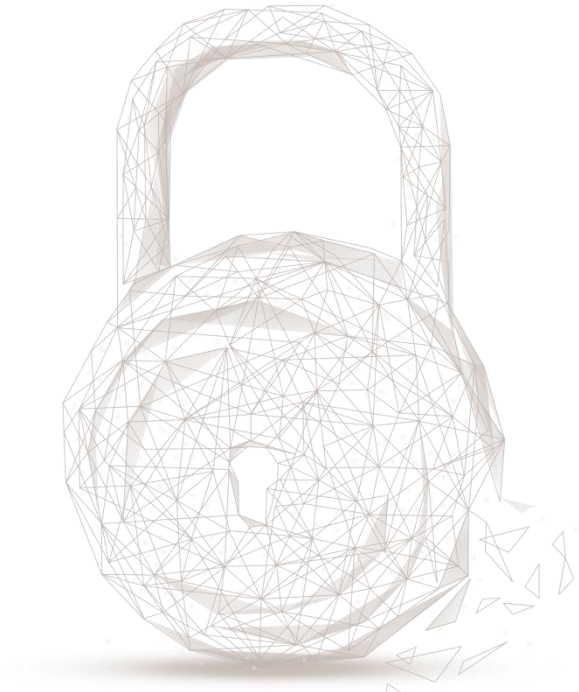
成都链安
BEOSIN

智能合约安全审计报告



审计编号：202101041732

审计合约名称：

FC

审计合约地址：

TKzf6cF5RsxzV7Shcf8MpuRRJcfDiG1hDn

审计合约链接地址：

https://tronscan.org/#/contract/TKzf6cF5RsxzV7Shcf8MpuRRJcfDiG1hDn/code

合约审计开始日期：2020.01.04

合约审计完成日期：2020.01.04

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	TRC-20 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。

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FC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，FC合约通过所有检测项，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	Filecoin Community Token
Token symbol	FC
decimals	6
totalSupply	总量恒定1亿
Token type	TRC-20

表1 代币基本信息

2、代币锁仓信息

无锁仓

3、其它函数功能描述

- transfer函数

如下图所示，合约修改了OpenZeppelin中ERC20的transfer方法，实现了转账扣除10%手续费的功能。

```
231     function transfer(address recipient, uint256 amount) public returns (bool) {
232         require(amount >= 0, "Cannot transfer lower 0");
233         uint256 feeValue = SafeMath.div(amount, 10);
234         uint256 taxedValue = SafeMath.sub(amount, feeValue);
235         _transfer(msg.sender, recipient, taxedValue);
236         _transfer(msg.sender, feeAddr, feeValue);
237         return true;
238     }
```

图 1 transfer函数源码截图

- internalTransfer函数

如下图所示，合约实现了internalTransfer函数，任何人可以调用此函数进行无需手续费的转账。

```
450     function internalTransfer(address recipient, uint256 amount)
451         public
452         returns (bool)
453     {
454         _transfer(msg.sender, recipient, amount);
455         return true;
456     }
```

图 2 internalTransfer函数源码截图

- setFeeAddr函数

如下图所示，合约实现了setFeeAddr函数用于修改手续费接收地址，仅合约的Owner权限可以进行调用。

```
461     function setFeeAddr(address _feeAddr) external onlyOwner returns (address) {
462         require(
463             _feeAddr != address(0),
464             "ERC20: _feeAddr set to the zero address"
465         );
466         feeAddr = _feeAddr;
467         return feeAddr;
468     }
```

