Smart Contract Security Audit Report

The SlowMist Security Team received the BORA team's application for smart contract security audit of the BORA Token on October 29, 2019. The following are the details and results of this smart contract security audit:

Token name :

BORA

The Contract address :

0x26fb86579e371c7AEdc461b2DdEF0A8628c93d3B

Link address :

https://etherscan.io/address/0x26fb86579e371c7AEdc461b2DdEF0A8628c93d3B

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 : 0X001910310001

Audit Date : October 31, 2019

Audit Team : SlowMist Security Team

(**Statement :** SlowMist only issues this report based on the fact that has occurred or existed before the report is issued, and bears the corresponding responsibility in this regard. For the facts occur or exist later after the report, SlowMist cannot judge the security status of its smart contract. SlowMist is not responsible for it. The security audit analysis and other contents of this report are based on the documents and materials provided by the information provider to SlowMist as of the date of this report (referred to as "the provided information"). SlowMist assumes that: there has been no information missing, tampered, deleted, or concealed. If the information provided has been missed, modified, deleted, concealed or reflected and is inconsistent with the actual situation, SlowMist will not bear any responsibility for the resulting loss and adverse effects. SlowMist will not bear any responsibility for the background or other circumstances of the project.)

Summary: This is a token contract that contains the tokenVault section. The total amount of the token can be changed, user can burn their token via burn function. The donor can revoke the locked token locked in the LockedToken Contract in anytime when revocable is true. The contract does not have the Overflow and the Race Conditions issue. The comprehensive evaluation contract is no risk.

The source code:

```
/**
 *Submitted for verification at Etherscan.io on 2018-07-06
*/

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

pragma solidity ^0.4.18;


//SlowMist// SafeMath security module is used, which is a commendable approach

library SafeMath {
    function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
```

```
        if (a == 0) {
            return 0;
        }
        c = a * b;
        assert(c / a == b); //SlowMist// It is recommended to replace "assert" with "require" to optimize Gas
        return c;
    }

    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        return a / b;
    }

    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        assert(b <= a); //SlowMist// It is recommended to replace "assert" with "require" to optimize Gas
        return a - b;
    }

    function add(uint256 a, uint256 b) internal pure returns (uint256 c) {
        c = a + b;
        assert(c >= a); //SlowMist// It is recommended to replace "assert" with "require" to optimize Gas
        return c;
    }
}


contract Ownable {
    address public owner;

    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

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

```
    modifier onlyOwner() {
        require(msg.sender == owner);
        _;
    }

    function transferOwnership(address newOwner) public onlyOwner {
        require(newOwner != address(0)); //SlowMist// This check is quite good in avoiding losing control of the contract caused by user mistakes
        OwnershipTransferred(owner, newOwner);
        owner = newOwner;
    }
}

contract Pausable is Ownable {
    event Pause();
    event Unpause();

    bool public paused = false;

    modifier whenNotPaused() {
        require(!paused);
        _;
    }

    modifier whenPaused() {
        require(paused);
        _;
    }

    //SlowMist// Suspending all transactions upon major abnormalities is a recommended approach.
    function pause() onlyOwner whenNotPaused public {
        paused = true;
        Pause();
    }

    function unpause() onlyOwner whenPaused public {
        paused = false;
```

```
            Unpause();
        }
}

contract ERC20Basic {
    function totalSupply() public view returns (uint256);
    function balanceOf(address who) public view returns (uint256);
    function transfer(address to, uint256 value) public returns (bool);
    event Transfer(address indexed from, address indexed to, uint256 value);
}

contract ERC20 is ERC20Basic {
    function allowance(address owner, address spender) public view returns (uint256);
    function transferFrom(address from, address to, uint256 value) public returns (bool);
    function approve(address spender, uint256 value) public returns (bool);
    event Approval(address indexed owner, address indexed spender, uint256 value);
}

contract BasicToken is ERC20Basic {
    using SafeMath for uint256;
    mapping(address => uint256) balances;
    uint256 totalSupply_;

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

    function transfer(address _to, uint256 _value) public returns (bool) {

        require(_to != address(0)); //SlowMist// This kind of check is very good, avoiding user
mistake leading to the loss of token during transfer

        require(_value <= balances[msg.sender]);

        balances[msg.sender] = balances[msg.sender].sub(_value);
        balances[_to] = balances[_to].add(_value);
        Transfer(msg.sender, _to, _value);

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

    }

    function balanceOf(address _owner) public view returns (uint256 balance) {
```

```
        return balances[_owner];
    }
}

contract StandardToken is ERC20, BasicToken {
    mapping (address => mapping (address => uint256)) internal allowed;

    function transferFrom(address _from, address _to, uint256 _value) public returns (bool) {
        require(_to != address(0)); //SlowMist// This kind of check is very good, avoiding user
mistake leading to the loss of token during transfer
        require(_value <= balances[_from]);
        require(_value <= allowed[_from][msg.sender]);

        balances[_from] = balances[_from].sub(_value);
        balances[_to] = balances[_to].add(_value);
        allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
        Transfer(_from, _to, _value);
        return true; //SlowMist// The return value conforms to the EIP20 specification
    }

    function approve(address _spender, uint256 _value) public returns (bool) {
        allowed[msg.sender][_spender] = _value;
        Approval(msg.sender, _spender, _value);
        return true; //SlowMist// The return value conforms to the EIP20 specification
    }

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

    function increaseApproval(address _spender, uint _addedValue) public returns (bool) {
        allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(_addedValue);
        Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
        return true;
    }

    function decreaseApproval(address _spender, uint _subtractedValue) public returns (bool) {
        uint oldValue = allowed[msg.sender][_spender];
```

```
        if (_subtractedValue > oldValue) {
            allowed[msg.sender][_spender] = 0;
        } else {
            allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
        }
        Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
        return true;
    }
}

contract PausableToken is StandardToken, Pausable {
    function transfer(address _to, uint256 _value) public whenNotPaused returns (bool) {
        return super.transfer(_to, _value);
    }

    function transferFrom(address _from, address _to, uint256 _value) public whenNotPaused returns (bool) {
        return super.transferFrom(_from, _to, _value);
    }

    function approve(address _spender, uint256 _value) public whenNotPaused returns (bool) {
        return super.approve(_spender, _value);
    }

    function increaseApproval(address _spender, uint _addedValue) public whenNotPaused returns (bool success) {
        return super.increaseApproval(_spender, _addedValue);
    }

    function decreaseApproval(address _spender, uint _subtractedValue) public whenNotPaused returns (bool
success) {
        return super.decreaseApproval(_spender, _subtractedValue);
    }
}

contract BoraToken is PausableToken {
    string public name;
    string public symbol;
    uint8 public decimals;

    event Burn(address to, uint256 amount, uint256 totalSupply);
    event Lock(address token, address beneficiary, uint256 amount, uint256 releaseTime);

    function BoraToken(uint256 _supply) public {
```

```
        require(_supply != 0);
        balances[msg.sender] = _supply;
        totalSupply_ = _supply;
        name = 'BORA';
        symbol = 'BORA';
        decimals = 18;
        Transfer(address(0), msg.sender, _supply);
    }

    function lock(address _donor, address _beneficiary, uint256 amount, uint256 _duration, bool _revocable)
onlyOwner public returns (LockedToken) {
        uint256 releaseTime = now.add(_duration.mul(1 days));
```

//SlowMist// require(_donor != address(0));

//SlowMist// It's recommend to add the code above, avoiding user mistake leading to the result that no one can revoke the locked token

```
        LockedToken lockedToken = new LockedToken(this, _donor, _beneficiary, releaseTime, _revocable);
        BasicToken.transfer(lockedToken, amount);
        Lock(lockedToken, _beneficiary, lockedToken.balanceOf(), releaseTime);
        return lockedToken;
    }

    function burn(uint256 _amount) onlyOwner public {
        require(_amount <= balances[msg.sender]);
        balances[msg.sender] = balances[msg.sender].sub(_amount);
        totalSupply_ = totalSupply_.sub(_amount);
        Burn(msg.sender, _amount, totalSupply_);
        Transfer(msg.sender, address(0), _amount);
    }
}
```

//SlowMist// The tokenVault logic

```
contract LockedToken {
    ERC20Basic public token;
    address public donor;
    address public beneficiary;
    uint256 public releaseTime;
    bool public revocable;
```

```
    event Claim(address beneficiary, uint256 amount, uint256 releaseTime);
    event Revoke(address donor, uint256 amount);

    function LockedToken(ERC20Basic _token, address _donor, address _beneficiary, uint256 _releaseTime, bool
_revocable) public {
        require(_token != address(0));
        require(_donor != address(0));
        require(_beneficiary != address(0));
        require(_releaseTime > now);

        token = ERC20Basic(_token);
        donor = _donor;
        beneficiary = _beneficiary;
        releaseTime = _releaseTime;
        revocable = _revocable;
    }

    function balanceOf() public view returns (uint256) {
        return token.balanceOf(this);
    }

    function revoke() public {
        require(revocable);
        require(msg.sender == donor);

        uint amount = token.balanceOf(this);
        require(amount > 0);
        token.transfer(donor, amount);
        Revoke(donor, amount);
    }

    function claim() public {
        require(now >= releaseTime);

        uint amount = token.balanceOf(this);
        require(amount > 0);
        token.transfer(beneficiary, amount);
        Claim(beneficiary, amount, releaseTime);
    }
}
```

慢雾科技
slow mist

Official Website
www.slowmist.com

E-mail
team@slowmist.com

Twitter
@SlowMist_Team

WeChat Official Account