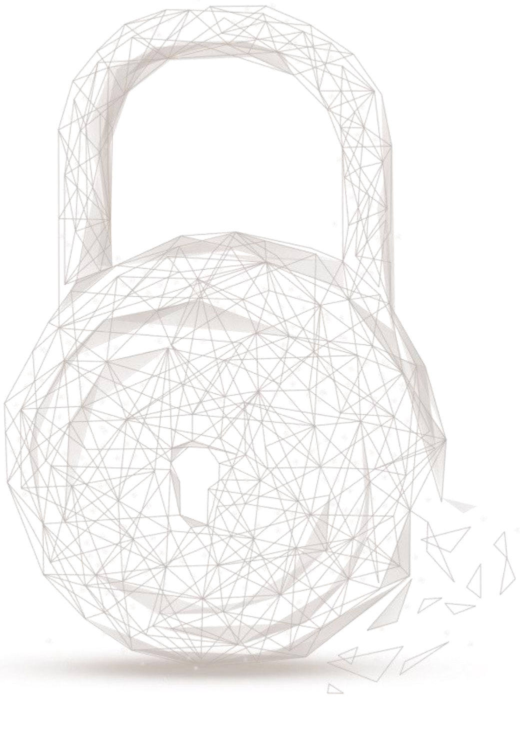
成都链安
BEOSIN

智能合约安全审计报告



审计编号：202106021319

审计合约名称：

MNST （MNST）

审计合约地址：

TM3tyxtGXUSzh91zEWszyHgC8aNKovAeNw

审计合约链接：

https://tronscan.org/#/contract/TM3tyxtGXUSzh91zEWszyHgC8aNKovAeNw/code

合约审计开始日期：2021.05.28

合约审计完成日期：2021.06.02

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
		能量消耗审计	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：本报告系针对项目代码而作出，本报告的任何描述、表达或措辞均不得被解释为对项目的认可、肯定或确认。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MNST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，MNST合约通过所有检测，合约审计结果为通过(优)。**以下为本合约基本信息。

1、代币基本信息

Token name	MNST
Token symbol	MNST
decimals	6
totalSupply	初始为0，可铸币，可销毁，无总量上限
Token type	TRC20

表1 代币基本信息

2、代币锁仓信息

无锁仓

合约源代码审计注释：

MSNT.sol

```
pragma solidity ^0.4.25;

import './TRC20.sol';
import './TRC20Detailed.sol';
import './TRC20Burnable.sol';
import './TRC20Mintable.sol';

/**
 * @title MNST Token Contract
 * @dev Responsible for receiving the token's details at deployment
   and creating the token with the TRC20 token standard. MNST token
   also has the burn feature
 * @author @wafflemakr
 */

contract MNST is TRC20Burnable, TRC20Mintable{


    /**
     * @notice Token Deployment
     * @param name Name of the token (MNST)
     * @param symbol Symbol of the token (MNST)
     * @param decimals Amount of decimals of the token (6)
     * @param supply Max supply of the token (27 Billion)
     * @param initialOwner Address of the person that will receive
        the total supply when deploying the token
     */
    // 成都链安 // 初始化代币名称，标识，精度，初始供应量和初始代币拥有者
    constructor
    (
        string name, string symbol,
        uint8 decimals, uint256 supply,
        address initialOwner
    )

        public TRC20Detailed(name, symbol, decimals)

    {
        mint(initialOwner, supply * (10 ** uint256(decimals)));
    }
}
// 成都链安 // 建议主合约继承 Pausable 模块，当出现重大异常时 owner 可以暂停所有交易
```

ITRC20.sol

```
pragma solidity ^0.4.25;

/**
 * @title TRC20 interface (compatible with ERC20 interface)
 */
// 成都链安 // 定义 TRC20 标准要求的接口函数与事件
interface ITRC20 {
    function totalSupply() external view returns (uint256);

    function balanceOf(address who) external view returns (uint256);

    function allowance(address owner, address spender)
    external view returns (uint256);

    function transfer(address to, uint256 value) external returns (bool);

    function approve(address spender, uint256 value)
    external returns (bool);

    function transferFrom(address from, address to, uint256 value)
    external returns (bool);

    event Transfer(
        address indexed from,
        address indexed to,
        uint256 value
    );

    event Approval(
        address indexed owner,
        address indexed spender,
        uint256 value
    );
}
```

