Smart Contract
Security Audit Report

The SlowMist Security Team received the team's application for smart contract security audit of the Klaymeta Token on 2022.02.23. The following are the details and results of this smart contract security audit:

Token Name :

Klaymeta Token

The contract address :

https://scope.klaytn.com/account/0xe815a060b9279eba642f8c889fab7afc0d0aca63?tabId=contractCode

The audit items and results :

(Other unknown security vulnerabilities are not included in the audit responsibility scope)

NO.	Audit Items	Result
1	Replay Vulnerability	Passed
2	Denial of Service Vulnerability	Passed
3	Race Conditions Vulnerability	Passed
4	Authority Control Vulnerability	Passed
5	Integer Overflow and Underflow Vulnerability	Passed
6	Gas Optimization Audit	Passed
7	Design Logic Audit	Passed
8	Uninitialized Storage Pointers Vulnerability	Passed
9	Arithmetic Accuracy Deviation Vulnerability	Passed
10	"False top-up" Vulnerability	Passed
11	Malicious Event Log Audit	Passed
12	Scoping and Declarations Audit	Passed

NO.	Audit Items	Result
13	Safety Design Audit	Passed

Audit Result : Passed

Audit Number : 0X002202250002

Audit Date : 2022.02.23 - 2022.02.25

Audit Team : SlowMist Security Team

Summary conclusion : This is a token contract that does not contain the tokenVault section. The total amount of contract tokens can be changed, users can burn their own tokens through the burn function. SafeMath security module is used, which is a recommended approach. The contract does not have the Overflow and the Race Conditions issue.

During the audit, we found the following information:

1. The owner role can set distributor and the distributor role can mine the tokens to the Distributor address through the mine function

The source code:

```
//SlowMist// The contract does not have the Overflow and the Race Conditions issue
pragma solidity 0.5.6;
//SlowMist// SafeMath security module is used, which is a recommended approach
library SafeMath {
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");

        return c;
    }

    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        return sub(a, b, "SafeMath: subtraction overflow");
    }

    function sub(uint256 a, uint256 b, string memory errorMessage) internal pure
```

```
returns (uint256) {
        require(b <= a, errorMessage);
        uint256 c = a - b;

        return c;
    }

    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b, "SafeMath: multiplication overflow");

        return c;
    }

    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        return div(a, b, "SafeMath: division by zero");
    }

    function div(uint256 a, uint256 b, string memory errorMessage) internal pure
returns (uint256) {
        require(b > 0, errorMessage);
        uint256 c = a / b;

        return c;
    }

    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        return mod(a, b, "SafeMath: modulo by zero");
    }

    function mod(uint256 a, uint256 b, string memory errorMessage) internal pure
returns (uint256) {
        require(b != 0, errorMessage);
        return a % b;
    }
}

library Address {
    function isContract(address account) internal view returns (bool) {
        bytes32 codehash;
```

```
        bytes32 accountHash =
0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470;
        assembly { codehash := extcodehash(account) }
        return (codehash != accountHash && codehash != 0x0);
    }

    function toPayable(address account) internal pure returns (address payable) {
        return address(uint160(account));
    }

    function sendValue(address payable recipient, uint256 amount) internal {
        require(address(this).balance >= amount, "Address: insufficient balance");

        (bool success, ) = recipient.call.value(amount)("");
        require(success, "Address: unable to send value, recipient may have
reverted");
    }
}

interface IKIP7Receiver {
    function onKIP7Received(address _operator, address _from, uint256 _amount, bytes
calldata _data) external returns (bytes4);
}

contract META {
    using SafeMath for uint256;
    using Address for address;

    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);

    address private _owner;

    string private _name;
    string private _symbol;
    uint8 private _decimals = 18;
    uint256 private _totalSupply;

    mapping (address => uint256) private _balances;
    mapping (address => mapping (address => uint256)) private _allowances;
    mapping (bytes4 => bool) private _supportedInterfaces;

    bytes4 private _KIP7_RECEIVED = 0x9d188c22;
    bytes4 private constant _INTERFACE_ID_KIP13 = 0x01ffc9a7;
```

```
    bytes4 private constant _INTERFACE_ID_KIP7 = 0x65787371;
    bytes4 private constant _INTERFACE_ID_KIP7_METADATA = 0xa219a025;
    bytes4 private constant _INTERFACE_ID_KIP7BURNABLE = 0x3b5a0bf8;

    // ======== mining ========
    address public distributor;

    uint256 public miningAmount;  // 총 마이닝으로 배포할 수량
    uint256 public blockAmount;
    uint256 public minableBlock;  // mining 시작 시점(block number) -- *이 블록부터* 마이닝
시작
    uint256 public lastMined;

    constructor(
        string memory name,
        string memory symbol,
        uint256 _minableBlock,
        address initialSupplyClaimer,
        uint256 initialSupply
    ) public {
        _registerInterface(_INTERFACE_ID_KIP13);
        _registerInterface(_INTERFACE_ID_KIP7);
        _registerInterface(_INTERFACE_ID_KIP7_METADATA);
        _registerInterface(_INTERFACE_ID_KIP7BURNABLE);

        _owner = msg.sender;
        _name = name;
        _symbol = symbol;
        blockAmount = 2109000000000000000;

        require(_minableBlock > block.number);
        minableBlock = _minableBlock;

        uint total = 200000000 * 10 ** 18;

        require(initialSupplyClaimer != address(0));
        require(initialSupply <= total);
        _mint(initialSupplyClaimer, initialSupply);

        miningAmount = total.sub(initialSupply);
        _mint(address(this), miningAmount);
    }

    ////////////////////////// ownable ////////////////////////
```

