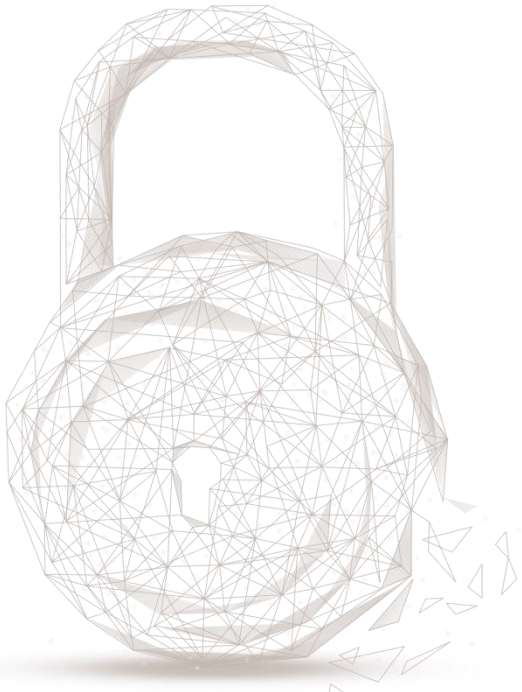
成都链安
BEOSIN

智能合约安全审计报告



审计编号：202009101516

审计合约名称：

MoonPool.finance (MFI)

审计合约地址：

TLuekEszx5fq61rPCLwga9c3RtAVyBc46j

审计合约源码地址：

https://tronscan.org/#/contract/TLuekEszx5fq61rPCLwga9c3RtAVyBc46j/code

合约审计开始日期：2020.09.09

合约审计完成日期：2020.09.10

审计结果：通过（优）

审计团队：成都链安科技有限公司

审计类型及结果：

序号	审计类型	审计子项	审计结果
1	代码规范审计	TRC20 Token 标准规范审计	通过
		编译器版本安全审计	通过
		可见性规范审计	通过
		gas 消耗审计	通过
		SafeMath 功能审计	通过
		fallback 函数使用审计	通过
		tx.origin 使用审计	通过
		弃用项审计	通过
		冗余代码审计	通过
		变量覆盖审计	通过
2	函数调用审计	函数调用权限审计	通过
		call/delegatecall 安全审计	通过
		返回值安全审计	通过
		自毁函数安全审计	通过
3	业务安全审计	owner 权限审计	通过
		业务逻辑审计	通过
		业务实现审计	通过
4	整型溢出审计	-	通过
5	可重入攻击审计	-	通过
6	异常可达状态审计	-	通过
7	交易顺序依赖审计	-	通过
8	块参数依赖审计	-	通过
9	伪随机数生成审计	-	通过
10	拒绝服务攻击审计	-	通过

11	代币锁仓审计	-	无锁仓
12	假充值审计	-	通过
13	event 安全审计	-	通过

备注：审计意见及建议请见代码注释。

免责声明：本次审计仅针对本报告载明的审计类型及结果表中给定的审计类型范围进行审计，其他未知安全漏洞不在本次审计责任范围之内。成都链安科技仅根据本报告出具前已经存在或发生的攻击或漏洞出具本报告，对于出具以后存在或发生的新的攻击或漏洞，成都链安科技无法判断其对智能合约安全状况可能的影响，亦不对此承担责任。本报告所作的安全审计分析及其他内容，仅基于合约提供者在本报告出具前已向成都链安科技提供的文件和资料，且该部分文件和资料不存在任何缺失、被篡改、删减或隐瞒的前提下作出的；如提供的文件和资料存在信息缺失、被篡改、删减、隐瞒或反映的情况与实际情况不符等情况或提供文件和资料在本报告出具后发生任何变动的，成都链安科技对由此而导致的损失和不利影响不承担任何责任。成都链安科技出具的本审计报告系根据合约提供者提供的文件和资料依靠成都链安科技现掌握的技术而作出的，由于任何机构均存在技术的局限性，成都链安科技作出的本审计报告仍存在无法完整检测出全部风险的可能性，成都链安科技对由此产生的损失不承担任何责任。

本声明最终解释权归成都链安科技所有。

审计结果说明：

本公司采用形式化验证、静态分析、动态分析、典型案例测试和人工审核的方式对智能合约MFI的代码规范性、安全性以及业务逻辑三个方面进行多维度全面的安全审计。**经审计，MFI合约通过所有检测，合约审计结果为通过(优)，合约可正常使用。**以下为本合约基本信息。

