



Smart Contract Security Audit Report



The SlowMist Security Team received the DVI team's application for smart contract security audit of the DVI on June 19, 2020. The following are the details and results of this smart contract security audit:

Token name :

DVI

The Contract address :

0x10633216e7e8281e33c86f02bf8e565a635d9770

Link address :

<https://etherscan.io/address/0x10633216e7e8281e33c86f02bf8e565a635d9770>

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 : 0X002006250001

Audit Date : Jun 25, 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 that contains the tokenVault section. The total amount of contract tokens can be changed, managers can burn their 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. It is suggested to set the owner to MultiSig contract to reduce the risk of being attacked.

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 Division 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 1,000,000,000
    uint constant public maxTotalSupply = 1000000000 * E18;
    // Sale 200,000,000 (20%)
    uint constant public maxSaleSupply = 200000000 * E18;
    // Development 250,000,000 (25%)
    uint constant public maxDevSupply = 250000000 * E18;
    // EcoSystem 170,000,000 (17%)
```

```

uint constant public maxEcoSupply      = 170000000 * E18;
// Marketing                          150,000,000 (15%)

uint constant public maxMktSupply      = 150000000 * E18;
// Reserve                           100,000,000 (10%)

uint constant public maxReserveSupply  = 100000000 * E18;
// Team                              100,000,000 (10%)

uint constant public maxTeamSupply     = 100000000 * E18;
// Advisors                           30,000,000 (3%)

uint constant public maxAdvisorSupply  = 30000000 * E18;

// Lock

uint constant public teamVestingSupply = 10000000 * E18;
uint constant public teamVestingLockDate = 12 * month;
uint constant public teamVestingTime = 10;

uint constant public advisorVestingSupply = 7500000 * E18;
uint constant public advisorVestingLockDate = 3 * month;
uint constant public advisorVestingTime = 4;

uint public totalTokenSupply;
uint public tokenIssuedSale;
uint public tokenIssuedDev;
uint public tokenIssuedEco;
uint public tokenIssuedMkt;
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 DevIssue(address indexed _to, uint _tokens);
event EcoIssue(address indexed _to, uint _tokens);
event MktIssue(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          = "Dvision";
    decimals       = 18;
    symbol         = "DVI";

    totalTokenSupply = 1000000000 * E18;
    balances[owner] = totalTokenSupply;

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

    burnTokenSupply      = 0;

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

    require(maxTotalSupply == maxSaleSupply + maxDevSupply + maxEcoSupply + maxMktSupply +
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 saleIssue(address _to) onlyOwner public
{
    require(tokenIssuedSale == 0);
    uint tokens = maxSaleSupply;

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

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

    tokenIssuedSale = tokenIssuedSale.add(tokens);

    emit SaleIssue(_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 ecolssue(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 Ecolssue(_to, tokens);
}

function mktlssue(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 Mktlssue(_to, tokens);
}

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

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

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

    emit TeamIssue(_to, tokens);
}

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

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

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

emit AdvIssue(_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() 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 * month);
        tmVestingBalances[i] = teamVestingSupply;
    }

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

    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);
    }

    //SlowMist// The owner can destroy the contract at any time through the close function

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



Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

[@SlowMist_Team](https://twitter.com/SlowMist_Team)

WeChat Official Account

