Smart Contract Security Audit Report

The SlowMist Security Team received the ASSEMBLE team's application for smart contract security audit of the ASM on May 08, 2020. The following are the details and results of this smart contract security audit:

Token name :

ASM

File name and HASH(SHA256) :

AssembleToken.sol:

d18f20ce9771253c359acad247fd32608b268503705fa71a9c6a0d61697b455d

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 : 0X002005120001

Audit Date : May 12, 2020

Audit Team : SlowMist Security Team

Summary: This is a token contract that contains the tokenVault section. The total amount of contract tokens can be changed, owner can burn his own tokens through the burnToken function. SafeMath security module is used, which is a commendable approach. The contract does not have the Overflow and the Race Conditions issue.

During the audit, we found that the owner authority is too large:

1. The tokens distribution for each role in the contract is controlled by the owner.

2. The owner can transfer the tokens in the contract to the owner at any time through the withdrawTokens function.

3. The owner can destroy the contract at any time through the close function.

After feeding back with the project side, they decided to change the owner to the MultiSig contract address through the transferOwnership function to solve the issue of owner authority

too large.

The source code:

```
//SlowMist// The contract does not have the Overflow and the Race Conditions issue
pragma solidity ^0.5.9;
//SlowMist// SafeMath security Module is used, which is a recommend approach
library SafeMath
{
    function mul(uint256 a, uint256 b) internal pure returns (uint256)
    {
        uint256 c = a * b;
        assert(a == 0 || 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)
    {
        uint256 c = a / b;
        return c;
    }
    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)
    {
        uint256 c = a + b;
```

```
                assert(c >= a); //SlowMist// It is recommended to replace "assert" with "require" to optimize Gas

                return c;
        }
}

contract OwnerHelper
{
        address public owner;

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

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

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

        function transferOwnership(address _to) onlyOwner public
        {
        require(_to != owner);

        require(_to != address(0x0)); //SlowMist// This check is quite good in avoiding losing control of the contract caused by user mistakes

    address from = owner;
        owner = _to;

        emit ChangeOwner(from, _to);
        }
}

contract ERC20Interface
```

```
{
    event Transfer( address indexed _from, address indexed _to, uint _value);
    event Approval( address indexed _owner, address indexed _spender, uint _value);

    function totalSupply() view public returns (uint _supply);
    function balanceOf( address _who ) public view returns (uint _value);
    function transfer( address _to, uint _value) public returns (bool _success);
    function approve( address _spender, uint _value ) public returns (bool _success);
    function allowance( address _owner, address _spender ) public view returns (uint _allowance);
    function transferFrom( address _from, address _to, uint _value) public returns (bool _success);
}

contract AssembleToken is ERC20Interface, OwnerHelper
{
    using SafeMath for uint;

    string public name;
    uint public decimals;
    string public symbol;

    uint constant private E18 = 1000000000000000000;
    uint constant private month = 2592000;

    // Total                                1,500,000,000
    uint constant public maxTotalSupply =     1500000000 * E18;

    // Sale Supply                          300,000,000 (20%)
    uint constant public maxSaleSupply =      300000000 * E18;

    // Strategic Partners                   270,000,000 (18%)
    uint constant public maxSPSupply =        270000000 * E18;

    // EcoSystem                            240,000,000 (16%)
    uint constant public maxEcoSupply =       240000000 * E18;

    // Marketing                            210,000,000 (14%)
    uint constant public maxMktSupply =       210000000 * E18;

    // Development                          180,000,000 (12%)
    uint constant public maxDevSupply =       180000000 * E18;

    // Reserve                              150,000,000 (10%)
```

```
uint constant public maxReserveSupply =          150000000 * E18;

// Team                                          75,000,000 (5%)
uint constant public maxTeamSupply =              75000000 * E18;

// Advisor                                       75,000,000 (5%)
uint constant public maxAdvisorSupply =           75000000 * E18;

uint constant public seedSaleSupply =             40000000 * E18;
uint constant public privateSaleSupply =         250000000 * E18;
uint constant public publicSaleSupply =           10000000 * E18;

// Lock
uint constant public teamVestingSupply =           3125000 * E18;