1、代币基本信息

Token name	MoonPool.finance
Token symbol	MFI
decimals	18
totalSupply	当前总量约7278(可增发，铸币上限为10000)
Token type	TRC20

表1 代币基本信息

2、代币锁仓信息

无锁仓

3、自定义函数功能说明

- 铸币

本合约可进行铸币，拥有铸币权限的地址可向指定地址进行铸币，铸币上限为10000，代币总量恒小于10000。

合约源代码审计注释：

```
pragma solidity ^0.5.4; // 成都链安 // 建议固定编译器版本

interface IERC20 {
    // 成都链安 // 定义 TRC20 Token 标准要求的接口函数
    function totalSupply() external view returns (uint);
    function balanceOf(address account) external view returns (uint);
    function transfer(address recipient, uint amount) external returns (bool);
    function allowance(address owner, address spender) external view returns (uint);
    function approve(address spender, uint amount) external returns (bool);
    function transferFrom(address sender, address recipient, uint amount) external returns
(bool);
    // 成都链安 // 声明代币转账事件与代币授权事件
    event Transfer(address indexed from, address indexed to, uint value);
    event Approval(address indexed owner, address indexed spender, uint value);
}

contract Context {
    constructor () internal { }
    // solhint-disable-previous-line no-empty-blocks
    // 成都链安 // 内部函数_msgSender，获取函数调用者地址
    function _msgSender() internal view returns (address payable) {
        return msg.sender;
    }
}

contract ERC20 is Context, IERC20 {
    using SafeMath for uint; // 成都链安 // 引入 SafeMath 安全数学运算库，避免数学运算整型溢
出

    mapping (address => uint) private _balances; // 成都链安 // 声明 mapping 变量_balances，
存储指定地址的代币余额

    mapping (address => mapping (address => uint)) private _allowances; // 成都链安 // 声明
mapping 变量_allowances，存储对应地址间的授权值

    uint private _totalSupply; // 成都链安 // 声明变量_totalSupply，存储代币总量
    uint public maxSupply = 10000 * 1e18; // 成都链安 // 声明变量 maxSupply，存储铸币上限
    // 成都链安 // 代币总量查询函数
    function totalSupply() public view returns (uint) {
        return _totalSupply;
    }
    // 成都链安 // 用户代币余额查询函数
    function balanceOf(address account) public view returns (uint) {
        return _balances[account];
    }
    // 成都链安 // 转账函数，调用者向指定地址转账一定数量代币
```

```
    function transfer(address recipient, uint amount) public returns (bool) {
        _transfer(_msgSender(), recipient, amount); // 成都链安 // 调用内部函数_transfer 进行代币转账
        return true;
    }
    // 成都链安 // 授权值查询函数
    function allowance(address owner, address spender) public view returns (uint) {
        return _allowances[owner][spender];
    }
    // 成都链安 // 用户调用该函数修改授权值时，可能导致多重授权。建议用户使用 increaseAllowance 与 decreaseAllowance 修改授权值
    function approve(address spender, uint amount) public returns (bool) {
        _approve(_msgSender(), spender, amount); // 成都链安 // 调用内部函数_approve 设置调用者对 spender 的授权值
        return true;
    }
    // 成都链安 // 代理转账函数，调用者代理代币持有者向指定地址转账一定数量代币
    function transferFrom(address sender, address recipient, uint amount) public returns (bool) {
        _transfer(sender, recipient, amount); // 成都链安 // 调用内部函数_transfer 进行代币转账
        _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount exceeds allowance")); // 成都链安 // 调用内部函数_approve 更新转账源地址 sender 对调用者的授权值
        return true;
    }
    // 成都链安 // 增加授权值函数，调用者增加对 spender 的授权值
    function increaseAllowance(address spender, uint addedValue) public returns (bool) {
        _approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue)); // 成都链安 // 调用内部函数_approve 增加调用者对 spender 的授权值，增加值为 addedValue
        return true;
    }
    // 成都链安 // 减少授权值函数，调用者减少对 spender 的授权值
    function decreaseAllowance(address spender, uint subtractedValue) public returns (bool) {
        _approve(_msgSender(), spender, _allowances[_msgSender()][spender].sub(subtractedValue, "ERC20: decreased allowance below zero")); // 成都链安 // 调用内部函数_approve 减少调用者对 spender 的授权值，减少值为 subtractedValue
        return true;
    }
    // 成都链安 // 内部函数_transfer，进行转账操作
    function _transfer(address sender, address recipient, uint amount) internal {