MinterRole.sol

```
pragma solidity ^0.4.25;

import "./Roles.sol";

contract MinterRole {
    using Roles for Roles.Role; // 成都链安 // 引用 Roles 库，用于 minter 角色控制

    event MinterAdded(address indexed account);
    event MinterRemoved(address indexed account);

    Roles.Role private _minters;

    constructor () internal {
        _addMinter(msg.sender); // 成都链安 // 调用内部函数_addMinter 将合约创建者设置为
minter 角色
    }
    // 成都链安 // onlyMinter 修饰器，被该修饰器修饰的函数只能被 minter 角色调用
    modifier onlyMinter() {
        require(isMinter(msg.sender));
        _;
    }
    // 成都链安 // isMinter 函数用来查询指定地址是否为 minter 角色
    function isMinter(address account) public view returns (bool) {
        return _minters.has(account);
    }
    function addMinter(address account) public onlyMinter returns (bool) {
        _addMinter(account); // 成都链安 // 调用内部函数_addMinter 将指定账户设置为
```

minter 角色

```
        return true;
    }

    function renounceMinter() public returns (bool){
        _removeMinter(msg.sender); // 成都链安 // 调用内部函数_removeMinter 取消函数调用
者的 minter 权限
        return true;
    }

    function _addMinter(address account) internal {
        _minters.add(account); // 成都链安 // 调用内部函数 add 向指定账户添加 minter 角色
权限
        emit MinterAdded(account); // 成都链安 // 触发 MinterAdded 事件
    }

    function _removeMinter(address account) internal {
        _minters.remove(account); // 成都链安 // 调用内部函数 remove 移除指定账户的
minter 角色权限
        emit MinterRemoved(account); // 成都链安 // 触发 MinterRemoved 事件
    }
}
```

Roles.sol

```
pragma solidity ^0.4.25;

/**
 * @title Roles
 * @dev Library for managing addresses assigned to a Role.
 */
// 成都链安 // Roles 库用于角色权限控制
library Roles {
    struct Role {
        mapping (address => bool) bearer;
    }

    /**
     * @dev give an account access to this role
     */
    function add(Role storage role, address account) internal {
        require(account != address(0));
        require(!has(role, account));

        role.bearer[account] = true;
    }
    /**
     * @dev remove an account's access to this role
     */
    function remove(Role storage role, address account) internal {
        require(account != address(0));
        require(has(role, account));

        role.bearer[account] = false;
    }

    /**
     * @dev check if an account has this role
     * @return bool
     */
    function has(Role storage role, address account) internal view returns (bool) {
        require(account != address(0));
        return role.bearer[account];
    }
}
```

SafeMath.sol

```
pragma solidity ^0.4.25;

/**
 * @title SafeMath
 * @dev Math operations with safety checks that revert on error
 */
// 成都链安 // SafeMath 库用于安全数学运算以避免整型溢出
library SafeMath {

    /**
     * @dev Multiplies two numbers, reverts on overflow.
     */
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b);
        return c;
    }

    /**
     * @dev Integer division of two numbers truncating the quotient, reverts on division
by zero.
     */
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b > 0); // Solidity only automatically asserts 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;
    }

    /**
     * @dev Subtracts two numbers, reverts on overflow (i.e. if subtrahend is greater
than minuend).
     */
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b <= a);
        uint256 c = a - b;

        return c;
    }

    /**
     * @dev Adds two numbers, reverts on overflow.
     */
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a);

        return c;
    }

    /**
     * @dev Divides two numbers and returns the remainder (unsigned integer modulo),
     * reverts when dividing by zero.
     */
    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b != 0);
        return a % b;
    }
}
```