uint constant public teamVestingLockDate =   6 * month;
uint constant public teamVestingTime = 24;

uint constant public advisorVestingSupply =       18750000 * E18;
uint constant public advisorVestingLockDate = 12 * month;
uint constant public advisorVestingTime = 4;

uint public totalTokenSupply;
uint public tokenIssuedSale;
uint public tokenIssuedSP;
uint public tokenIssuedEco;
uint public tokenIssuedMkt;
uint public tokenIssuedDev;
uint public tokenIssuedRsv;
uint public tokenIssuedTeam;
uint public tokenIssuedAdv;

uint public burnTokenSupply;

mapping (address => uint) public balances;
mapping (address => mapping ( address => uint )) public approvals;

mapping (uint => uint) public tmVestingTimer;
mapping (uint => uint) public tmVestingBalances;

mapping (uint => uint) public advVestingTimer;
mapping (uint => uint) public advVestingBalances;
```

```
bool public tokenLock = true;
bool public saleTime = true;
uint public endSaleTime = 0;

event SaleIssue(address indexed _to, uint _tokens);
event SPIssue(address indexed _to, uint _tokens);
event EcoIssue(address indexed _to, uint _tokens);
event MktIssue(address indexed _to, uint _tokens);
event DevIssue(address indexed _to, uint _tokens);
event RsvIssue(address indexed _to, uint _tokens);
event TeamIssue(address indexed _to, uint _tokens);
event AdvIssue(address indexed _to, uint _tokens);

event Burn(address indexed _from, uint _tokens);

event TokenUnlock(address indexed _to, uint _tokens);
event EndSale(uint _date);

constructor() public
{
    name            = "ASSEMBLE";
    decimals        = 18;
    symbol          = "ASM";

    totalTokenSupply      = 0;

    tokenIssuedSale       = 0;
    tokenIssuedSP         = 0;
    tokenIssuedEco        = 0;
    tokenIssuedMkt        = 0;
    tokenIssuedDev        = 0;
    tokenIssuedRsv        = 0;
    tokenIssuedTeam       = 0;
    tokenIssuedAdv        = 0;


    burnTokenSupply       = 0;

    require(maxTeamSupply == teamVestingSupply.mul(teamVestingTime));
    require(maxAdvisorSupply == advisorVestingSupply.mul(advisorVestingTime));

    require(maxSaleSupply == seedSaleSupply + privateSaleSupply + publicSaleSupply);
```

```
        require(maxTotalSupply == maxSaleSupply + maxSPSupply +   maxEcoSupply + maxMktSupply + maxDevSupply +
maxReserveSupply + maxTeamSupply + maxAdvisorSupply);
    }

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

    function balanceOf(address _who) view public returns (uint)
    {
        return balances[_who];
    }

    function transfer(address _to, uint _value) public returns (bool)
    {
        require(isTransferable() == true);
        require(balances[msg.sender] >= _value);

        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
    }

    function approve(address _spender, uint _value) public returns (bool)
    {
        require(isTransferable() == true);
        require(balances[msg.sender] >= _value);

        approvals[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) view public returns (uint)
```

```
{
    return approvals[_owner][_spender];
}

function transferFrom(address _from, address _to, uint _value) public returns (bool)
{
    require(isTransferable() == true);
    require(balances[_from] >= _value);
    require(approvals[_from][msg.sender] >= _value);

    approvals[_from][msg.sender] = approvals[_from][msg.sender].sub(_value);
    balances[_from] = balances[_from].sub(_value);
    balances[_to]   = balances[_to].add(_value);

    emit Transfer(_from, _to, _value);


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

function spIssue(address _to) onlyOwner public
{
    require(saleTime == false);
    require(tokenIssuedSP == 0);

    uint tokens = maxSPSupply;

    balances[_to] = balances[_to].add(tokens);

    totalTokenSupply = totalTokenSupply.add(tokens);
    tokenIssuedSP = tokenIssuedSP.add(tokens);

    emit SPIssue(_to, tokens);
}

function ecoIssue(address _to) onlyOwner public
{
    require(saleTime == false);
    require(tokenIssuedEco == 0);

    uint tokens = maxEcoSupply;
```

```
        balances[_to] = balances[_to].add(tokens);

        totalTokenSupply = totalTokenSupply.add(tokens);
        tokenIssuedEco = tokenIssuedEco.add(tokens);

        emit EcoIssue(_to, tokens);
    }

    function mktIssue(address _to) onlyOwner public
    {
        require(saleTime == false);
        require(tokenIssuedMkt == 0);

        uint tokens = maxMktSupply;

        balances[_to] = balances[_to].add(tokens);

        totalTokenSupply = totalTokenSupply.add(tokens);
        tokenIssuedMkt = tokenIssuedMkt.add(tokens);

        emit MktIssue(_to, tokens);
    }

    function devIssue(address _to) onlyOwner public
    {
        require(saleTime == false);
        require(tokenIssuedDev == 0);

        uint tokens = maxDevSupply;

        balances[_to] = balances[_to].add(tokens);

        totalTokenSupply = totalTokenSupply.add(tokens);
        tokenIssuedDev = tokenIssuedDev.add(tokens);

        emit DevIssue(_to, tokens);
    }

    function rsvIssue(address _to) onlyOwner public
    {
        require(saleTime == false);
