Smart Contract Security Audit Report

The SlowMist Security Team received the REDi team's application for smart contract security audit of the REDi Token on November 22, 2019. The following are the details and results of this smart contract security audit:

Token name :

REDi

The File Name and HASH(SHA256)

IERC20.sol: 5c4e8b86da2ec9eb258972a7740a2707a7980fd4edd43a377e0cae0df6bc46ca

SafeMath.sol: a017a3139597ae9c2393492b8839f0e29135e440b6568477c12bde1598b6e1e3

REDiToken.sol: fb29a2a9619419489fc3e6d647e81c4bfb561821dc93e8ca8df25cff40b16613

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 : 0X001911260001

Audit Date : November 26, 2019

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 recommend approach. The total amount of the token can be changed, user can burn their token through burn and burnFrom functions. The comprehensive evaluation contract is no risk.

The source code:

IERC20.sol

```
pragma solidity ^0.5.2;

/**
 * @title ERC20 interface
 * @dev see https://eips.ethereum.org/EIPS/eip-20
 */
interface IERC20 {
    function transfer(address to, uint256 value) external returns (bool);

    function approve(address spender, uint256 value) external returns (bool);

    function transferFrom(address from, address to, uint256 value) external returns (bool);
```

```
    function totalSupply() external view returns (uint256);

    function balanceOf(address who) external view returns (uint256);

    function allowance(address owner, address spender) external view returns (uint256);

    event Transfer(address indexed from, address indexed to, uint256 value);

    event Approval(address indexed owner, address indexed spender, uint256 value);
}
```

SafeMath.sol

```
pragma solidity ^0.5.2;

/**
 * @title SafeMath
 * @dev Unsigned math operations with safety checks that revert on error
 */
```

//SlowMist// OpenZeppelin’s SafeMath security Module is used, which is a recommend approach

```
library SafeMath {
    /**
     * @dev Multiplies two unsigned integers, reverts on overflow.
     */
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b);

        return c;
    }

    /**
```

```
    * @dev Integer division of two unsigned integers truncating the quotient, reverts on division by zero.
    */
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        // Solidity only automatically asserts when dividing by 0
        require(b > 0);
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in which this doesn't hold

        return c;
    }

    /**
    * @dev Subtracts two unsigned integers, reverts on overflow (i.e. if subtrahend is greater than minuend).
    */
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b <= a);
        uint256 c = a - b;

        return c;
    }

    /**
    * @dev Adds two unsigned integers, reverts on overflow.
    */
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a);

        return c;
    }

    /**
    * @dev Divides two unsigned integers and returns the remainder (unsigned integer modulo),
    * reverts when dividing by zero.
    */
    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b != 0);
        return a % b;
    }
}
```

REDiToken.sol

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

```
pragma solidity ^0.5.2;

import "./IERC20.sol";
import "./SafeMath.sol";

contract REDiToken is IERC20 {

    using SafeMath for uint256;

    mapping (address => uint256) private _balances;
    mapping (address => mapping (address => uint256)) private _allowed;

    string public name = "REDiToken";
    string public symbol = "REDi";
    uint8 public decimals = 18;
    uint256 private _totalSupply = 10000000000 * 10 ** uint256(decimals);

    constructor() public {
      _balances[msg.sender] = _totalSupply;
    }

    function totalSupply() public view returns (uint256) {
      return _totalSupply;
    }

    function balanceOf(address owner) public view returns (uint256) {
      return _balances[owner];
    }

    function allowance(address owner, address spender) public view returns (uint256) {
      return _allowed[owner][spender];
    }

    function transfer(address to, uint256 value) public returns (bool) {
      _transfer(msg.sender, to, value);
      return true; //SlowMist// The return value conforms to the EIP20 specification
    }

    function approve(address spender, uint256 value) public returns (bool) {
```

```
    _approve(msg.sender, spender, value);
    return true; //SlowMist// The return value conforms to the EIP20 specification
  }

  function transferFrom(address from, address to, uint256 value) public returns (bool) {
    _transfer(from, to, value);
    _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));
    return true; //SlowMist// The return value conforms to the EIP20 specification
  }

  function burn(uint256 value) public {
    _burn(msg.sender, value);
  }

  function burnFrom(address from, uint256 value) public {
    _burnFrom(from, value);
  }

  function increaseAllowance(address spender, uint256 addedValue) public returns (bool) {
    _approve(msg.sender, spender, _allowed[msg.sender][spender].add(addedValue));
    return true;
  }

  function decreaseAllowance(address spender, uint256 subtractedValue) public returns (bool) {
    _approve(msg.sender, spender, _allowed[msg.sender][spender].sub(subtractedValue));
    return true;
  }

  function _transfer(address from, address to, uint256 value) internal {
    require(to != address(0)); //SlowMist// This kind of check is very good, avoiding user mistake
leading to the loss of token during transfer

    _balances[from] = _balances[from].sub(value);
    _balances[to] = _balances[to].add(value);
    emit Transfer(from, to, value);
  }

  function _burn(address account, uint256 value) internal {
```

```
        require(account != address(0));

        _totalSupply = _totalSupply.sub(value);
        _balances[account] = _balances[account].sub(value);
        emit Transfer(account, address(0), value);
    }

    function _approve(address owner, address spender, uint256 value) internal {
        require(spender != address(0));
        require(owner != address(0));

        _allowed[owner][spender] = value;
        emit Approval(owner, spender, value);
    }

    //SlowMist// Because burnFrom() and transferFrom() share the _allowed amount of
approve(), if the agent be evil, there is the possibility of malicious burn

    function _burnFrom(address account, uint256 value) internal {
        _burn(account, value);
        _approve(account, msg.sender, _allowed[account][msg.sender].sub(value));
    }
}
```

慢雾科技
slow mist

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

WeChat Official Account