        require(sender != address(0), "ERC20: transfer from the zero address"); // 成都链安 // sender 非零地址检查
        require(recipient != address(0), "ERC20: transfer to the zero address"); // 成都链安 // recipient 非零地址检查，避免转账代币丢失
        // 成都链安 // 修改转账双方地址的代币余额
```

```
        _balances[sender] = _balances[sender].sub(amount, "ERC20: transfer amount exceeds balance");
        _balances[recipient] = _balances[recipient].add(amount);
        emit Transfer(sender, recipient, amount); // 成都链安 // 触发 Transfer 事件
    }
    // 成都链安 // 内部函数_mint，进行铸币操作
    function _mint(address account, uint amount) internal {
        require(account != address(0), "ERC20: mint to the zero address"); // 成都链安 // account 非零地址检查
        require(_totalSupply.add(amount) < maxSupply, "ERC20: cannot mint over max supply"); // 成都链安 // 要求本次铸币后代币总量小于铸币上限

        _totalSupply = _totalSupply.add(amount); // 成都链安 // 更新代币总量
        _balances[account] = _balances[account].add(amount); // 成都链安 // 修改代币销毁地址 account 的代币余额
        emit Transfer(address(0), account, amount); // 成都链安 // 触发 Transfer 事件
    }
    // 成都链安 // 内部函数_burn，进行代币销毁操作
    // 成都链安 // 注意：该内部函数未被任何 public/external 类型函数调用，属于冗余代码，建议删除
    function _burn(address account, uint amount) internal {
        require(account != address(0), "ERC20: burn from the zero address"); // 成都链安 // account 非零地址检查

        _balances[account] = _balances[account].sub(amount, "ERC20: burn amount exceeds balance"); // 成都链安 // 修改代币销毁地址 account 的代币余额
        _totalSupply = _totalSupply.sub(amount); // 成都链安 // 更新代币总量
        emit Transfer(account, address(0), amount); // 成都链安 // 触发 Transfer 事件
    }
    // 成都链安 // 内部函数_approve，进行授权值设置操作
    function _approve(address owner, address spender, uint amount) internal {
        require(owner != address(0), "ERC20: approve from the zero address"); // 成都链安 // owner 非零地址检查
        require(spender != address(0), "ERC20: approve to the zero address"); // 成都链安 // spender 非零地址检查

        _allowances[owner][spender] = amount; // 成都链安 // 设置 owner 对 spender 的授权值为 amount
        emit Approval(owner, spender, amount); // 成都链安 // 触发 Approval 事件
    }
}

contract ERC20Detailed is IERC20 {
    string private _name; // 成都链安 // 声明变量_name，存储代币名称
    string private _symbol; // 成都链安 // 声明变量_symbol，存储代币简称
    uint8 private _decimals; // 成都链安 // 声明变量_decimals，存储代币精度
    // 成都链安 // 构造函数，初始化代币基本信息
    constructor (string memory name, string memory symbol, uint8 decimals) public {
```

```
        _name = name; // 成都链安 // 初始化代币名称
        _symbol = symbol; // 成都链安 // 初始化代币简称
        _decimals = decimals; // 成都链安 // 初始化代币精度
    }
    // 成都链安 // 代币名称查询函数
    function name() public view returns (string memory) {
        return _name;
    }
    // 成都链安 // 代币简称查询函数
    function symbol() public view returns (string memory) {
        return _symbol;
    }
    // 成都链安 // 代币精度查询函数
    function decimals() public view returns (uint8) {
        return _decimals;
    }
}
// 成都链安 // SafeMath 库定义如下函数用于安全数学运算
library SafeMath {
    function add(uint a, uint b) internal pure returns (uint) {
        uint c = a + b;
        require(c >= a, "SafeMath: addition overflow");

        return c;
    }
    function sub(uint a, uint b) internal pure returns (uint) {
        return sub(a, b, "SafeMath: subtraction overflow");
    }
    function sub(uint a, uint b, string memory errorMessage) internal pure returns (uint) {
        require(b <= a, errorMessage);
        uint c = a - b;

        return c;
    }
    function mul(uint a, uint b) internal pure returns (uint) {
        if (a == 0) {
            return 0;
        }

        uint c = a * b;
        require(c / a == b, "SafeMath: multiplication overflow");

        return c;
    }
    function div(uint a, uint b) internal pure returns (uint) {