        require(tokenIssuedRsv == 0);
```

```
        uint tokens = maxReserveSupply;

        balances[_to] = balances[_to].add(tokens);

        totalTokenSupply = totalTokenSupply.add(tokens);
        tokenIssuedRsv = tokenIssuedRsv.add(tokens);

        emit RsvIssue(_to, tokens);
    }

    // _time : 0 ~ 24
    function teamIssue(address _to, uint _time) onlyOwner public
    {
        require(saleTime == false);
        require( _time < teamVestingTime);

        uint nowTime = now;
        require( nowTime > tmVestingTimer[_time] );

        uint tokens = teamVestingSupply;

        require(tokens == tmVestingBalances[_time]);
        require(maxTeamSupply >= tokenIssuedTeam.add(tokens));

        balances[_to] = balances[_to].add(tokens);
        tmVestingBalances[_time] = 0;

        totalTokenSupply = totalTokenSupply.add(tokens);
        tokenIssuedTeam = tokenIssuedTeam.add(tokens);

        emit TeamIssue(_to, tokens);
    }

    // _time : 0 ~ 4
    function advisorIssue(address _to, uint _time) onlyOwner public
    {
        require(saleTime == false);
        require( _time < advisorVestingTime);

        uint nowTime = now;
        require( nowTime > advVestingTimer[_time] );
```

```
        uint tokens = advisorVestingSupply;

        require(tokens == advVestingBalances[_time]);
        require(maxAdvisorSupply >= tokenIssuedAdv.add(tokens));

        balances[_to] = balances[_to].add(tokens);
        advVestingBalances[_time] = 0;

        totalTokenSupply = totalTokenSupply.add(tokens);
        tokenIssuedAdv = tokenIssuedAdv.add(tokens);

        emit AdvIssue(_to, tokens);
    }

    function seedSaleIssue(address _to) onlyOwner public
    {
        require(tokenIssuedSale == 0);

        uint tokens = seedSaleSupply;

        balances[_to] = balances[_to].add(tokens);

        totalTokenSupply = totalTokenSupply.add(tokens);
        tokenIssuedSale = tokenIssuedSale.add(tokens);

        emit SaleIssue(_to, tokens);
    }

    function privateSaleIssue(address _to) onlyOwner public
    {
        require(tokenIssuedSale == seedSaleSupply);

        uint tokens = privateSaleSupply;

        balances[_to] = balances[_to].add(tokens);

        totalTokenSupply = totalTokenSupply.add(tokens);
        tokenIssuedSale = tokenIssuedSale.add(tokens);

        emit SaleIssue(_to, tokens);
    }
```

```
    function publicSaleIssue(address _to) onlyOwner public
    {
        require(tokenIssuedSale == seedSaleSupply + privateSaleSupply);

        uint tokens = publicSaleSupply;

        balances[_to] = balances[_to].add(tokens);

        totalTokenSupply = totalTokenSupply.add(tokens);
        tokenIssuedSale = tokenIssuedSale.add(tokens);

        emit SaleIssue(_to, tokens);
    }

    function isTransferable() private view returns (bool)
    {
        if(tokenLock == false)
        {
            return true;
        }
        else if(msg.sender == owner) //SlowMist// The Owner can transfer tokens without restrictions
        {
            return true;
        }

        return false;
    }

    function setTokenUnlock() onlyOwner public
    {
        require(tokenLock == true);
        require(saleTime == false);

        tokenLock = false;
    }
    //SlowMist// Suspending all transactions upon major abnormalities is a recommended approach
    function setTokenLock() onlyOwner public
```

```
{
    require(tokenLock == false);

    tokenLock = true;
}

function endSale() onlyOwner public
{
    require(saleTime == true);
    require(maxSaleSupply == tokenIssuedSale);

    saleTime = false;

    uint nowTime = now;
    endSaleTime = nowTime;

    for(uint i = 0; i < teamVestingTime; i++)
    {
        tmVestingTimer[i] = endSaleTime + teamVestingLockDate + ((i+1) * month);
        tmVestingBalances[i] = teamVestingSupply;
    }

    for(uint i = 0; i < advisorVestingTime; i++)
    {
        advVestingTimer[i] = endSaleTime + advisorVestingLockDate + (3 * (i+1) * month);
        advVestingBalances[i] = advisorVestingSupply;
    }

    emit EndSale(endSaleTime);
}

function withdrawTokens(address _contract, uint _decimals, uint _value) onlyOwner public
{

    if(_contract == address(0x0))
    {
        uint eth = _value.mul(10 ** _decimals);
        msg.sender.transfer(eth);
    }
    else
    {
        uint tokens = _value.mul(10 ** _decimals);
```

```
            ERC20Interface(_contract).transfer(msg.sender, tokens);

            emit Transfer(address(0x0), msg.sender, tokens);
        }
    }

    function burnToken(uint _value) onlyOwner public
    {
        uint tokens = _value * E18;

        require(balances[msg.sender] >= tokens);

        balances[msg.sender] = balances[msg.sender].sub(tokens);

        burnTokenSupply = burnTokenSupply.add(tokens);
        totalTokenSupply = totalTokenSupply.sub(tokens);

        emit Burn(msg.sender, tokens);
    }

    function close() onlyOwner public
    {
        selfdestruct(msg.sender);
    }

}
```

慢雾科技
slow mist

Official Website
www.slowmist.com

E-mail
team@slowmist.com

Twitter
@SlowMist_Team

WeChat Official Account