTokens` function, so it is possible to add tokens on contract address.

```
5     function issueTokens(address account, uint256 tokens)
6         require(currentState != VOTING);
7         require(account != address(0));
8         require(_voted[account] != 0);
```

4.2 Vulnerabilities review

`arbitrage.sol`
`Ownable.sol`
2:

The main contract has an owner, which can be changed by a function `transferOwnership` call, because arbitrage contract uses contract Ownable.sol as a base.

The attack follows next steps: current contract owner can add a fake shareholder to the holders list, issue enough tokens on that address and then transfer ownership to it. In next step original ownership is being transferred to that fake address, so in the result, the owner of contract will have tokens and ownership.

Also, if the voting result will be OK, the owner could easily transfer Ethereum coins out, because transfer functions do not have any additional restrictions.

5. Conclusion and recommendations

We recommend to re-implement several checks in shareholder-adding and token-issuing functions. It is also recommended to add some constrains on transferring Ethereum coins out.

We also recommend to improve the ownership functionality or remove it totally.

In case of fixing only one of these problems, there could be a ‘locking’ scenario: actual owner could pass the ownership to any new shareholder, but the latest couldn’t use his tokens because of added restrictions.

We also recommend to improve code using advices on page 3.

D-D-S for IFT Token
Moscow 2018