Smart Contract Security Audit Report

[2021]

The SlowMist Security Team received the UBSN team's application for smart contract security audit of the UBSN on 2021.10.19. The following are the details and results of this smart contract security audit:

Token Name :

UBSN

The contract address :

https://etherscan.io/address/0x86EFc496DcA70bcFD92D19194290e8457a375773

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 : 0X002110210004

Audit Date : 2021.10.19 - 2021.10.21

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 remains unchangeable. 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 decimals is set to 0.
2. The constructor uses the function of the same name.

The source code:

```
/**
 *Submitted for verification at Etherscan.io on 2021-09-08
*/

/**
 *Submitted for verification at Etherscan.io on 2020-05-06
*/
//SlowMist// The contract does not have the Overflow and the Race Conditions issue
pragma solidity ^0.4.17;

/**
 * @title SafeMath
 * @dev Math operations with safety checks that throw on error
 */
//SlowMist// SafeMath security module is used, which is a recommended approach
library SafeMath {
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        if (a == 0) {
            return 0;
        }
        uint256 c = a * b;
        //SlowMist// It is recommended to replace "assert" with "require" to optimize
Gas
        assert(c / a == b);
        return c;
    }

    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        // assert(b > 0); // Solidity automatically throws when dividing by 0
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in which this doesn't hold
        return c;
    }

    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        //SlowMist// It is recommended to replace "assert" with "require" to optimize
Gas
        assert(b <= a);
        return a - b;
    }

    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        //SlowMist// It is recommended to replace "assert" with "require" to optimize
Gas
        assert(c >= a);
        return c;
    }
}

/**
 * @title Ownable
 * @dev The Ownable contract has an owner address, and provides basic authorization
control
 * functions, this simplifies the implementation of "user permissions".
 */
contract Ownable {
    address public owner;

    /**
      * @dev The Ownable constructor sets the original `owner` of the contract to the
sender
      * account.
      */
    function Ownable() public {
        owner = msg.sender;
    }

    /**
      * @dev Throws if called by any account other than the owner.
      */
    modifier onlyOwner() {
        require(msg.sender == owner);
        _;
    }

    /**
    * @dev Allows the current owner to transfer control of the contract to a
newOwner.
    * @param newOwner The address to transfer ownership to.
    */
    function transferOwnership(address newOwner) public onlyOwner {
        if (newOwner != address(0)) {
            owner = newOwner;
        }
    }
}

/**
 * @title ERC20Basic
 * @dev Simpler version of ERC20 interface
 * @dev see https://github.com/ethereum/EIPs/issues/20
 */
contract ERC20Basic {
    uint public _totalSupply;
    function totalSupply() public constant returns (uint);
    function balanceOf(address who) public constant returns (uint);
    function transfer(address to, uint value) public;
    event Transfer(address indexed from, address indexed to, uint value);
}

/**
 * @title ERC20 interface
 * @dev see https://github.com/ethereum/EIPs/issues/20
 */
contract ERC20 is ERC20Basic {
    function allowance(address owner, address spender) public constant returns
(uint);
```

```
    function transferFrom(address from, address to, uint value) public;
    function approve(address spender, uint value) public;
    event Approval(address indexed owner, address indexed spender, uint value);
}

/**
 * @title Basic token
 * @dev Basic version of StandardToken, with no allowances.
 */
contract BasicToken is Ownable, ERC20Basic {
    using SafeMath for uint;

    mapping(address => uint) public balances;

    /**
    * @dev Fix for the ERC20 short address attack.
    */
    modifier onlyPayloadSize(uint size) {
        require(!(msg.data.length < size + 4));
        _;
    }

    /**
    * @dev transfer token for a specified address
    * @param _to The address to transfer to.
    * @param _value The amount to be transferred.
    */
    function transfer(address _to, uint _value) public onlyPayloadSize(2 * 32) {
        //SlowMist// require(_to != address(0));
        //SlowMist// It is recommended to add the above line check to avoid user
mistakes leading to the loss of Token
        uint sendAmount = _value;
        balances[msg.sender] = balances[msg.sender].sub(_value);
        balances[_to] = balances[_to].add(sendAmount);
        Transfer(msg.sender, _to, sendAmount);
    }

    /**
    * @dev Gets the balance of the specified address.
    * @param _owner The address to query the the balance of.
    * @return An uint representing the amount owned by the passed address.
    */
    function balanceOf(address _owner) public constant returns (uint balance) {
        return balances[_owner];
    }
```

```
}

/**
 * @title Standard ERC20 token
 *
 * @dev Implementation of the basic standard token.
 * @dev https://github.com/ethereum/EIPs/issues/20
 * @dev Based oncode by FirstBlood:
https://github.com/Firstbloodio/token/blob/master/smart_contract/FirstBloodToken.sol
 */
contract StandardToken is BasicToken, ERC20 {

    mapping (address => mapping (address => uint)) public allowed;

    uint public constant MAX_UINT = 2**256 - 1;

    /**
    * @dev Transfer tokens from one address to another
    * @param _from address The address which you want to send tokens from
    * @param _to address The address which you want to transfer to
    * @param _value uint the amount of tokens to be transferred
    */
    function transferFrom(address _from, address _to, uint _value) public
onlyPayloadSize(3 * 32) {
        var _allowance = allowed[_from][msg.sender];
        //SlowMist// require(_to != address(0));
        //SlowMist// It is recommended to add the above line check to avoid user
mistakes leading to the loss of Token
        // Check is not needed because sub(_allowance, _value) will already throw if
this condition is not met
        // if (_value > _allowance) throw;

        if (_allowance < MAX_UINT) {
            allowed[_from][msg.sender] = _allowance.sub(_value);
        }
        uint sendAmount = _value;
        balances[_from] = balances[_from].sub(_value);
        balances[_to] = balances[_to].add(sendAmount);
        Transfer(_from, _to, sendAmount);
    }

    /**
    * @dev Approve the passed address to spend the specified amount of tokens on
behalf of msg.sender.
```

```
    * @param _spender The address which will spend the funds.
    * @param _value The amount of tokens to be spent.
    */
    function approve(address _spender, uint _value) public onlyPayloadSize(2 * 32) {

        // To change the approve amount you first have to reduce the addresses`
        //  allowance to zero by calling `approve(_spender, 0)` if it is not
        //  already 0 to mitigate the race condition described here:
        //  https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
        require(!((_value != 0) && (allowed[msg.sender][_spender] != 0)));

        allowed[msg.sender][_spender] = _value;
        Approval(msg.sender, _spender, _value);
    }

    /**
    * @dev Function to check the amount of tokens than an owner allowed to a spender.
    * @param _owner address The address which owns the funds.
    * @param _spender address The address which will spend the funds.
    * @return A uint specifying the amount of tokens still available for the spender.
    */
    function allowance(address _owner, address _spender) public constant returns
(uint remaining) {
        return allowed[_owner][_spender];
    }
}

contract OurToken is StandardToken {

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

    //  The contract can be initialized with a number of tokens
    //  All the tokens are deposited to the owner address
    //
    // @param _balance Initial supply of the contract
    // @param _name Token Name
    // @param _symbol Token symbol
    //SlowMist// The decimals is set to 0
    function OurToken(uint _initialSupply, string _name, string _symbol) public {
        _totalSupply = _initialSupply;
        name = _name;
        symbol = _symbol;
        decimals = 0;
        balances[owner] = _initialSupply;
    }

    function totalSupply() public constant returns (uint) {
        return _totalSupply;
    }
}
```

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.