慢雾科技
slow mist

Smart Contract Security Audit Report

The SlowMist Security Team received the team's application for smart contract security audit of the SUN Token on Sep. 02, 2020. The following are the details and results of this smart contract security audit:

Token name :

SUN

The File Name and HASH(SHA256):

SafeMath.sol:

157cffe42b43d7a96f99275d82e1d97490cd336c8d19c8a8ff9144aee7e6b108

SunStakerInterface.sol:

e481c5097f01c1b07b8002fae8557941d5626b1b43023d4dd205eceb1cf43f14

SunStakerSimpleStandAlone.sol:

c2e6a5985af5d19087d8c0b385833879ffaa50a89527e4719372561d41bf2b43

SunStakerStorage.sol:

af04efe20bc038846ef6dfa5aeef1d7954cf9b1a8abdbd35bf711452f99755d8

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 : 0X002009020003

Audit Date : Sep. 02, 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 does not contain the tokenVault section. OpenZeppelin's SafeMath security Module is used, which is a recommend approach. The comprehensive evaluation contract is no risk.

The source code:

SunStakeSimpleStandAlone.sol

```
import "./SunStakerInterface.sol";
pragma solidity ^0.5.8;


contract SunStakerSimpleStandAlone is SunStakerInterface {
    using   SafeMath for uint256;
```

```
modifier checkStart() {
    require(block.timestamp >= starttime , "not started");
    require(block.timestamp < periodFinish, "already ended");
_;
}

modifier checkEnd() {
    require(block.timestamp >= periodFinish, "not end");
    _;
}

event  Rescue(address indexed dst, uint sad);
event  Deposit(address indexed dst, uint sad);
event  Withdrawal(address indexed src, uint sad);

constructor(uint256 _starttime, uint256 _periodFinish) public{
    starttime = _starttime;
    periodFinish = _periodFinish;
}

function() external payable {
    deposit();
}
//SlowMist// This function is currently meaningless
function rewardOneSun() public view returns (uint256) {
    return 0;
}
//SlowMist// This function is currently meaningless
function earned(address account) public view returns (uint256) {
    return 0;
}
//SlowMist// This function is currently meaningless
function earned(address account, address token) public view returns (uint256) {
    return 0;
}
//SlowMist// This function is currently meaningless
```

```
    function lastTimeRewardApplicable() public view returns (uint256) {
        return 0;
    }

    function deposit() checkStart public payable {
        require(msg.value > 0, "deposit must gt 0");
        balanceOf_[msg.sender] = balanceOf_[msg.sender].add(msg.value);
        totalSupply_ = totalSupply_.add(msg.value);
        emit Deposit(msg.sender, msg.value);
    }

    function withdraw(address token) checkEnd public {
        token;
        uint256 sad = balanceOf_[msg.sender];
        require(sad > 0, "balance must gt 0");
        sad = min(sad, totalSupply_);
        balanceOf_[msg.sender] = 0;
        msg.sender.transfer(sad);
        totalSupply_ = totalSupply_.sub(sad);
        emit Withdrawal(msg.sender, sad);
    }

    function totalSupply() public view returns (uint) {
        return totalSupply_;
    }

    function balanceOf(address guy) public view returns (uint){
        return balanceOf_[guy];
    }

    function getInfo(address _user) public view returns(uint256 _balanceTRX, uint256 _balance, uint256 _totalSupply){
        _balanceTRX = _user.balance;
        _balance = balanceOf_[_user];
        _totalSupply = totalSupply_;
    }

    /**         * @dev rescue simple transfered TRX.         */
    function rescue(address payable to_, uint256 amount_) checkEnd public{
        require(msg.sender == gov, "must gov");
        require(to_ != address(0), "must not 0");
        require(amount_ > 0, "must gt 0");
        uint256 sad = min(address(this).balance.sub(totalSupply_), amount_);
        to_.transfer(sad);
        emit Rescue(to_, sad);
    }

    /**      * @dev Returns the smallest of two numbers.      */
    function min(uint256 a, uint256 b) internal pure returns (uint256) {
        return a < b ? a : b;
    }
}
```

SunStakeInterface.sol

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

pragma solidity ^0.5.8;
import "./SunStakerStorage.sol";import "./SafeMath.sol";
contract SunStakerInterface is SunStakerStorage {

    function deposit() public payable ;

    function withdraw(address token) public;

    function lastTimeRewardApplicable() public view returns (uint256);

    function rewardOneSun() public view returns (uint256);

    function earned(address account) public view returns (uint256);

    function earned(address account,address token) public view returns (uint256);

    function totalSupply() public view returns (uint);

    function balanceOf(address guy) public view returns (uint);
}
```

SunStroage.sol

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

pragma solidity ^0.5.8;
```

```
contract SunStakerStorage {

    uint256 internal totalSupply_;
    mapping(address => uint256) internal balanceOf_;
    mapping(address => uint256) public    rewards;
    mapping(address => uint256) public    userRewardOneSunPaid;

    uint256 public starttime;
    uint256 public periodFinish;


    uint256 public rewardRate = 10**18;
    uint256 public lastUpdateTime;
    uint256 public rewardOneSunStored;
    uint256 public rawAll = 0;

    address public gov = msg.sender;

    bool public withdrawOn;
}
```

SafeMath.sol

```
//SlowMist// The contract does not have the Overflow and the Race Conditions issue
pragma solidity ^0.5.8;
//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) 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;
    }}
```

慢雾科技
slow mist

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

WeChat Official Account