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

    function owner() public view returns (address) {
        return _owner;
    }

    event OwnershipTransferred(address indexed previousOwner, address indexed
newOwner);
    function renounceOwnership() public onlyOwner {
        emit OwnershipTransferred(_owner, address(0));
        _owner = address(0);
    }

    function transferOwnership(address newOwner) public onlyOwner {
        //SlowMist// This check is quite good in avoiding losing control of the
contract caused by user mistakes
        require(newOwner != address(0), "Ownable: new owner is the zero address");
        emit OwnershipTransferred(_owner, newOwner);
        _owner = newOwner;
    }

    event SetDistributor(address indexed previousDistributor, address indexed
newDistributor);
    function setDistributor(address _distributor) public onlyOwner {
        require(_distributor != address(0));
        emit SetDistributor(distributor, _distributor);
        distributor = _distributor;
    }
    ////////////////////////////////////////////////////////

    ////////////////////////// mining /////////////////////////
    function mined() public view returns (uint256) {
        uint256 endBlock = block.number;
        uint256 startBlock = minableBlock;
        if (endBlock < startBlock) return 0;

        uint256 curMined = ((endBlock.sub(startBlock)).add(1)).mul(blockAmount);

        return curMined > miningAmount ? miningAmount : curMined;
    }
```

```
modifier onlyDistributor {
    require(msg.sender == distributor);
    _;
}

event Mine(uint256 thisMined, uint256 lastMined, uint256 curMined);
function mine() public onlyDistributor {
    uint256 curMined = mined();
    uint256 thisMined = curMined.sub(lastMined);
    if(thisMined == 0) return;

    emit Mine(thisMined, lastMined, curMined);

    lastMined = curMined;
    _transfer(address(this), distributor, thisMined);
}
////////////////////////////////////////////////////////