TRC20.sol

```
pragma solidity ^0.4.25;

import "./ITRC20.sol";
import "./SafeMath.sol";

/**
 * @title Standard TRC20 token (compatible with ERC20 token)
 *
 * @dev Implementation of the basic standard token.
 */
contract TRC20 is ITRC20 {
    using SafeMath for uint256; // 成都链安 // 引用 SafeMath 安全库，用于安全数学运算

    mapping (address => uint256) 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 => uint256)) private _allowed; // 成都链安 //
声明 mapping 变量_allowed，存储对应地址间的授权值

    uint256 private _totalSupply; // 成都链安 // 声明变量_totalSupply，存储代币总量

    /**
     * @dev Total number of tokens in existence
     */
    function totalSupply() public view returns (uint256) {
        return _totalSupply; // 成都链安 // 返回代币总量
    }

    /**
     * @dev Gets the balance of the specified address.
     * @param owner The address to query the balance of.
     * @return An uint256 representing the amount owned by the passed address.
     */
    function balanceOf(address owner) public view returns (uint256) {
        return _balances[owner]; // 成都链安 // 返回账户代币余额
    }

    /**
     * @dev Function to check the amount of tokens that 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 uint256 specifying the amount of tokens still available for the
spender.
     */
    function allowance(
        address owner,
        address spender
    )
    public
    view
    returns (uint256)
    {
        return _allowed[owner][spender]; // 成都链安 // 返回 owner 对 spender 的授权值
    }

    /**
     * @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, uint256 value) public returns (bool) {
        _transfer(msg.sender, to, value); // 成都链安 // 调用内部函数_transfer 进行代币
转账
        return true;
    }

    /**
     * @dev Approve the passed address to spend the specified amount of tokens on behalf
of msg.sender.
     * Beware that changing an allowance with this method brings the risk that someone
may use both the old
     * and the new allowance by unfortunate transaction ordering. One possible solution
to mitigate this
     * race condition is to first reduce the spender's allowance to 0 and set the
desired value afterwards:
     * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
     * @param spender The address which will spend the funds.
     * @param value The amount of tokens to be spent.
     */
    // 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, uint256 value) public returns (bool) {
        require(spender != address(0)); // 成都链安 // spender 非零地址检查
        _allowed[msg.sender][spender] = value; // 成都链安 // 更新函数调用者对 spender 的
授权值
        emit Approval(msg.sender, spender, value); // 成都链安 // 触发 Approval 事件
        return true;
    }

    /**
     * @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 uint256 the amount of tokens to be transferred
     */

    function transferFrom(
        address from,
        address to,
        uint256 value
    )
    public
    returns (bool)
    {
        _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value); // 成都链安
// 更新 from 对函数调用者的授权值
        _transfer(from, to, value); // 成都链安 // 调用内部函数_transfer 进行代币转账
        return true;
    }

    /**
     * @dev Increase the amount of tokens that an owner allowed to a spender.
     * approve should be called when allowed_[_spender] == 0. To increment
     * allowed value is better to use this function to avoid 2 calls (and wait until
     * the first transaction is mined)
     * From MonolithDAO Token.sol
     * @param spender The address which will spend the funds.
     * @param addedValue The amount of tokens to increase the allowance by.
     */

    function increaseAllowance(
        address spender, // 成都链安 // spender 非零地址检查
        uint256 addedValue
    )
    public
    returns (bool)
    {
        require(spender != address(0)); // 成都链安 // spender 非零地址检查
        _allowed[msg.sender][spender] = (
        _allowed[msg.sender][spender].add(addedValue)); // 成都链安 // 增加函数调用者对
spender 的授权值
        emit Approval(msg.sender, spender, _allowed[msg.sender][spender]); // 成都链安
// 触发 Approval 事件
        return true;
    }

    /**
     * @dev Decrease the amount of tokens that an owner allowed to a spender.
     * approve should be called when allowed_[_spender] == 0. To decrement
     * allowed value is better to use this function to avoid 2 calls (and wait until
     * the first transaction is mined)
     * From MonolithDAO Token.sol
     * @param spender The address which will spend the funds.
     * @param subtractedValue The amount of tokens to decrease the allowance by.
     */
    function decreaseAllowance(
        address spender,
        uint256 subtractedValue
    )
    public
    returns (bool)
    {
        require(spender != address(0)); // 成都链安 // spender 非零地址检查

        _allowed[msg.sender][spender] = (
        _allowed[msg.sender][spender].sub(subtractedValue)); // 成都链安 // 减少函数调用
者对 spender 的授权值
        emit Approval(msg.sender, spender, _allowed[msg.sender][spender]); // 成都链安
// 触发 Approval 事件
        return true;
    }

    /**
     * @dev Transfer token for a specified addresses
     * @param from The address to transfer from.
     * @param to The address to transfer to.
     * @param value The amount to be transferred.
     */
    function _transfer(address from, address to, uint256 value) internal {
        require(to != address(0)); // 成都链安 // to 非零地址检查

        _balances[from] = _balances[from].sub(value); // 成都链安 // 更新 from 地址代币余
额
        _balances[to] = _balances[to].add(value); // 成都链安 // 更新 to 地址代币余额
        emit Transfer(from, to, value); // 成都链安 // 触发 Transfer 事件
    }

    /**
     * @dev Internal function that mints an amount of the token and assigns it to
     * an account. This encapsulates the modification of balances such that the
     * proper events are emitted.
     * @param account The account that will receive the created tokens.
     * @param value The amount that will be created.
     */
    function _mint(address account, uint256 value) internal {
        require(account != address(0)); // 成都链安 // account 非零地址检查

        _totalSupply = _totalSupply.add(value); // 成都链安 // 更新代币总量
        _balances[account] = _balances[account].add(value); // 成都链安 // 更新 account
地址代币余额
```

```
        emit Transfer(address(0), account, value); // 成都链安 // 触发 Transfer 事件
    }

    /**
     * @dev Internal function that burns an amount of the token of a given
     * account.
     * @param account The account whose tokens will be burnt.
     * @param value The amount that will be burnt.
     */
    function _burn(address account, uint256 value) internal {
        require(account != address(0)); // 成都链安 // account 非零地址检查
        _totalSupply = _totalSupply.sub(value); // 成都链安 // 更新代币总量
        _balances[account] = _balances[account].sub(value); // 成都链安 // 更新 account
地址代币余额
        emit Transfer(account, address(0), value); // 成都链安 // 触发 Transfer 事件
    }

    /**
     * @dev Internal function that burns an amount of the token of a given
     * account, deducting from the sender's allowance for said account. Uses the
     * internal burn function.
     * @param account The account whose tokens will be burnt.
     * @param value The amount that will be burnt.
     */
    function _burnFrom(address account, uint256 value) internal {
        // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be
accepted,
        // this function needs to emit an event with the updated approval.
        _allowed[account][msg.sender] = _allowed[account][msg.sender].sub(
            value); // 成都链安 //增加 account 对函数调用者的授权值
        _burn(account, value); // 成都链安 // 调用内部函数_burn 销毁账户代币
    }
}
```

