Smart Contract Security Audit Report

The SlowMist Security Team received the NFUP team's application for smart contract security audit of the NFUP_DOC_V06 on Aug. 31, 2020. The following are the details and results of this smart contract security audit:

Token name :

NFUP

The Contract address :

0xcdeeef3551d61434a7d21243e80717ac556f513f

Link address :

https://etherscan.io/address/0xcdeeef3551d61434a7d21243e80717ac556f513f

As of our report issuance time: 2020-09-01 10:00(UTC), the proxy contract address for this contract is 0x26CBC7008cD879f4B63B69a915378f2D9b17bBF0

Reference:

txid: 0x3a641c8c46d1335444eda6b33fad7c52a07a7a6a8aff860606ac5cbe47cdf562

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 : 0X002009010002

Audit Date : Sep. 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, admin can burn their own tokens through the burn function. OpenZeppelin's SafeMath security module is used, which is a commendable approach. The contract does not have the Overflow and the Race Conditions issue.

The source code:

```
/**
 *Submitted for verification at Etherscan.io on 2020-08-31
*/

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

```
pragma solidity 0.5.0;

//SlowMist// OpenZeppelin's SafeMath security Module is used, which is a recommend 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, string memory errorMessage) internal pure returns (uint256) {
        require(b <= a, errorMessage);
        uint256 c = a - b;
        return c;
    }
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {return sub(a, b, "SafeMath: subtraction overflow");}
}


contract DOC_v06 {
    using SafeMath for uint256;
    mapping (address => uint256) private _balances;
    mapping (address => mapping (address => uint256)) private _allowances;

    address private admin;
    string private _name;
    string private _symbol;
    uint8 private _decimals;
    uint256 private _totalSupply;
    string private last_useVersion;

    struct LockDetails{
        uint256 lockedTokencnt;
        uint256 releaseTime;
    }
    struct managerDetail{
        string managername;
        uint8 managerlevel;
    }
    mapping(address => LockDetails) private Locked_list;
    address[] private managerList;
    mapping(address => managerDetail) private Managers;
    mapping(address => mapping(bytes32 => string)) user_dataList;
```

```
    mapping (address => mapping (address => uint256)) private _allowances_new;

    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);
    ////////////////////////////////////// Mint handle ////////////////////////////////////      function Contadmin() public view returns (address) {return admin;}
    function totalSupply() public view returns (uint256) {return _totalSupply;}
    function name() public view returns (string memory) {return _name;}
    function symbol() public view returns (string memory) {return _symbol;}
    function getlast_useVersion() public view returns (string memory) {return last_useVersion;}
    function decimals() public view returns (uint8) {return _decimals;}
    ////////////////////////////////////// manager handle //////////////////////////////////////
    //function admin_Add_manager(address adr, string memory mname, uint8 mlevel) public returns (bool) {
    //    managerDetail memory isManager = Managers[msg.sender];
    //    if(msg.sender != admin){
    //        require( isManager.managerlevel > 14, "This is manager level over 15 only ecode-01");
    //        if( mlevel > 15 ) mlevel = 15;
    //    }
    //    require( msg.sender != adr , "Can not change self infomation ! ecode-01");
    //
    //    isManager = Managers[adr];
    //    bytes memory a1 = bytes(isManager.managername);
    //    bytes memory a2 = bytes("del");
    //    if(keccak256(a1) == keccak256(a2)) {
    //        isManager.managername = mname;
    //        isManager.managerlevel = mlevel;
    //    }else if( isManager.managerlevel != 0 ){
    //        isManager.managername = mname;
    //        isManager.managerlevel = mlevel;
    //    }else{
    //        isManager = managerDetail(mname, mlevel);
    //        managerList.push(adr);
    //    }
    //    Managers[adr] = isManager;
    //    return true;
    //}
    //function get_nth_adr_manager(uint256 nth) public view returns (address) {
    //    //managerDetail memory isManager = Managers[msg.sender];
    //    //require( isManager.managerlevel > 14, "This is manager level over 15 only ecode-02");
    //    require( nth > 0 && nth <= managerList.length,"outofrange");
    //    return managerList[nth];
    //}
```

```
//function remove_manager( address adr) public returns (bool) {
//      require( admin != adr, "contract creater cannot be deleted");
//
//      managerDetail memory isManager = Managers[msg.sender];
//      require( isManager.managerlevel > 14, "This is manager level over 15 only ecode-03");
//
//      managerDetail memory DeleteTarget = Managers[adr];
//
//      require( DeleteTarget.managerlevel < 15, "Only super admin can delete level 15 ecode-04");
//      isManager = managerDetail("del", 0);
//      Managers[adr] = isManager;
//      return true;
//}

//function get_count_manager() public view returns (uint256) {
//      //managerDetail memory isManager = Managers[msg.sender];
//      //require( isManager.managerlevel > 14, "This is manager level over 15 only ecode-04");
//      return managerList.length;
//}
//function get_managername(address adr) public view returns (string memory) {
//      //managerDetail memory isManager = Managers[msg.sender];
//      //require( isManager.managerlevel > 14, "This is manager level over 15 only ecode-05");
//      managerDetail memory isManager = Managers[adr];
//      return isManager.managername;
//}