图 3 setFeeAddr函数源码截图

合约源代码审计注释：

```
// 0.5.1-c8a2
// Enable optimization
pragma solidity ^0.5.0; // 成都链安 // 建议固定编译器版本


// 成都链安 // 防溢出的安全数学运算库
library SafeMath {
  /**
```

```
 * @dev Returns the addition of two unsigned integers, reverting on
 * overflow.
 *
 * Counterpart to Solidity's `+` operator.
 *
 * Requirements:
 * - Addition cannot overflow.
 */
function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    require(c >= a, "SafeMath: addition overflow");

    return c;
}

/**
 * @dev Returns the subtraction of two unsigned integers, reverting on
 * overflow (when the result is negative).
 *
 * Counterpart to Solidity's `-` operator.
 *
 * Requirements:
 * - Subtraction cannot overflow.
 */
function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    require(b <= a, "SafeMath: subtraction overflow");
    uint256 c = a - b;

    return c;
}

/**
 * @dev Returns the multiplication of two unsigned integers, reverting on
 * overflow.
 *
 * Counterpart to Solidity's `*` operator.
 *
 * Requirements:
```

```
 * - Multiplication cannot 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-contracts/pull/522
    if (a == 0) {
        return 0;
    }

    uint256 c = a * b;
    require(c / a == b, "SafeMath: multiplication overflow");

    return c;
}

/**
 * @dev Returns the integer division of two unsigned integers. Reverts on
 * division by zero. The result is rounded towards zero.
 *
 * Counterpart to Solidity's `/` operator. Note: this function uses a
 * `revert` opcode (which leaves remaining gas untouched) while Solidity
 * uses an invalid opcode to revert (consuming all remaining gas).
 *
 * Requirements:
 * - The divisor cannot be zero.
 */
function div(uint256 a, uint256 b) internal pure returns (uint256) {
    // Solidity only automatically asserts when dividing by 0
    require(b > 0, "SafeMath: division by zero");
    uint256 c = a / b;
    // assert(a == b * c + a % b); // There is no case in which this doesn't hold

    return c;
}

/**
 * @dev Returns the remainder of dividing two unsigned integers. (unsigned integer modulo),
```

```
     * Reverts when dividing by zero.
     *
     * Counterpart to Solidity's `%` operator. This function uses a `revert`
     * opcode (which leaves remaining gas untouched) while Solidity uses an
     * invalid opcode to revert (consuming all remaining gas).
     *
     * Requirements:
     * - The divisor cannot be zero.
     */
    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b != 0, "SafeMath: modulo by zero");
        return a % b;
    }
}
// 成都链安 // 定义 TRC-20 Token 标准要求的接口函数
interface IERC20 {
    /**
     * @dev Returns the amount of tokens in existence.
     */
    function totalSupply() external view returns (uint256);

    /**
     * @dev Returns the amount of tokens owned by `account`.
     */
    function balanceOf(address account) external view returns (uint256);

    /**
     * @dev Moves `amount` tokens from the caller's account to `recipient`.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * Emits a {Transfer} event.
     */
    function transfer(address recipient, uint256 amount) external returns (bool);

    /**
     * @dev Returns the remaining number of tokens that `spender` will be
     * allowed to spend on behalf of `owner` through {transferFrom}. This is
```

```
     * zero by default.
     *
     * This value changes when {approve} or {transferFrom} are called.
     */
    function allowance(address owner, address spender) external view returns (uint256);

    /**
     * @dev Sets `amount` as the allowance of `spender` over the caller's tokens.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * IMPORTANT: 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
     *
     * Emits an {Approval} event.
     */
    function approve(address spender, uint256 amount) external returns (bool);

    /**
     * @dev Moves `amount` tokens from `sender` to `recipient` using the
     * allowance mechanism. `amount` is then deducted from the caller's
     * allowance.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * Emits a {Transfer} event.
     */
    function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);

    /**
     * @dev Emitted when `value` tokens are moved from one account (`from`) to
     * another (`to`).
     *
     * Note that `value` may be zero.
```

```
    */
    event Transfer(address indexed from, address indexed to, uint256 value); // 成都链安 // 声明代币转账事件

    /**
     * @dev Emitted when the allowance of a `spender` for an `owner` is set by
     * a call to {approve}. `value` is the new allowance.
     */
    event Approval(address indexed owner, address indexed spender, uint256 value); // 成都链安 // 声明代币授权事件
}
contract Ownable {
    address owner; // 成都链安 // 声明变量 owner，用于存储合约所有者地址
    address newOwner; // 成都链安 // 声明变量 newOwner，用于存储合约新的所有者地址


    constructor() internal {
        owner = msg.sender; // 成都链安 // 构造函数，设置合约创建者为 owner
    }

    // 成都链安 // 修饰器，检查调用者为 owner
    modifier onlyOwner() {
        require(msg.sender == owner);
        _;
    }
    // 成都链安 // 修改 newOwner 地址变量的值，仅 owner 可以调用
    function ChangeOwnership(address p_newOwner) external onlyOwner {
        newOwner = p_newOwner;
    }
    // 成都链安 // newOwner 就收 owner 权限，仅 newOwner 可以调用
    function AcceptOwnership() external {
        require(msg.sender == newOwner);
        owner = newOwner;
    }
    // 成都链安 // 返回当前 owner 地址
    function ownner() public view returns (address) {
        return owner;
```

```
    }
}
contract ERC20 is IERC20, Ownable {
