Smart Contract Security Audit Report

The SlowMist Security Team received the JUST team's application for smart contract security audit of the JST on April 24, 2020. The following are the details and results of this smart contract security audit:

Token name :

JST

The Contract address :

TCFLL5dx5ZJdKnWuesXxi1VPwjLVmWZZy9

Link address :

https://tronscan.org/#/token20/TCFLL5dx5ZJdKnWuesXxi1VPwjLVmWZZy9

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 : 0X002004280002

Audit Date : April 28, 2020

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 on TRON that does not contain the tokenVault section. The total amount of contract tokens can be changed, Auth can burn tokens through the burn function. Auth can mint unlimited tokens through the mint function. According to the feedback from the project party, the authority of `Auth` role will be transferred to the voting system of JUST in the future, and the voting system will decide the number of tokens to be issued. The contract does not have the Overflow and the Race Conditions issue. The comprehensive evaluation contract is no risk.

The source code:

auth.sol:

```
// This program is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
```

```
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.

// You should have received a copy of the GNU General Public License
// along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

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

```
pragma solidity >=0.4.23;

contract DSAuthority {
    function canCall(
        address src, address dst, bytes4 sig
    ) public view returns (bool);
}

contract DSAuthEvents {
    event LogSetAuthority (address indexed authority);
    event LogSetOwner     (address indexed owner);
}

contract DSAuth is DSAuthEvents {
    DSAuthority  public  authority;
    address      public  owner;

    constructor() public {
        owner = msg.sender;
        emit LogSetOwner(msg.sender);
    }

    function setOwner(address owner_)
    public
    auth
    {
        owner = owner_;
        emit LogSetOwner(owner);
    }

    function setAuthority(DSAuthority authority_)
    public
```

```
    auth
    returns (bool result)
    {
        authority = authority_;
        emit LogSetAuthority(address(authority));
        result = true;
    }

    modifier auth {
        require(isAuthorized(msg.sender, msg.sig), "ds-auth-unauthorized");
        _;
    }

    function isAuthorized(address src, bytes4 sig) internal view returns (bool) {
        if (src == address(this)) {
            return true;
        } else if (src == owner) {
            return true;
        } else if (authority == DSAuthority(0)) {
            return false;
        } else {
            return authority.canCall(src, address(this), sig);
        }
    }
}
```

note.sol:

```
/// note.sol -- the `note' modifier, for logging calls as events

// This program is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.

// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.

// You should have received a copy of the GNU General Public License
// along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

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

```
pragma solidity >=0.4.23;

contract DSNote {
    event LogNote(
        bytes4   indexed sig,
        address  indexed guy,
        bytes32  indexed foo,
        bytes32  indexed bar,
        uint256 sad,
        bytes fax
    ) anonymous;

    modifier note {
        bytes32 foo;
        bytes32 bar;
        uint256 sad;

        assembly {
            foo := calldataload(4)
            bar := calldataload(36)
            sad := callvalue
        }

        emit LogNote(msg.sig, msg.sender, foo, bar, sad, msg.data);

        _;
    }
}
```

stop.sol:

```
/// stop.sol -- mixin for enable/disable functionality

// Copyright (C) 2017  DappHub, LLC

// This program is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.

// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
```

```
// GNU General Public License for more details.

// You should have received a copy of the GNU General Public License
// along with this program.    If not, see <http://www.gnu.org/licenses/>.
```

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

```
pragma solidity >=0.4.23;

import "./auth.sol";
import "./note.sol";

contract DSStop is DSNote, DSAuth {
    bool public stopped;

    modifier stoppable {
        require(!stopped, "ds-stop-is-stopped");
        _;
    }
```

//SlowMist// Suspending all transactions upon major abnormalities is a recommended approach

```
    function stop() public auth note {
        stopped = true;
    }
    function start() public auth note {
        stopped = false;
    }

}
```

trc20.sol:

```
/// TRC20.sol -- API for the TRC20 token standard

// See <https://github.com/tronprotocol/tips/blob/master/tip-20.md>.

// This file likely does not meet the threshold of originality
// required for copyright to apply.    As a result, this is free and
// unencumbered software belonging to the public domain.
```

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

```
pragma solidity ^0.4.8;
```

//SlowMist// The contract allows the use of a compiler version of 0.4.8 or

above, and it is recommended to select a newer compiler version for compilation and deployment when deploying the contract

```
contract TRC20Events {
    event Approval(address indexed src, address indexed guy, uint wad);
    event Transfer(address indexed src, address indexed dst, uint wad);
}

contract TRC20 is TRC20Events {
    function totalSupply() public view returns (uint);
    function balanceOf(address guy) public view returns (uint);
    function allowance(address src, address guy) public view returns (uint);

    function approve(address guy, uint wad) public returns (bool);
    function transfer(address dst, uint wad) public returns (bool);
    function transferFrom(
        address src, address dst, uint wad
    ) public returns (bool);
}
```

math.sol:

```
/// math.sol -- mixin for inline numerical wizardry

// This program is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.

// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.

// You should have received a copy of the GNU General Public License
// along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

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

