慢雾科技
slow mist

Smart Contract Security Audit Report

The SlowMist Security Team received the MyCreditChain team's application for smart contract security audit of the MCC on February 19, 2020. The following are the details and results of this smart contract security audit:

Token name :

MCC

File name and HASH(SHA256) :

ManagedToken.sol:

0309813e7d0d52142d2759831e5475702c89e1207022bda5ec6968361c9d16a7

TransferLimitedToken.sol:

b905167df8ec335a45394f5c2bc2c0650ce564c7dc35de545ae99c3d5750ace7

MCCToken.sol:

5b5249f9c525e243746706d9ef7046614e2a827b6852c7b71da89d8ad3ba1d1f

The audit items and results :

(Other unknown security vulnerabilities are not included in the audit responsibility scope)

No.	Audit Items	Audit Subclass	Audit Subclass Result
1	Overflow Audit	-	Passed
2	Race Conditions Audit	-	Passed
3	Authority Control Audit	Permission vulnerability audit	Passed
		Excessive auditing authority	Passed
4	Safety Design Audit	Zeppelin module safe use	Passed
		Compiler version security	Passed
		Hard-coded address security	Passed
		Fallback function safe use	Passed
		Show coding security	Passed
		Function return value security	Passed
		Call function security	Passed
5	Denial of Service Audit	-	Passed

6	Gas Optimization Audit	-	Passed
7	Design Logic Audit	-	Passed
8	“False Deposit” vulnerability Audit	-	Passed
9	Malicious Event Log Audit	-	Passed
10	Scoping and Declarations Audit	-	Passed
11	Replay Attack Audit	ECDSA's Signature Replay Audit	Passed
12	Uninitialized Storage Pointers Audit	-	Passed
13	Arithmetic Accuracy Deviation Audit	-	Passed

Audit Result : **Passed**

Audit Number : 0X002002240002

Audit Date : February 24, 2020

Audit Team : SlowMist Security Team

Summary: This is a token contract that does not contain the tokenVault section. OpenZeppelin's SafeMath security module is used, which is a commendable approach. The contract does not have the Overflow and the Race Conditions issue. The comprehensive evaluation contract is no risk.

The source code:

ManagedToken.sol:

```
// Klaytn IDE uses solidity 0.4.24, 0.5.6 versions.

//SlowMist// The contract does not have the Overflow and the Race Conditions issue

pragma solidity >=0.4.24 <=0.6.0;
```

```
import "@openzeppelin/contracts/ownership/Ownable.sol";
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/token/ERC20/ERC20Detailed.sol";

/**
 * @title ManagedToken
 * @dev ERC20 compatible token with issue and destroy facilities
 * @dev All transfers can be monitored by token event listener
 */
contract ManagedToken is ERC20, ERC20Detailed, Ownable {
    bool private _allowTransfer = false;

    event AllowTransfersChanged(bool _state);

    modifier transfersAllowed() {
        require(_allowTransfer);
        _;
    }

    /**
     * @dev ManagedToken constructor
     * @param _owner: Owner
     */
    constructor(string memory _name, string memory _symbol, uint8 _decimals, address _owner) public
ERC20Detailed(_name, _symbol, _decimals) {
                    _transferOwnership(_owner);
    }

    /**
     * @dev Enable/disable token transfers. Can be called only by owners
     * @param _state: true - allow / false - disable
     */
    function setAllowTransfers(bool _state) external onlyOwner returns (bool) {
        _allowTransfer = _state;
        emit AllowTransfersChanged(_allowTransfer);
        return true;
    }

    function allowTransfer() public view returns (bool) {
        return _allowTransfer;
    }
    /**
     * @dev Override transfer function.
     */
    function transfer(address _to, uint256 _value) public transfersAllowed returns (bool) {
        bool success = super.transfer(_to, _value);

        return success; //SlowMist// The return value conforms to the EIP20 specification

    }

    /**
     * @dev Override transferFrom function. Add event listener condition
     */
    function transferFrom(address _from, address _to, uint256 _value) public transfersAllowed returns (bool) {
        bool success = super.transferFrom(_from, _to, _value);

        return success; //SlowMist// The return value conforms to the EIP20 specification

    }
}
```

TransferLimitedToken.sol:

```
// Klaytn IDE uses solidity 0.4.24, 0.5.6 versions.

//SlowMist// The contract does not have the Overflow and the Race Conditions issue

pragma solidity >=0.4.24 <=0.6.0;

import "./ManagedToken.sol";

/**
 * @title TransferLimitedToken
 * @dev Token with ability to limit transfers within wallets included in limitedWallets list for certain period of time
 */
contract TransferLimitedToken is ManagedToken {
    uint8 public constant LIMITED_NONE = 0;
    uint8 public constant LIMITED_SENDER = 1;
    uint8 public constant LIMITED_RECEIVER = 2;
    uint8 public constant LIMITED_ALL = 3;

    mapping(address => bool) public limitedSenderWallets;
    mapping(address => bool) public limitedReceiverWallets; // TODO: how to get keys(addesses) only without other logic?

    /**
     * @dev Check if transfer between addresses is available
     * @param _from: From address
     * @param _to: To address
     */
    modifier canTransfer(address _from, address _to)   {
        require((!limitedSenderWallets[_from]  && !limitedReceiverWallets[_to]), "One or both addresses are limited for
transfering");
        _;
    }

    /**
     * @dev TransferLimitedToken constructor
     * @param _owner: Owner
     */
    constructor(string memory _name, string memory _symbol, uint8 _decimals, address _owner) public
        ManagedToken(_name, _symbol, _decimals, _owner)
    {}

    /**
     * @dev Add address to limitedWallets by owner
     * @param _wallet: limitation target
     * @param _targetStatus: limitation level
     * @param _registration: true / false
     */
    function setLimitedWalletAddress(address _wallet, uint8 _targetStatus, bool _registration) public onlyOwner returns
(bool) {
        require(LIMITED_SENDER <= _targetStatus && LIMITED_ALL >= _targetStatus);
        if (_targetStatus >= LIMITED_RECEIVER) {
            limitedReceiverWallets[_wallet] = _registration;
            _targetStatus -= LIMITED_RECEIVER;
        }
        if (_targetStatus == LIMITED_SENDER) {
            limitedSenderWallets[_wallet] = _registration;
        }
        return true;
    }

    function isLimitedWalletAddress(address _wallet) public view returns(uint8) {
        uint8 targetStatus = LIMITED_NONE;
        if (limitedSenderWallets[_wallet]) {
            targetStatus += LIMITED_SENDER;
        }
        if (limitedReceiverWallets[_wallet]) {
            targetStatus += LIMITED_RECEIVER;
        }
        return targetStatus;
    }

    /**
     * @dev Override transfer function. Add canTransfer modifier to check possibility of transferring
     */
    function transfer(address _to, uint256 _value) public canTransfer(msg.sender, _to) returns (bool) {
        return super.transfer(_to, _value);
    }

    /**
     * @dev Override transferFrom function. Add canTransfer modifier to check possibility of transferring
     */
    function transferFrom(address _from, address _to, uint256 _value) public canTransfer(_from, _to) returns (bool) {
        return super.transferFrom(_from, _to, _value);
    }

    /**
     * @dev Override approve function. Add canTransfer modifier to check possibility of transferring
     */
    function approve(address _spender, uint256 _value) public canTransfer(msg.sender, _spender) returns (bool) {
        return super.approve(_spender,_value);
    }
}
```

