Smart Contract Security Audit Report

The SlowMist Security Team received the STON team's application for smart contract security audit of the STON Token on January 25, 2019. The following are the details and results of this smart contract security audit:

Token name :

STON

The Contract address :

0x1571c3815bd411c39daed0e61b8eecb37c441486

Link address :

https://etherscan.io/address/0x1571c3815bd411c39daed0e61b8eecb37c441486

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	Uninitialized Storage Pointers Audit	-	Passed
11	Arithmetic Accuracy Deviation Audit	-	Passed

Audit Result : **Passed**

Audit Number : 0X001901280001

Audit Date :January 28, 2019

Audit Team : SlowMist Security Team

Summary: This is a token contract that does not contain the tokenVault section. The contract does not have the Overflow, the Race Conditions issue. The comprehensive evaluation contract is no risk.

The source code:

```
pragma solidity ^0.5.0;
```

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

```
// ----------------------------------------------------------------------------
// Symbol      : STON
// Name        : STONetwork
// Total supply: 100,000,0000.000000000000000000
// Decimals    : 18
// Copyright (c) 2018 <STONetwork>. The MIT Licence.
// ----------------------------------------------------------------------------
```

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

```
// ----------------------------------------------------------------------------
// Safe maths
// ----------------------------------------------------------------------------
```

```
library SafeMath {
    function add(uint a, uint b) internal pure returns (uint c) {
        c = a + b;
        require(c >= a);
    }
    function sub(uint a, uint b) internal pure returns (uint c) {
        require(b <= a);
        c = a - b;
    }
    function mul(uint a, uint b) internal pure returns (uint c) {
        c = a * b;
        require(a == 0 || c / a == b);
    }
    function div(uint a, uint b) internal pure returns (uint c) {
        require(b > 0);
        c = a / b;
    }
}



// ----------------------------------------------------------------------------
// ERC Token Standard #20 Interface
// https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
// ----------------------------------------------------------------------------
contract ERC20Interface {
    function totalSupply() public view returns (uint);
    function balanceOf(address tokenOwner) public view returns (uint balance);
    function allowance(address tokenOwner, address spender) public view returns (uint remaining);
    function transfer(address to, uint _value) public returns (bool success);
    function approve(address spender, uint _value) public returns (bool success);
    function transferFrom(address _from, address _to, uint _value) public returns (bool success);

    event Transfer(address indexed _from, address indexed _to, uint _value);
    event Approval(address indexed tokenOwner, address indexed spender, uint _value);
}



// ----------------------------------------------------------------------------
// Contract function to receive approval and execute function in one call
//
// Borrowed from MiniMeToken
// ----------------------------------------------------------------------------
```

```
contract ApproveAndCallFallBack {
    function receiveApproval(address _from, uint256 _value, address token, bytes memory data) public;
}


// ----------------------------------------------------------------------------
// Owned contract
// ----------------------------------------------------------------------------
contract Owned {
    address public owner;
    address public newOwner;

    event OwnershipTransferred(address indexed _from, address indexed _to);

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

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

    function transferOwnership(address _newOwner) public onlyOwner {
        newOwner = _newOwner;
    }
    function acceptOwnership() public {
        require(msg.sender == newOwner);
        emit OwnershipTransferred(owner, newOwner);
        owner = newOwner;
        newOwner = address(0);
    }
}


// ----------------------------------------------------------------------------
// ERC20 Token, with the addition of symbol, name and decimals and a
// fixed supply
// ----------------------------------------------------------------------------
contract STONetwork is ERC20Interface, Owned {
    using SafeMath for uint;
```

```
string public symbol;
string public  name;
uint8 public decimals;
uint _initialTokenNumber;
uint _totalSupply;

mapping(address => uint) balances;
mapping(address => mapping(address => uint)) allowed;

uint exchangerToken;
uint reservedToken;
uint developedToken;

address public constant developed1Address     = 0xcFCb491953Da1d10D037165dFa1298D00773fcA7;
address public constant developed2Address     = 0xA123BceDB9d2E4b09c8962C62924f091380E1Ad7;
address public constant developed3Address     = 0x51aeD4EDC28aad15C353D958c5A813aa21F351b6;
address public constant exchangedAddress     = 0x2630e8620d53C7f64f82DAEA50257E83297eE009;

// ------------------------------------------------------------------------
// Constructor
// ------------------------------------------------------------------------
constructor() public {
    symbol = "STON";
    name = "STONetwork";
    decimals = 18;
    _initialTokenNumber = 1000000000;
    _totalSupply = _initialTokenNumber * 10 ** uint(decimals);

    reservedToken = _totalSupply * 40 / 100;  // 40%

    developedToken = _totalSupply * 10 / 100; //30% 3 Address

    exchangerToken = _totalSupply * 30 / 100; // 30%

    balances[owner] = reservedToken;
    emit Transfer(address(0), owner, reservedToken);