////////////////////// KIP7 Standard //////////////////////
function supportsInterface(bytes4 interfaceId) external view returns (bool) {
    return _supportedInterfaces[interfaceId];
}

function name() public view returns (string memory) {
    return _name;
}

function symbol() public view returns (string memory) {
    return _symbol;
}

function decimals() external view returns (uint8) {
    return _decimals;
}

function totalSupply() public view returns (uint256) {
    return _totalSupply;
}

function balanceOf(address account) public view returns (uint256) {
    return _balances[account];
}

function transfer(address recipient, uint256 amount) public returns (bool) {
```

```
        _transfer(msg.sender, recipient, amount);
        //SlowMist// The return value conforms to the KIP7 specification
        return true;
    }

    function allowance(address owner_, address spender) public view returns (uint256)
{
        return _allowances[owner_][spender];
    }

    function approve(address spender, uint256 value) public returns (bool) {
        _approve(msg.sender, spender, value);
        //SlowMist// The return value conforms to the KIP7 specification
        return true;
    }

    function increaseAllowance(address spender, uint256 addedValue) public returns
(bool) {
        _approve(msg.sender, spender, _allowances[msg.sender]
[spender].add(addedValue));
        return true;
    }

    function decreaseAllowance(address spender, uint256 subtractedValue) public
returns (bool) {
        _approve(msg.sender, spender, _allowances[msg.sender]
[spender].sub(subtractedValue));
        return true;
    }

    function transferFrom(address sender, address recipient, uint256 amount) public
returns (bool) {
        _transfer(sender, recipient, amount);
        _approve(sender, msg.sender, _allowances[sender][msg.sender].sub(amount));
        //SlowMist// The return value conforms to the KIP7 specification
        return true;
    }

    function safeTransfer(address recipient, uint256 amount) public {
        safeTransfer(recipient, amount, "");
    }

    function safeTransfer(address recipient, uint256 amount, bytes memory data)
public {
```

```
        transfer(recipient, amount);
        require(_checkOnKIP7Received(msg.sender, recipient, amount, data), "KIP7:
transfer to non KIP7Receiver implementer");
    }

    function safeTransferFrom(address sender, address recipient, uint256 amount)
public {
        safeTransferFrom(sender, recipient, amount, "");
    }

    function safeTransferFrom(address sender, address recipient, uint256 amount,
bytes memory data) public {
        transferFrom(sender, recipient, amount);
        require(_checkOnKIP7Received(sender, recipient, amount, data), "KIP7:
transfer to non KIP7Receiver implementer");
    }

    function burn(uint256 amount) public {
        _burn(msg.sender, amount);
    }

    function _transfer(address sender, address recipient, uint256 amount) internal {
        require(sender != address(0), "KIP7: transfer from the zero address");
        //SlowMist// This kind of check is very good, avoiding user mistake leading
to the loss of token during transfer
        require(recipient != address(0), "KIP7: transfer to the zero address");

        _balances[sender] = _balances[sender].sub(amount);
        _balances[recipient] = _balances[recipient].add(amount);
        emit Transfer(sender, recipient, amount);
    }

    function _mint(address account, uint256 amount) internal {
        require(account != address(0), "KIP7: mint to the zero address");

        _totalSupply = _totalSupply.add(amount);
        _balances[account] = _balances[account].add(amount);
        emit Transfer(address(0), account, amount);
    }

    function _burn(address account, uint256 value) internal {
        require(account != address(0), "KIP7: burn from the zero address");

        _totalSupply = _totalSupply.sub(value);
```

```
        _balances[account] = _balances[account].sub(value);
        emit Transfer(account, address(0), value);
    }

    function _approve(address owner_, address spender, uint256 value) internal {
        require(owner_ != address(0), "KIP7: approve from the zero address");
        //SlowMist// This kind of check is very good, avoiding user mistake leading
to approve errors
        require(spender != address(0), "KIP7: approve to the zero address");

        _allowances[owner_][spender] = value;
        emit Approval(owner_, spender, value);
    }

    function _checkOnKIP7Received(address sender, address recipient, uint256 amount,
bytes memory _data)
        internal returns (bool)
    {
        if (!recipient.isContract()) {
            return true;
        }

        bytes4 retval = IKIP7Receiver(recipient).onKIP7Received(msg.sender, sender,
amount, _data);
        return (retval == _KIP7_RECEIVED);
    }

    function _registerInterface(bytes4 interfaceId) internal {
        require(interfaceId != 0xffffffff, "KIP13: invalid interface id");
        _supportedInterfaces[interfaceId] = true;
    }
    ////////////////////////////////////////////////////////
}
```

Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist