FASTTOKEN SMART CONTRACT AUDIT REPORT

24.10.2022

CONTENTS

SUMMARY

SEVERITY	NUMBER OF FINDINGS
CRITICAL	0
HIGH	1
MEDIUM	1
LOW	2
INFORMATIONAL	3

TOTAL: 7

SCOPE

The analyzed resources are located on:
https://github.com/fasttoken1/fasttoken-distribution-eth-contracts/commit/8c6dc446f99233154c39aec789c2b08deafac165

The issues described in this report were fixed in the following commit:
https://github.com/fasttoken1/fasttoken-distribution-eth-contracts/commit/56dd4ee95f995046cdbdb7ac2e6c658c54cf1a64

WEAKNESSES

This section contains the list of discovered weaknesses.

1. INCORRECT OWNER CHANGE LOGIC IN MULTISIG

SEVERITY: High

PATH: MultiSigWallet.sol:25

REMEDIATION: invert conditions by respectively removing and adding the negation operator, so they match logically with the revert reason

STATUS: fixed

DESCRIPTION:

The conditions at lines 126-127 are both inverted (but not their “reason”, which is correct), making owner change executions revert when they should pass and vice versa. Since these requirements directly contradict the conditions imposed when the transaction was submitted (lines 92-93), this leaves the contract without any means to update owner addresses.

```
if (TransactionType.ChangeOwner == transaction.transactionType) {
            require(! isOwner[transaction.address1], 'address1 must be owner');
            require(isOwner[transaction.address2], 'address2 cannot be
owner');
            uint256 index = _ownerIndex(transaction.address1);
            owners[index] = transaction.address2;
            isOwner[transaction.address1] = false;
            isOwner[transaction.address2] = true;
        }
```

2. IMPROVE EVENTS

SEVERITY: Medium

PATH: MultiSigWallet.sol

REMEDIATION: improve the events to contain the caller’s address as well

STATUS: acknowledged

DESCRIPTION:

The following event signatures contain only transaction index information and don't show the address that invoked the action. Not having an invoker address logged in the event rises a possibility for phishing attacks on owner change mechanisms; as one can start a vote to change other owner’s public key.

```
emit SubmitTransaction(txIndex);
emit ConfirmTransaction(_txIndex);
emit ExecuteTransaction(_txIndex);
emit RevokeConfirmation(_txIndex);
```

3. REDUNDANT VIEW METHODS

SEVERITY: Low

PATH: MultiSigWallet.sol:158, 168

REMEDIATION: change variable visibility from public to internal, or delete the getter method to save on gas

STATUS: acknowledged

DESCRIPTION:

The listed methods duplicate the functionality provided by the compiler for all public variables, which include a getter in the ABI, making them or the custom getters redundant.

```
  function getOwners() public view returns (address[] memory) {

      return owners;
  }
...
  function getTransactions() public view returns (Transaction[] memory) {

      return transactions;
  }
```

4. STRUCTURES USED WITH ARRAYS

SEVERITY: Low

PATH: MultiSigWallet.sol:25

REMEDIATION: use a mapping instead of an array, add an index array if necessary

STATUS: acknowledged

DESCRIPTION:

Arrays and structs don't play well together in Solidity, but in most cases using a mapping solves the issue and offers efficiency gains.

```
address[] public owners;
mapping(address => bool) public isOwner;
uint256 public numberOfRequiredConfirmations;

// mapping from transaction index => owner index => bool
mapping(uint256 => mapping(uint256 => bool)) public isConfirmed;

Transaction[] public transactions;
```

5. GAS OPTIMISATION

SEVERITY: Informational

PATH: MultiSigWallet.sol

REMEDIATION: change the visibility to external

STATUS: acknowledged

DESCRIPTION:

The methods listed below are not being used locally, their visibility is excessive.

MultiSigWallet.sol

- submitTransaction():L82
- confirmTransaction():L101
- executeTransaction():L116
- revokeConfirmation():L141
- getOwners():L158
- getTransactionCount():L163
- getTransactions():L168

6. FLOATING PRAGMA

SEVERITY: Informational

PATH: MultiSigWallet.sol, FasttokenDistribution.sol, Fasttoken.sol

REMEDIATION: modify the pragma to enforce a version at least equal to the latest stable compiler version

STATUS: fixed

DESCRIPTION:

The pragma is defined with a caret (^), allowing any version higher than **0.8.0** to compile the code. For the branch that will contain the code to be deployed, it is recommended to use a fixed pragma (without caret), to ensure compilations are consistent across time and dev environments, and contract verification is made easier.

```
pragma solidity ^0.8.0;
```

7. GRIEVING BY OWNERS

SEVERITY: Informational

PATH: MultiSigWallet.sol:141

REMEDIATION: to entirely avoid this scenario, at the expense of slower execution speed, introduce a period for confirmation and a period for revoke, then execute transactions only after revoke period

STATUS: acknowledged

DESCRIPTION:

Owners are considered friendly unless compromised. In this case, hostile owners are able to sign transactions, making it seem like it can be executed, then frontrun the execution transaction with a cancel confirmation tx, getting below threshold and failing the execution.

This is only slightly different from grieving by refusing to sign, and is therefore rated only informational.

```
function revokeConfirmation(uint256 _txIndex)
        public
        onlyOwner
        transactionExists(_txIndex)
        notExecuted(_txIndex) {

     Transaction storage transaction = transactions[_txIndex];

     uint256 index = _ownerIndex(msg.sender);
     require(isConfirmed[_txIndex][index], 'transaction not confirmed');

     transaction.numberOfConfirmations -= 1;
     isConfirmed[_txIndex][index] = false;

     emit RevokeConfirmation(_txIndex);
  }
```

hexens