    balances[exchangedAddress] = exchangerToken;
    emit Transfer(address(0), exchangedAddress, exchangerToken);

    balances[developed1Address] = developedToken;
    emit Transfer(address(0), developed1Address, developedToken);
```

```
        balances[developed2Address] = developedToken;
        emit Transfer(address(0), developed2Address, developedToken);
        balances[developed3Address] = developedToken;
        emit Transfer(address(0), developed3Address, developedToken);

    }


    // ------------------------------------------------------------------------
    // Total supply
    // ------------------------------------------------------------------------
    function totalSupply() public view returns (uint) {
        return _totalSupply;
    }


    // ------------------------------------------------------------------------
    // Get the token balance for account `tokenOwner`
    // ------------------------------------------------------------------------
    function balanceOf(address _owner) public view returns (uint balance) {
        return balances[_owner];
    }


    // ------------------------------------------------------------------------
    // Transfer the balance from token owner's account to `to` account
    // - Owner's account must have sufficient balance to transfer
    // - 0 value transfers are allowed
    // ------------------------------------------------------------------------
    function transfer(address _to, uint _value) public returns (bool success) {
        require(balances[msg.sender] >= _value);
        require(_to != address(0)); //SlowMist// This kind of check is very good, avoiding user mistake leading to the loss of token during transfer
        balances[msg.sender] = balances[msg.sender].sub(_value);
        balances[_to] = balances[_to].add(_value);
        emit Transfer(msg.sender, _to, _value);
        return true;  //SlowMist// The return value conforms to the EIP20 specification
    }
```

```
    // ------------------------------------------------------------------------
    // Token owner can approve for `spender` to transferFrom(...) `tokens`
    // from the token owner's account
    //
    // https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md
    // recommends that there are no checks for the approval double-spend attack
    // as this should be implemented in user interfaces
    // ------------------------------------------------------------------------
    function approve(address _spender, uint _value) public returns (bool success) {
        require(_spender != address(0)); //SlowMist// This kind of check is very good, avoiding user mistake leading to waste Gas
        allowed[msg.sender][_spender] = _value;
        emit Approval(msg.sender, _spender, _value);
        return true;  //SlowMist// The return value conforms to the EIP20 specification
    }


    // ------------------------------------------------------------------------
    // Transfer `tokens` from the `from` account to the `to` account
    //
    // The calling account must already have sufficient tokens approve(...)-d
    // for spending from the `from` account and
    // - From account must have sufficient balance to transfer
    // - Spender must have sufficient allowance to transfer
    // - 0 value transfers are allowed
    // ------------------------------------------------------------------------
    function transferFrom(address _from, address _to, uint _value) public returns (bool success) {
        require(_to != address(0)); //SlowMist// This kind of check is very good, avoiding user mistake leading to the loss of token during transfer
        require(balances[_from] >= _value);
        require(allowed[_from][msg.sender] >= _value);

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

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


    // ------------------------------------------------------------------------
    // Returns the amount of tokens approved by the owner that can be
    // transferred to the spender's account
    // ------------------------------------------------------------------------
    function allowance(address _owner, address _spender) public view returns (uint remaining) {
        return allowed[_owner][_spender];
    }


    // ------------------------------------------------------------------------
    // Token owner can approve for `spender` to transferFrom(...) `tokens`
    // from the token owner's account. The `spender` contract function
    // `receiveApproval(...)` is then executed
    // ------------------------------------------------------------------------
    function approveAndCall(address _spender, uint _value, bytes memory data) public returns (bool success)
{
        allowed[msg.sender][_spender] = _value;
        emit Approval(msg.sender, _spender, _value);
        ApproveAndCallFallBack(_spender).receiveApproval(msg.sender, _value, address(this), data);
        return true;
    }


    // ------------------------------------------------------------------------
    // Don't accept ETH
    // ------------------------------------------------------------------------
    function () external payable {
        revert();
    }


    // ------------------------------------------------------------------------
    // Owner can transfer out any accidentally sent ERC20 tokens
    // ------------------------------------------------------------------------
    function transferAnyERC20Token(address _tokenAddress, uint _value) public onlyOwner returns (bool
success) {
        return ERC20Interface(_tokenAddress).transfer(owner, _value);
```

```
    }
}
```

慢雾科技
slow mist

Official Website
www.slowmist.com

E-mail
team@slowmist.com

Twitter
@SlowMist_Team

WeChat Official Account