Smart Contract Security Audit Report

The SlowMist Security Team received the MOTIV Protocol team's application for smart contract security audit of the MOV on May 26, 2020. The following are the details and results of this smart contract security audit:

Token name :

MOV

The Contract address :

0x40284109c3309A7C3439111bFD93BF5E0fBB706c

Link address :

https://etherscan.io/address/0x40284109c3309A7C3439111bFD93BF5E0fBB706c

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 : 0X002006010002

Audit Date : June 01, 2020

Audit Team : SlowMist Security Team

Summary: This is a token contract that does not contain the tokenVault section. The total amount of contract tokens can be changed, manager 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. The tokens distribution for each role in the contract is controlled by the owner. The owner can destroy the contract at any time through the close function.

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;
    address public manager;

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

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

    modifier onlyManager
    {
        require(msg.sender == manager);
        _;
    }

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

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

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

```
            require(_to != manager);
            require(_to != address(0x0));

            address from = manager;
            manager = _to;

            emit ChangeManager(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 MOTIVProtocol is ERC20Interface, OwnerHelper
{
    using SafeMath for uint;

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

    uint constant private E18 = 1000000000000000000;

    // Total                                    500,000,000
    uint constant public maxTotalSupply     = 500000000 * E18;

    // Sale Supply                              100,000,000 (20%)
    uint constant public maxSaleSupply      = 100000000 * E18;

    // Marketing                                90,000,000 (18%)
    uint constant public maxMktSupply       = 90000000 * E18;
```

```
// EcoSystem                                90,000,000 (18%)
uint constant public maxEcoSupply         = 90000000 * E18;

// Business Development                     70,000,000 (14%)
uint constant public maxDevSupply         = 70000000 * E18;

// Reserve                                  50,000,000 (10%)
uint constant public maxReserveSupply     = 50000000 * E18;

// Team & Founders                          50,000,000 (10%)
uint constant public maxTeamSupply        = 50000000 * E18;

// Advisors / Early Supporters              25,000,000 (5%)
uint constant public maxAdvisorSupply     = 25000000 * E18;

// Legal & Compliance                       25,000,000 (5%)
uint constant public maxLegalSupply       = 25000000 * E18;

// privateSale                              95,000,000
uint constant public privateSaleSupply    = 95000000 * E18;
// publicSale                               5,000,000
uint constant public publicSaleSupply     = 5000000 * E18;


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

uint public burnTokenSupply;

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

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

```
    event SaleIssue(address indexed _to, uint _tokens);
    event MktIssue(address indexed _to, uint _tokens);
    event EcoIssue(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 LegalIssue(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            = "MOTIV Protocol";
        decimals        = 18;
        symbol          = "MOV";

        totalTokenSupply = 500000000 * E18;
        balances[owner] = totalTokenSupply;

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


        burnTokenSupply       = 0;

        require(maxTotalSupply == maxSaleSupply + maxMktSupply + maxEcoSupply + maxDevSupply +
maxReserveSupply + maxTeamSupply + maxAdvisorSupply + maxLegalSupply);
        require(maxSaleSupply == privateSaleSupply + publicSaleSupply);
    }

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

    uint tokens = maxMktSupply;

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

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

    tokenIssuedMkt = tokenIssuedMkt.add(tokens);

    emit MktIssue(_to, tokens);
}

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

    uint tokens = maxEcoSupply;

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

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

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

        emit EcoIssue(_to, tokens);
    }



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

        uint tokens = maxDevSupply;

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

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

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

        tokenIssuedRsv = tokenIssuedRsv.add(tokens);

        emit RsvIssue(_to, tokens);
    }

    function teamIssue(address _to) onlyOwner public
    {
        require(saleTime == false);
```

```
        require(tokenIssuedTeam == 0);

        uint tokens = maxTeamSupply;

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

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

        tokenIssuedTeam = tokenIssuedTeam.add(tokens);

        emit TeamIssue(_to, tokens);
    }

    function advisorIssue(address _to) onlyOwner public
    {
        require(saleTime == false);
        require(tokenIssuedAdv == 0);

        uint tokens = maxAdvisorSupply;

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

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

        tokenIssuedAdv = tokenIssuedAdv.add(tokens);

        emit AdvIssue(_to, tokens);
    }

    function legalIssue(address _to) onlyOwner public
    {
        require(saleTime == false);
        require(tokenIssuedLegal == 0);

        uint tokens = maxLegalSupply;

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

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

        tokenIssuedLegal = tokenIssuedLegal.add(tokens);
```

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

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

        uint tokens = privateSaleSupply;

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

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

        tokenIssuedSale = tokenIssuedSale.add(tokens);

        emit SaleIssue(_to, tokens);
    }

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

        uint tokens = publicSaleSupply;

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

        balances[_to] = balances[_to].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() onlyManager public
    {
        require(tokenLock == true);
        require(saleTime == false);

        tokenLock = false;
    }

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

    function setTokenLock() onlyManager public
    {
        require(tokenLock == false);

        tokenLock = true;
    }

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

        saleTime = false;

        uint nowTime = now;
        endSaleTime = nowTime;

        emit EndSale(endSaleTime);
    }

    function transferAnyERC20Token(address tokenAddress, uint tokens) onlyOwner public returns (bool success)
    {
        return ERC20Interface(tokenAddress).transfer(manager, tokens);
    }

    function burnToken(uint _value) onlyManager 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