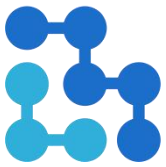
密级	商密

[文档编号:MOYUNSEC-SJ-VIP-20190516]



墨云科技
vackbot.com

VIP合约安全审计报告

版本说明

修订人	修订内容	修订时间	版本号	审阅人
ASTRO	内容编撰	20190516	V.1.0	谢鑫

文档信息

文档名称	VIP 合约安全审计报告	文档编号	MOYUNSEC-SJ-VIP-20190516
文档版本号	V.1.0	保密级别	商密
扩散范围			
扩散批准人			

文档说明

此文档作为北京墨云科技有限公司的正式文档，用于公司对外发布的各种服务报告等文档。

版权声明

目录

1. 概述

2019 年 05 月 16 日，受 VIP FOUND 委托，北京墨云科技有限公司安全服务团队对 VIPToken 合约进行了安全测试，测试发现**风险 0 处**。

1.1 测试目标

合约名称	合约地址
VIPToken	https://etherscan.io/address/0xc0d9da090194d62b2027e4009d9123de399ea7bf#code

2. 漏洞信息汇总

序号	漏洞类型	风险级别	漏洞数量
1	无	无	无

责任声明：北京墨云科技提交之本报告基于已经发生或者存在的事实出具报告，并为此承担责任。本报告只针对该项目当前提交代码有效。对出具报告以后发生或存在的事实或者审计项之外的安全问题，北京墨云科技不承担责任。

3. 总体安全现状

本次测试过程中，在 VIPToken 中发现存在 0 处安全问题

4. 安全审计项

4.1. 算术溢出漏洞

在执行加减乘除运算时，应该使用 safemath 库或者在运算前进行严格溢出检查，否则容易造成溢出漏洞。

检测结果：通过

4.2. Storage 局部变量未初始化

未初始化的 storage 局部变量会指向其他 storage 变量，导致变量覆盖。

检测结果：通过

4.3. 函数循环调用 dos

循环执行的次数来自于用户提供的参数，若未对该参数做校验，那么合约有被 dos 的风险。

检测结果：通过

4.4. 构造函数书写问题

在 0.4.22 版本以后，引入了 constructor 关键字作为构造函数声明，但不需要 function 关键字。

检测结果：通过

4.5. 除法运算向下取整

在做除法运算时，结果会向下取整，后面部分会被丢弃。

检测结果：通过

4.6. call 函数注入

call 会触发外部合约代码执行，需要对调用函数做严格安全限制。

在该合约中，含有 call 函数的使用，此处是安全使用，并不会导致 call 注入。

检测结果：通过

4.7. 函数重入漏洞

智能合约避免使用 call 来做交易，call 和 transfer 的最大区别是没有 gas 限制，这有可能导致重入漏洞。

本合约中的 call 调用并不会导致重入漏洞。

检测结果：通过

4.8. 弱随机漏洞

随机数生成时避免只使用 block 信息作为随机数种子，这会导致随机数可以被预测或者控制。

检测结果：通过

4.9. 交易顺序依赖

以太坊中的交易顺序会收到 gasprice 的影响，开发者应该考虑这个问题。

检测结果：通过

4.10. Gas 消耗不当

对某些不涉及状态变化的变量或函数，使用 constant 来修饰，节省 gas。

检测结果：通过

4.11. 函数权限控制不当

合约中的是否正确使用了 public，private，internal，onlyowner 等访问控制。

检测结果：通过

4.12. Approve 函数条件竞争

在修改 allowance 前，应先修改为 0，再修改为_value，或者使用 increaseAllowance/decreaseAllowance.