```
pragma solidity ^0.4.25;

contract DSMath {
```

```
/*
standard uint256 functions
 */
function add(uint256 x, uint256 y) internal pure returns (uint256 z) {
    require((z = x + y) >= x, "ds-math-add-overflow");
}

function sub(uint256 x, uint256 y) internal pure returns (uint256 z) {
    require((z = x - y) <= x, "ds-math-sub-underflow");
}

function mul(uint256 x, uint256 y) internal pure returns (uint256 z) {
    require(y == 0 || (z = x * y) / y == x, "ds-math-mul-overflow");
}
```

//SlowMist// It's redundant code

```
function min(uint256 x, uint256 y) internal pure returns (uint256 z) {
    return x <= y ? x : y;
}
```

//SlowMist// It's redundant code

```
function max(uint256 x, uint256 y) internal pure returns (uint256 z) {
    return x >= y ? x : y;
}

/*
uint128 functions (h is for half)
*/
```

//SlowMist// It's redundant code

```
function hadd(uint128 x, uint128 y) pure internal returns (uint128 z) {
    require((z = x + y) >= x);
}
```

//SlowMist// It's redundant code

```
function hsub(uint128 x, uint128 y) pure internal returns (uint128 z) {
    require((z = x - y) <= x);
}
```

//SlowMist// It's redundant code

```
function hmul(uint128 x, uint128 y) pure internal returns (uint128 z) {
    require(y == 0 || (z = x * y) / y == x, "ds-math-mul-overflow");
```

```
}
```

//SlowMist// It's redundant code

```
function hmin(uint128 x, uint128 y) pure internal returns (uint128 z) {
    return x <= y ? x : y;
}
```

//SlowMist// It's redundant code

```
function hmax(uint128 x, uint128 y) pure internal returns (uint128 z) {
    return x >= y ? x : y;
}
/*
 * int256 functions
 **/
```

//SlowMist// It's redundant code

```
function imin(int256 x, int256 y) internal pure returns (int256 z) {
    return x <= y ? x : y;
}
```

//SlowMist// It's redundant code

```
function imax(int256 x, int256 y) internal pure returns (int256 z) {
    return x >= y ? x : y;
}


/*
 * WAD math
 **/
uint256 constant WAD = 10 ** 18;
uint256 constant RAY = 10 ** 27;
```

//SlowMist// It's redundant code

```
function wmul(uint128 x, uint128 y) internal pure returns (uint128 z) {
    z = cast((uint256(x) * y + WAD / 2) / WAD);
}
```

//SlowMist// It's redundant code

```
function whdiv(uint128 x, uint128 y) internal pure returns (uint128 z) {
    z = cast((uint256(x) * WAD + y / 2) / y);
}
```

//SlowMist// It's redundant code

```
function wmul(uint256 x, uint256 y) internal pure returns (uint256 z) {
    z = add(mul(x, y), WAD / 2) / WAD;
}
```

//SlowMist// It's redundant code

```
function wdiv(uint256 x, uint256 y) internal pure returns (uint256 z) {
    z = add(mul(x, WAD), y / 2) / y;
}
```

//SlowMist// It's redundant code

```
function rmul(uint128 x, uint128 y) internal pure returns (uint128 z) {
    z = cast((uint256(x) * y + RAY / 2) / RAY);
}
```

//SlowMist// It's redundant code

```
function rdiv(uint128 x, uint128 y) internal pure returns (uint128 z) {
    z = cast((uint256(x) * RAY + y / 2) / y);
}


function rmul(uint256 x, uint256 y) internal pure returns (uint256 z) {
    z = add(mul(x, y), RAY / 2) / RAY;
}
```

//SlowMist// It's redundant code

```
function rdiv(uint256 x, uint256 y) internal pure returns (uint256 z) {
    z = add(mul(x, RAY), y / 2) / y;
}


// This famous algorithm is called "exponentiation by squaring"
// and calculates x^n with x as fixed-point256 and n as regular unsigned.
//
// It's O(log n), instead of O(n) for naive repeated multiplication.
//
// These facts are why it works:
//
//  If n is even, then x^n = (x^2)^(n/2).
//  If n is odd,  then x^n = x * x^(n-1),
//   and applying the equation for even x gives
//    x^n = x * (x^2)^((n-1) / 2).
//
//  Also, EVM division is flooring and
//    floor[(n-1) / 2] = floor[n / 2].
```

```
    //
    function rpow(uint256 x, uint256 n) internal pure returns (uint256 z) {
        z = n % 2 != 0 ? x : RAY;

        for (n /= 2; n != 0; n /= 2) {
            x = rmul(x, x);

            if (n % 2 != 0) {
                z = rmul(z, x);
            }
        }
    }

    function cast(uint256 x) internal pure returns (uint128 z) {
        require((z = uint128(x)) == x);
    }
}
```

token_base.sol:

```
/// base.sol -- basic ERC20 implementation

// Copyright (C) 2015, 2016, 2017  DappHub, LLC

// This program is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.

// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.

// You should have received a copy of the GNU General Public License
// along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

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

```
pragma solidity >=0.4.23;

import "./trc20.sol";
import "./math.sol";
```

```
contract DSTokenBase is TRC20, DSMath {
    uint256                                            _supply;
    mapping(address => uint256)                        _balances;
    mapping(address => mapping(address => uint256))    _approvals;

    constructor(uint supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }

    function totalSupply() public view returns (uint) {
        return _supply;
    }

    function balanceOf(address src) public view returns (uint) {
        return _balances[src];
    }

    function allowance(address src, address guy) public view returns (uint) {
        return _approvals[src][guy];
    }

    function transfer(address dst, uint wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }

    function transferFrom(address src, address dst, uint wad)
    public
    returns (bool)
    {
        if (src != msg.sender) {
            require(_approvals[src][msg.sender] >= wad, "ds-token-insufficient-approval");
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        require(_balances[src] >= wad, "ds-token-insufficient-balance");
        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);


        return true; //SlowMist// The return value conforms to the TIP20 specification
```

```
    }

    function approve(address guy, uint wad) public returns (bool) {
        _approvals[msg.sender][guy] = wad;

        emit Approval(msg.sender, guy, wad);

        return true; //SlowMist// The return value conforms to the TIP20 specification
    }
}
```

token.sol:

```
/// token.sol -- ERC20 implementation with minting and burning

// Copyright (C) 2015, 2016, 2017  DappHub, LLC

// This program is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.

// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.

// You should have received a copy of the GNU General Public License
// along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

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

```
pragma solidity >=0.4.23;

import "./stop.sol";

import "./token_base.sol";

contract DSToken is DSTokenBase(0), DSStop {

    string  public  symbol;
    uint256  public  decimals = 18; // standard token precision. override to customize
```

```
constructor(string symbol_) public {
    symbol = symbol_;
}

event Mint(address indexed guy, uint wad);
event Burn(address indexed guy, uint wad);

function approve(address guy) public stoppable returns (bool) {
    return super.approve(guy, uint(-1));
}

function approve(address guy, uint wad) public stoppable returns (bool) {
    return super.approve(guy, wad);
}

function transferFrom(address src, address dst, uint wad)
public
stoppable
returns (bool)
{
    if (src != msg.sender && _approvals[src][msg.sender] != uint(-1)) {
        require(_approvals[src][msg.sender] >= wad, "ds-token-insufficient-approval");
        _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
    }

    require(_balances[src] >= wad, "ds-token-insufficient-balance");
    _balances[src] = sub(_balances[src], wad);
    _balances[dst] = add(_balances[dst], wad);

    emit Transfer(src, dst, wad);


    return true; //SlowMist// The return value conforms to the TIP20 specification
}

function push(address dst, uint wad) public {
    transferFrom(msg.sender, dst, wad);
}
function pull(address src, uint wad) public {
    transferFrom(src, msg.sender, wad);
}
function move(address src, address dst, uint wad) public {
```

```
        transferFrom(src, dst, wad);
    }

    function mint(uint wad) public {
        mint(msg.sender, wad);
    }
    function burn(uint wad) public {
        burn(msg.sender, wad);
    }
    function mint(address guy, uint wad) public auth stoppable {
        _balances[guy] = add(_balances[guy], wad);
        _supply = add(_supply, wad);
        emit Mint(guy, wad);
    }
    function burn(address guy, uint wad) public auth stoppable {
        if (guy != msg.sender && _approvals[guy][msg.sender] != uint(-1)) {
            require(_approvals[guy][msg.sender] >= wad, "ds-token-insufficient-approval");
            _approvals[guy][msg.sender] = sub(_approvals[guy][msg.sender], wad);
        }

        require(_balances[guy] >= wad, "ds-token-insufficient-balance");
        _balances[guy] = sub(_balances[guy], wad);
        _supply = sub(_supply, wad);
        emit Burn(guy, wad);
    }

    // Optional token name
    string    public   name = "";

    function setName(string name_) public auth {
        name = name_;
    }
    // Optional symbol name
    function setSymbol(string symbol_) public auth {
        symbol = symbol_;
    }
}
```

慢雾科技
slow mist

Official Website
www.slowmist.com

E-mail
team@slowmist.com

Twitter
@SlowMist_Team

WeChat Official Account