        return div(a, b, "SafeMath: division by zero");
    }
    function div(uint a, uint b, string memory errorMessage) internal pure returns (uint) {
```

```
        // Solidity only automatically asserts when dividing by 0
        require(b > 0, errorMessage);
        uint c = a / b;

        return c;
    }
}
// 成都链安 // Address 库定义 isContract 函数用于检查指定地址是否为合约地址
library Address {
    function isContract(address account) internal view returns (bool) {
        bytes32 codehash;
        bytes32 accountHash =
0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470;
        // solhint-disable-next-line no-inline-assembly
        assembly { codehash := extcodehash(account) }
        return (codehash != 0x0 && codehash != accountHash);
    }
}
// 成都链安 // SafeERC20 库，其内部函数用于安全外部 TRC20 合约转账相关操作
library SafeERC20 {
    using SafeMath for uint; // 成都链安 // 引入 SafeMath 安全数学运算库，避免数学运算整型溢
出
    using Address for address; // 成都链安 // 使用 Address 库中函数检查指定地址是否为合约地址

    function safeTransfer(IERC20 token, address to, uint value) internal {
        callOptionalReturn(token, abi.encodeWithSelector(token.transfer.selector, to,
value));
    }

    function safeTransferFrom(IERC20 token, address from, address to, uint value) internal {
        callOptionalReturn(token, abi.encodeWithSelector(token.transferFrom.selector, from,
to, value));
    }

    function safeApprove(IERC20 token, address spender, uint value) internal {
        require((value == 0) || (token.allowance(address(this), spender) == 0),
            "SafeERC20: approve from non-zero to non-zero allowance"
        );
        callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector, spender,
value));
    }
    function callOptionalReturn(IERC20 token, bytes memory data) private {
        require(address(token).isContract(), "SafeERC20: call to non-contract");

        // solhint-disable-next-line avoid-low-level-calls
        (bool success, bytes memory returndata) = address(token).call(data);
        require(success, "SafeERC20: low-level call failed");
```

```
        if (returndata.length > 0) { // Return data is optional
            // solhint-disable-next-line max-line-length
            require(abi.decode(returndata, (bool)), "SafeERC20: ERC20 operation did not
succeed");
        }
    }
}

contract MFI is ERC20, ERC20Detailed {
  using SafeERC20 for IERC20; // 成都链安 // 引入 SafeERC20 库，其内部函数用于安全外部 TRC20
合约转账相关操作
  using Address for address; // 成都链安 // 使用 Address 库中函数检查指定地址是否为合约地址
  using SafeMath for uint; // 成都链安 // 引入 SafeMath 安全数学运算库，避免数学运算整型溢出


  address public governance; // 成都链安 // 声明变量 governance，存储治理管理员地址
  mapping (address => bool) public minters; // 成都链安 // 声明 mapping 变量 minters，存储指定
地址的铸币权限
  // 成都链安 // 构造函数，初始化代币基本信息与治理管理员地址
  constructor () public ERC20Detailed("MoonPool.finance", "MFI", 18) {
      governance = msg.sender;
  }
  // 成都链安 // 铸币函数，拥有铸币权限地址向指定地址铸币
  function mint(address account, uint amount) public {
      require(minters[msg.sender], "!minter");
      _mint(account, amount);
  }
  // 成都链安 // 设置治理管理员地址
  function setGovernance(address _governance) public {
      require(msg.sender == governance, "!governance"); // 成都链安 // 要求调用者必须为当前
治理管理员地址
      governance = _governance; // 成都链安 // 更新 governance
  }
  // 成都链安 // 添加铸币权限函数
  function addMinter(address _minter) public {
      require(msg.sender == governance, "!governance"); // 成都链安 // 要求调用者必须为当前
治理管理员地址
      minters[_minter] = true; // 成都链安 // 变更指定地址_minter 的铸币权限为 true
  }
  // 成都链安 // 移除铸币权限函数
  function removeMinter(address _minter) public {
      require(msg.sender == governance, "!governance"); // 成都链安 // 要求调用者必须为当前
治理管理员地址
      minters[_minter] = false; // 成都链安 // 变更指定地址_minter 的铸币权限为 false
  }
}
```

成都链安
BEOSIN

官方网址

https://lianantech.com

电子邮箱

vaas@lianantech.com

微信公众号