建议持币者在修改 allowance 前，应先修改为 0，再修改为_value

检测结果：通过

4.13. 假充值漏洞

转账函数中，对余额以及转账金额的安全检查，需要使用 require 函数抛出错误，否则会错误的判断为交易成功。

检测结果：通过

4.14. Owner 权限过大

合约 owner 权限过大，可以任意修改合约规则，任意转账，烧币铸造币。

检测结果：通过

5. 附录-合约代码

```
/**
 * Source Code first verified at https://etherscan.io on Sunday, April 28, 2019
 (UTC) */

pragma solidity ^0.4.8;


contract Token {
    /* This is a slight change to the ERC20 base standard.
    function totalSupply() constant returns (uint256 supply);
    is replaced with:
    uint256 public totalSupply;
```

```
        This automatically creates a getter function for the totalSupply.
        This is moved to the base contract since public getter functions are not
        currently recognised as an implementation of the matching abstract
        function by the compiler.
        */
        /// total amount of tokens
        uint256 public totalSupply;

        /// @param _owner The address from which the balance will be retrieved
        /// @return The balance
        function balanceOf(address _owner) public view returns (uint256 balance);

        /// @notice send `_value` token to `_to` from `msg.sender`
        /// @param _to The address of the recipient
        /// @param _value The amount of token to be transferred
        /// @return Whether the transfer was successful or not
        function transfer(address _to, uint256 _value) public returns (bool success);

        /// @notice send `_value` token to `_to` from `_from` on the condition it is approved by `_from`
        /// @param _from The address of the sender
        /// @param _to The address of the recipient
```

```
    /// @param _value The amount of token to be transferred
    /// @return Whether the transfer was successful or not
    function transferFrom(address _from, address _to, uint256 _value) public returns (bool success);

    /// @notice `msg.sender` approves `_spender` to spend `_value` tokens
    /// @param _spender The address of the account able to transfer the tokens
    /// @param _value The amount of tokens to be approved for transfer
    /// @return Whether the approval was successful or not
    function approve(address _spender, uint256 _value) public returns (bool success);

    /// @param _owner The address of the account owning tokens
    /// @param _spender The address of the account able to transfer the tokens
    /// @return Amount of remaining tokens allowed to spent
    function allowance(address _owner, address _spender) public view returns (uint256 remaining);

    event Transfer(address indexed _from, address indexed _to, uint256 _value);
    event Approval(address indexed _owner, address indexed _spender, uint256 _value);
}
```

```
contract StandardToken is Token {

    uint256 constant MAX_UINT256 = 2**256 - 1;

    function transfer(address _to, uint256 _value) public returns (bool success)
{
        //Default assumes totalSupply can't be over max (2^256 - 1).
        //If your token leaves out totalSupply and can issue more tokens as
time goes on, you need to check if it doesn't wrap.
        //Replace the if with this one instead.
        //require(balances[msg.sender] >= _value && balances[_to] +
_value > balances[_to]);
        require(balances[msg.sender] >= _value);
        balances[msg.sender] -= _value;
        balances[_to] += _value;
        Transfer(msg.sender, _to, _value);
        return true;
    }

    function transferFrom(address _from, address _to, uint256 _value) public
returns (bool success) {
```

```
        //same as above. Replace this line with the following if you want to protect against wrapping uints.
        //require(balances[_from] >= _value && allowed[_from][msg.sender] >= _value && balances[_to] + _value > balances[_to]);
        uint256 allowance = allowed[_from][msg.sender];
        require(balances[_from] >= _value && allowance >= _value);
        balances[_to] += _value;
        balances[_from] -= _value;
        if (allowance < MAX_UINT256) {
            allowed[_from][msg.sender] -= _value;
        }
        Transfer(_from, _to, _value);
        return true;
    }

    function balanceOf(address _owner) view public returns (uint256 balance) {
        return balances[_owner];
    }

    function approve(address _spender, uint256 _value) public returns (bool success) {
        allowed[msg.sender][_spender] = _value;
        Approval(msg.sender, _spender, _value);
```

```
        return true;
    }

    function allowance(address _owner, address _spender)
    view public returns (uint256 remaining) {
      return allowed[_owner][_spender];
    }

    mapping (address => uint256) balances;
    mapping (address => mapping (address => uint256)) allowed;
}

contract VIPToken is StandardToken {

    function VIPToken() public {
        balances[msg.sender] = initialAmount;      // Give the creator all initial
balances is defined in StandardToken.sol
        totalSupply = initialAmount;                        // Update total supply,
totalSupply is defined in Tocken.sol
    }
```

```
        function() public {


        }


        /* Approves and then calls the receiving contract */

        function approveAndCall(address _spender, uint256 _value, bytes _extraData) public returns (bool success) {

            allowed[msg.sender][_spender] = _value;

            Approval(msg.sender, _spender, _value);


            //call the receiveApproval function on the contract you want to be notified.

            //This crafts the function signature manually so one doesn't have to include a contract in here just for this.

            //receiveApproval(address _from, uint256 _value, address _tokenContract, bytes _extraData)

            //it is assumed when one does this that the call *should* succeed, otherwise one would use vanilla approve instead.


require(_spender.call(bytes4(bytes32(keccak256("receiveApproval(address,uint256,address,bytes)"))), msg.sender, _value, this, _extraData));

            return true;

        }
```

```
        string public name = "VIP";

        uint8 public decimals = 18;

        string public symbol = "VIP";

        string public version = "v1.0";

        uint256 public initialAmount = 99 * (10 ** 8) * (10 ** 18);


    }
```

墨云科技
vackbot.com

北京墨云科技有限公司
电话：010-62960509
网址：www.vackbot.com
邮箱：contact@vackbot.com