//function get_managerLevel(address adr) public view returns (uint8) {
//      managerDetail memory isManager = Managers[msg.sender];
//      //require( isManager.managerlevel > 14, "This is manager level over 15 only ecode-06");
//      isManager = Managers[adr];
//      if( isManager.managerlevel > 0 ){
//            return isManager.managerlevel;
//      }else{
//            return 0;
//      }
//}

////////////////////////////////////// Lock token handle //////////////////////////////////////
//function Lock_wallet(address _adr, uint256 lockamount,uint256 releaseTime ) public returns (bool) {
//      require(Managers[msg.sender].managerlevel > 9 , "NFUP Manager only");
//      _Lock_wallet(_adr,lockamount,releaseTime);
//      return true;
```

```
    //}
    //function _Lock_wallet(address account, uint256 amount,uint256 releaseTime) internal {
    //      LockDetails memory eaLock = Locked_list[account];
    //      if( eaLock.releaseTime > 0 ){
    //            eaLock.lockedTokencnt = amount;
    //            eaLock.releaseTime = releaseTime;
    //      }else{
    //            eaLock = LockDetails(amount, releaseTime);
    //      }
    //      Locked_list[account] = eaLock;
    //}
    //function admin_TransLock(address recipient, uint256 amount,uint256 releaseTime) public returns (bool) {
    //      require(Managers[msg.sender].managerlevel > 9 , "NFUP Manager only");
    //      require(recipient != address(0), "ERC20: transfer to the zero address");
    //       _Lock_wallet(recipient,amount,releaseTime);
    //      _transfer(msg.sender, recipient, amount);
    //      return true;
    //}
    //function getwithdrawablemax(address account) public view returns (uint256) {
    //      return Locked_list[account].lockedTokencnt;
    //}
    function getLocked_list(address account) public view returns (uint256) {
        return Locked_list[account].releaseTime;
    }
    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);

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

    }
    function _transfer(address sender, address recipient, uint256 amount) internal {
        last_useVersion = "Ver 1";
        require(sender != address(0), "ERC20: transfer from the zero address");

        require(recipient != address(0), "ERC20: transfer to the zero address"); //SlowMist// This kind of check is
very good, avoiding user mistake leading to the loss of token during transfer

        uint256 LockhasTime = Locked_list[sender].releaseTime;
        uint256 LockhasMax = Locked_list[sender].lockedTokencnt;
        if( block.timestamp < LockhasTime){
```

```
            require( _balances[sender] >= amount , "You have not enough balance.");
            uint256 OK1 = _balances[sender].sub( LockhasMax);
            require( OK1 >= amount , "Your Wallet has time lock");
        }

        _balances[sender] = _balances[sender].sub(amount, "ERC20: transfer amount exceeds balance");
        _balances[recipient] = _balances[recipient].add(amount);
        emit Transfer(sender, recipient, amount);
    }
    function allowance(address owner, address spender) public view returns (uint256) {
        return _allowances_new[owner][spender];
    }
    function approve(address spender, uint256 amount) public returns (bool) {
        _approve(msg.sender, spender, amount);
        return true; //SlowMist// The return value conforms to the EIP20 specification
    }
    function transferFrom(address sender, address recipient, uint256 amount) public returns (bool) {
        _approve(sender, msg.sender, _allowances_new[sender][msg.sender].sub(amount, "ERC20: transfer amount
exceeds allowance"));
        _transfer(sender, recipient, amount);
        return true; //SlowMist// The return value conforms to the EIP20 specification
    }
    function increaseAllowance(address spender, uint256 addedValue) public returns (bool) {
        _approve(msg.sender, spender, _allowances_new[msg.sender][spender].add(addedValue));
        return true;
    }
    function decreaseAllowance(address spender, uint256 subtractedValue) public returns (bool) {
        _approve(msg.sender, spender, _allowances_new[msg.sender][spender].sub(subtractedValue, "ERC20: decreased
allowance below zero"));
        return true;
    }
    function burn(uint256 amount) public returns (bool) {
        _burn(msg.sender, amount);
    }


    function _burn(address account, uint256 amount) internal {
        require(account != address(0), "ERC20: burn from the zero address");
        require(msg.sender == admin, "Admin only can burn   8547");
```

```
        _balances[account] = _balances[account].sub(amount, "ERC20: burn amount exceeds balance");
        _totalSupply = _totalSupply.sub(amount);
        emit Transfer(account, address(0), amount);
    }
    function _approve(address owner, address spender, uint256 amount) internal {
        require(owner != address(0), "ERC20: approve from the zero address");

        require(spender != address(0), "ERC20: approve to the zero address"); //SlowMist// This kind of check is very good, avoiding user mistake leading to approve errors


        _allowances_new[owner][spender] = amount;
        emit Approval(owner, spender, amount);
    }

    //////////////////////////////////////// Lock token handle ////////////////////////////////////////

    function getStringData(bytes32 key) public view returns (string memory) {
        return user_dataList[msg.sender][key];
    }
    function setStringData(bytes32 key, string memory value) public {
        user_dataList[msg.sender][key] = value;
    }
}
```

慢雾科技
slow mist

Official Website
www.slowmist.com

E-mail
team@slowmist.com

Twitter
@SlowMist_Team

WeChat Official Account