Smart Contract Security Audit Report

The SlowMist Security Team received the TMC Token team's application for smart contract security audit of the TMC on June 10, 2020. The following are the details and results of this smart contract security audit:

Token name :

TMC

The Contract address :

0x1c153BADb7e54AbcdCB65f0A09fCd6f10dE36aA3

Link address :

https://etherscan.io/address/0x1c153BADb7e54AbcdCB65f0A09fCd6f10dE36aA3

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 : 0X002006120001

Audit Date : June 12, 2020

Audit Team : SlowMist Security Team

Summary: This is a token contract that does not contain the tokenVault section. The total amount of contract tokens remains unchanged. The Owner can add any user to the blacklist using the BlackListing function. The contract does not have the Overflow and the Race Conditions issue. The transfer and transferFrom functions return values of 'true' or 'false', with the risk of `False top-up`.

The source code:

```
/**
 *Submitted for verification at Etherscan.io on 2020-02-06
*/

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

pragma solidity ^0.4.21;

// ownership contract
contract Owned {
    address public owner;
```

```
    event TransferOwnership(address oldaddr, address newaddr);

    modifier onlyOwner() { if (msg.sender != owner) return; _; }

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

    function transferOwnership(address _new) onlyOwner public {
        address oldaddr = owner;
        owner = _new;
        emit TransferOwnership(oldaddr, owner);
    }
}

// erc20
contract ERC20Interface {
            uint256 public totalSupply;
            function balanceOf(address _owner) public constant returns (uint256 balance);
            function transfer(address _to, uint256 _value) public returns (bool success);
            function transferFrom(address _from, address _to, uint256 _value) public returns (bool success);
            function approve(address _spender, uint256 _value) public returns (bool success);
            function allowance(address _owner, address _spender) public constant returns (uint256 remaining);
            event Transfer(address indexed _from, address indexed _to, uint256 _value);
            event Approval(address indexed _owner, address indexed _spender, uint256 _value);
}

contract TMC_Contract is ERC20Interface, Owned {
            string public constant symbol = "TMC";
            string public constant name = "TMC Token";
            uint8 public constant decimals = 18;
            uint256 public constant totalSupply = 500000000000000000000000000;

            bool public stopped;

            mapping (address => int8) public blackList;

            mapping (address => uint256) public balances;
            mapping (address => mapping (address => uint256)) public allowed;
```

```
    event Blacklisted(address indexed target);
    event DeleteFromBlacklist(address indexed target);
    event RejectedPaymentToBlacklistedAddr(address indexed from, address indexed to, uint256 value);
    event RejectedPaymentFromBlacklistedAddr(address indexed from, address indexed to, uint256 value);


    modifier notStopped {
        require(!stopped);
        _;
    }

// constructor
            function TMC_Contract() public {
                        balances[msg.sender] = totalSupply;
            }

// function made for airdrop
            function airdrop(address[] _to, uint256[] _value) onlyOwner notStopped public {
                for(uint256 i = 0; i < _to.length; i++){
                    if(balances[_to[i]] > 0){
                        continue;
                    }
                    transfer(_to[i], _value[i]);
                }
            }

// blacklist management
    function blacklisting(address _addr) onlyOwner public {
        blackList[_addr] = 1;
        emit Blacklisted(_addr);
    }
    function deleteFromBlacklist(address _addr) onlyOwner public {
        blackList[_addr] = -1;
        emit DeleteFromBlacklist(_addr);
    }

// stop the contract
    //SlowMist// Suspending all transactions upon major abnormalities is a recommended approach
    function stop() onlyOwner {
        stopped = true;
    }
```

```
    function start() onlyOwner {
        stopped = false;
    }

// ERC20 functions
        function balanceOf(address _owner) public constant returns (uint256 balance){
                return balances[_owner];
        }

        //SlowMist// The function returns' true 'or' false ',with the risk of `False top-up`

        function transfer(address _to, uint256 _value) notStopped public returns (bool success){
                require(balances[msg.sender] >= _value);

                if(blackList[msg.sender] > 0){
                        emit RejectedPaymentFromBlacklistedAddr(msg.sender, _to, _value);
                        return false;
                }
                if(blackList[_to] > 0){
                        emit RejectedPaymentToBlacklistedAddr(msg.sender, _to, _value);
                        return false;
                }

                balances[msg.sender] -= _value;
                balances[_to] += _value;
                emit Transfer(msg.sender, _to, _value);

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

        }

        //SlowMist// The function returns' true 'or' false ',with the risk of `False top-up`

        //SlowMist// Unchecked whether `msg.sender` is on the blacklist

        function transferFrom(address _from, address _to, uint256 _value) notStopped public returns (bool success){
                require(balances[_from] >= _value
                        && allowed[_from][msg.sender] >= _value);

                if(blackList[_from] > 0){
                        emit RejectedPaymentFromBlacklistedAddr(_from, _to, _value);
                        return false;
                }
                if(blackList[_to] > 0){
                        emit RejectedPaymentToBlacklistedAddr(_from, _to, _value);
```

```
                return false;
            }

            balances[_from] -= _value;
            allowed[_from][msg.sender] -= _value;
            balances[_to] += _value;
            emit Transfer(_from, _to, _value);

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

        }
        function approve(address _spender, uint256 _value) notStopped public returns (bool success){
            allowed[msg.sender][_spender] = _value;
            emit Approval(msg.sender, _spender, _value);

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

        }
        function allowance(address _owner, address _spender) public constant returns (uint256 remaining){
            return allowed[_owner][_spender];
        }
}
```

慢雾科技
slow mist

Official Website
www.slowmist.com

E-mail
team@slowmist.com

Twitter
@SlowMist_Team

WeChat Official Account