TRC20Burnable.sol

```
pragma solidity ^0.4.25;

import "./TRC20.sol";

/**
 * @title Burnable Token
 * @dev Token that can be irreversibly burned (destroyed).
 */
contract TRC20Burnable is TRC20 {
    /**
     * @dev Burns a specific amount of tokens.
     * @param value The amount of token to be burned.
     */
    function burn(uint256 value) public returns (bool){
        _burn(msg.sender, value); // 成都链安 // 调用内部函数_burn 销毁函数调用者指定数
量代币
        return true;
    }

    /**
     * @dev Burns a specific amount of tokens from the target address and decrements
allowance
     * @param from address The address which you want to send tokens from
     * @param value uint256 The amount of token to be burned
     */
    function burnFrom(address from, uint256 value) public returns (bool){
        _burnFrom(from, value); // 成都链安 // 调用内部函数_burnfrom 销毁 from 地址指定数
量的代币
        return true;
    }
}
```

TRC20Detailed.sol

```
pragma solidity ^0.4.25;

import "./TRC20.sol";

/**
 * @title TRC20Detailed token
 * @dev The decimals are only for visualization purposes.
 * All the operations are done using the smallest and indivisible token unit,
 * just as on TRON all the operations are done in sun.
 *
 * Example inherits from basic TRC20 implementation but can be modified to
 */
contract TRC20Detailed is TRC20 {
    string private _name; // 成都链安 // 声明变量_name，用于存储代币名称
    string private _symbol; // 成都链安 // 声明变量_symbol,用于存储代币标识
    uint8 private _decimals; // 成都链安 // 声明变量_decimals,用于存储代币精度

    constructor (string name, string symbol, uint8 decimals) public {
        _name = name; // 成都链安 // 初始化代币名称
        _symbol = symbol; // 成都链安 // 初始化代币标识
        _decimals = decimals; // 成都链安 // 初始化代币精度
    }

    /**
     * @return the name of the token.
     */
    function name() public view returns (string) {
        return _name;
    }

    /**
     * @return the symbol of the token.
     */
    function symbol() public view returns (string) {
        return _symbol;
    }

    /**
     * @return the number of decimals of the token.
     */
    function decimals() public view returns (uint8) {
        return _decimals;
    }
}
```

TRC20Mintable.sol

```
pragma solidity ^0.4.25;

import "./TRC20.sol";
import "./MinterRole.sol";
import "./TRC20Detailed.sol";

/**
 * @title TRC20Mintable
 * @dev TRC20 minting logic
 */
contract TRC20Mintable is MinterRole,TRC20Detailed  {



    /**
     * @dev Function to mint tokens
     * @param to The address that will receive the minted tokens.
     * @return A boolean that indicates if the operation was successful.
     */
    function mint(address to, uint256 value) public onlyMinter returns (bool) {

        _mint(to, value); // 成都链安 // 调用内部函数_mint 铸币并发送到指定地址
        return true;
    }
}
```

Beosin

成都链安

BEOSIN

官方网址

https://lianantech.com

电子邮箱

vaas@lianantech.com

微信公众号