MCCToken.sol:

```
// Klaytn IDE uses solidity 0.4.24, 0.5.6 versions.
```

//SlowMist// The contract does not have the Overflow and the Race Conditions issue

```
pragma solidity >=0.4.24 <=0.6.0;

import "./token/TransferLimitedToken.sol";
import "@openzeppelin/contracts/math/SafeMath.sol";

contract MCCToken is TransferLimitedToken {
    //
==========================================================================
==================================
    //                                    Members
    //
```

```
===============================================================================
==================================
    using SafeMath for uint256;

    address private _tokenManager;
    address private _seedPublisher;

    mapping(address => bool) public multiTransferSenderWallets;

    event SetOwner(address indexed _owner);
    event SetSeedPublisher(address indexed _seedPublisher);
    event Goodmorn(uint256 _refKey, uint256 _from, uint256 _to, uint8 _seed, string _penalty, string _senderCountry, string
_sendTime);

    modifier onlyManager() {
        require(msg.sender == _tokenManager, "msg.sender is not token manager");
        _;
    }

    modifier onlySeedPublisher() {
        require(msg.sender == _seedPublisher, "msg.sender is not seed publisher");
        _;
    }

    modifier canMultiTransfer(address _sender)   {
        require(multiTransferSenderWallets[_sender], "the sender address is not on multiTransferSenderWallets");
        require(!limitedSenderWallets[_sender], "the sender address is limited");
        _;
    }

    //
===============================================================================
==================================
    //                                        Constructor
    //
===============================================================================
==================================

    /**
     * @dev MCC Token
     */
    constructor(uint256 initialSupply, string memory _name, string memory _symbol, uint8 _decimals, address
_exchangeMaster, address _owner, address _manager) public
        TransferLimitedToken(_name, _symbol, _decimals, _owner)
    {
        _tokenManager = _manager;
                    _mint(_exchangeMaster, initialSupply.mul(10 ** uint256(_decimals)));
    }

    function () external payable {}

    function goodmorn(uint256 _refKey, uint256 _from, uint256 _to, uint8 _seed, string calldata _penalty, string calldata
_senderCountry, string calldata _sendTime) onlySeedPublisher external payable {
        emit Goodmorn(_refKey, _from, _to, _seed, _penalty, _senderCountry, _sendTime);
    }

    function refund() external onlyOwner {
        _transfer(address(this), msg.sender, address(this).balance);
    }

    /**
     * @dev set token owner who can set token limitation
     * @param _owner: token owner
     */
    function setOwner(address _owner) external onlyManager returns (bool) {
        _transferOwnership(_owner);
        emit SetOwner(_owner);
        return true;
    }

    function manager() public view returns (address) {
        return _tokenManager;
    }

    /**
     * @dev set seed publisher who can set seed publisher
     * @param _publisher: seed publisher
     */
    function setSeedPublisher(address _publisher) external onlyOwner returns (bool) {
        _seedPublisher = _publisher;
        emit SetSeedPublisher(_publisher);
        return true;
    }
    function seedPublisher() public view returns (address) {
        return _seedPublisher;
    }

    /**
     * @dev set sender who who can use multiple transfer
     * @param _wallet: sender
     * @param _approval: true / false
     */
    function setMultiTransferSenderWalletAddress(address _wallet, bool _approval) public onlyOwner returns (bool) {
        multiTransferSenderWallets[_wallet] = _approval;
        return true;
    }

    function isMultiTransferSenderWalletAddress(address _wallet) public view returns (bool) {
        return multiTransferSenderWallets[_wallet];
    }

    /**
     * @dev transferMulti
     * @param _targets token receiver list
     * @param _values token amount to send
     */
    function    transferMulti(address[]    calldata    _targets,    uint256[]    calldata    _values)    external    transfersAllowed
canMultiTransfer(msg.sender) {
        uint256 totalValues = 0;
        for ( uint i = 0 ; i < _targets.length ; i++ ) {
            require(!limitedReceiverWallets[_targets[i]], "the receiver address is limited");
            totalValues = totalValues.add(_values[i]);
            require(balanceOf(msg.sender) >= totalValues, "the sender doesn't have enough balance");
        }

        for( uint i = 0 ; i < _targets.length ; i++ ) {
                                transfer(_targets[i], _values[i]);
        }
     }
}
```

慢雾科技
slow mist

Official Website
www.slowmist.com

E-mail
team@slowmist.com

Twitter
@SlowMist_Team

WeChat Official Account