智能合约安全审计报告

慢雾安全团队于 2020-09-02 日，收到 JustSwap 团队对 SUN 项目智能合约安全审计申请。如下为本次智能合约安全审计细节及结果：

Token 名称：

SUN

合约文件名及哈希（SHA256）：

SafeMath.sol:

157cffe42b43d7a96f99275d82e1d97490cd336c8d19c8a8ff9144aee7e6b108

SunStakerInterface.sol:

e481c5097f01c1b07b8002fae8557941d5626b1b43023d4dd205eceb1cf43f14

SunStakerSimpleStandAlone.sol:

c2e6a5985af5d19087d8c0b385833879ffaa50a89527e4719372561d41bf2b43

SunStakerStorage.sol:

af04efe20bc038846ef6dfa5aeef1d7954cf9b1a8abdbd35bf711452f99755d8

本次审计项及结果：

（其他未知安全漏洞不包含在本次审计责任范围）

序号	审计大类	审计子类	审计结果
1	溢出审计	-	通过
2	条件竞争审计	-	通过
3	权限控制审计	权限漏洞审计	通过
		权限过大审计	通过
4	安全设计审计	Zeppelin 模块使用安全	通过
		编译器版本安全	通过
		硬编码地址安全	通过
		Fallback 函数使用安全	通过
		显现编码安全	通过
		函数返回值安全	通过
		call 调用安全	通过

5	拒绝服务审计	-	通过
6	Gas 优化审计	-	通过
7	设计逻辑审计	-	通过
8	“假充值”漏洞审计	-	通过
9	恶意 Event 事件日志审计	-	通过
10	变量声明及作用域审计	-	通过
11	重放攻击审计	ECDSA 签名重放审计	通过
12	未初始化的存储指针	-	通过
13	算术精度误差	-	通过

备注：审计意见及建议见代码注释 //SlowMist//……

审计结果：**通过**

审计编号：0X002009020003

审计日期：2020 年 09 月 02 日

审计团队：慢雾安全团队

总结：此为代币(token)合约，不包含锁仓(tokenVault)部分。使用了 OpenZeppelin 的 SafeMath 安全模块，值得称赞的做法。综合评估合约无风险。

合约源代码如下：

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// 此函数目前无作用

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

//SlowMist// 此函数目前无作用

```
function earned(address account) public view returns (uint256) {
    return 0;
}
```

//SlowMist// 此函数目前无作用

```
function earned(address account, address token) public view returns (uint256) {
    return 0;
}
```

//SlowMist// 此函数目前无作用

```
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// 合约不存在溢出、条件竞争问题

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// 合约不存在溢出、条件竞争问题

pragma solidity ^0.5.8;

//SlowMist// 使用了 OpenZeppelin 的 SafeMath 安全模块，值得称赞的做法

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

官方网址

www.slowmist.com

电子邮箱

team@slowmist.com

微信公众号