    using SafeMath for uint256; // 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 feeAddr; // 成都链安 // 声明 address 变量 feeAddr，存储接收手续费的地址

    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 _allowances; // 成都链安 // 声明 mapping 变量_allowances，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

    constructor() internal {
        feeAddr = msg.sender; // 成都链安 // 构造函数，将部署地址赋值给 feeAddr 地址变量
    }

    /**
     * @dev See {IERC20-totalSupply}.
     */
    function totalSupply() public view returns (uint256) {
        return _totalSupply;
    }

    /**
     * @dev See {IERC20-balanceOf}.
     */
    function balanceOf(address account) public view returns (uint256) {
        return _balances[account];
    }

    /**
     * @dev See {IERC20-transfer}.
     *
     * Requirements:
```

```
     *
     * - `recipient` cannot be the zero address.
     * - the caller must have a balance of at least `amount`.
     */
    function transfer(address recipient, uint256 amount) public returns (bool) {
        require(amount >= 0, "Cannot transfer lower 0"); // 成都链安 // 检查转账金额大于等于 0
        uint256 feeValue = SafeMath.div(amount, 10);// 成都链安 // 计算出转账手续费
        uint256 taxedValue = SafeMath.sub(amount, feeValue);// 成都链安 // 计算出实际转账金额
        _transfer(msg.sender, recipient, taxedValue);// 成都链安 // 调用内部函数_transfer 进行代币转账
        _transfer(msg.sender, feeAddr, feeValue);// 成都链安 // 调用内部函数_transfer 进行手续费转账
        return true;
    }

    /**
     * @dev See {IERC20-allowance}.
     */
    function allowance(address owner, address spender)
        public
        view
        returns (uint256)
    {
        return _allowances[owner][spender];
    }

    /**
     * @dev See {IERC20-approve}.
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     */
    // 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
    // 成都链安 // 建议用户使用 increaseAllowance 与 decreaseAllowance 修改授权值
    function approve(address spender, uint256 value) public returns (bool) {
        _approve(msg.sender, spender, value); // 成都链安 // 调用内部函数_approve 设置调用者对 spender 的授权值
        return true;
```

```
    }

    /**
     * @dev See {IERC20-transferFrom}.
     *
     * Emits an {Approval} event indicating the updated allowance. This is not
     * required by the EIP. See the note at the beginning of {ERC20};
     *
     * Requirements:
     * - `sender` and `recipient` cannot be the zero address.
     * - `sender` must have a balance of at least `value`.
     * - the caller must have allowance for `sender`'s tokens of at least
     * `amount`.
     */
    function transferFrom(
        address sender,
        address recipient,
        uint256 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,
            msg.sender,
            _allowances[sender][msg.sender].sub(amount) // 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;
    }

    /**
     * @dev Atomically increases the allowance granted to `spender` by the caller.
     *
     * This is an alternative to {approve} that can be used as a mitigation for
     * problems described in {IERC20-approve}.
     *
     * Emits an {Approval} event indicating the updated allowance.
     *
```

```
 * Requirements:
 *
 * - `spender` cannot be the zero address.
 */
function increaseAllowance(address spender, uint256 addedValue)
    public
    returns (bool)
{
    _approve(
        msg.sender,
        spender,
        _allowances[msg.sender][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;
}

/**
 * @dev Atomically decreases the allowance granted to `spender` by the caller.
 *
 * This is an alternative to {approve} that can be used as a mitigation for
 * problems described in {IERC20-approve}.
 *
 * Emits an {Approval} event indicating the updated allowance.
 *
 * Requirements:
 *
 * - `spender` cannot be the zero address.
 * - `spender` must have allowance for the caller of at least
 * `subtractedValue`.
 */
function decreaseAllowance(address spender, uint256 subtractedValue)
    public
    returns (bool)
{
    _approve(
        msg.sender,
        spender,
        _allowances[msg.sender][spender].sub(subtractedValue)
```

```
        ); // 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;
    }

    /**
     * @dev Moves tokens `amount` from `sender` to `recipient`.
     *
     * This is internal function is equivalent to {transfer}, and can be used to
     * e.g. implement automatic token fees, slashing mechanisms, etc.
     *
     * Emits a {Transfer} event.
     *
     * Requirements:
     *
     * - `sender` cannot be the zero address.
     * - `recipient` cannot be the zero address.
     * - `sender` must have a balance of at least `amount`.
     */
    function _transfer(
        address sender,
        address recipient,
        uint256 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

        _balances[sender] = _balances[sender].sub(amount); // 成都链安 // 修改转账双方地址的代币余额
        _balances[recipient] = _balances[recipient].add(amount); // 成都链安 // 触发 Transfer 事件
        emit Transfer(sender, recipient, amount);
    }

    /** @dev Creates `amount` tokens and assigns them to `account`, increasing
     * the total supply.
     *
     * Emits a {Transfer} event with `from` set to the zero address.
     *
```

```
 * Requirements
 *
 * - `to` cannot be the zero address.
 */
function _mint(address account, uint256 amount) internal {
    require(account != address(0), "ERC20: mint to the zero address"); // 成都链安 // account 非零地址检查

    _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 account 地址的代币余额
    emit Transfer(address(0), account, amount); // 成都链安 // 触发 Transfer 事件
}

/**
 * @dev Destroys `amount` tokens from `account`, reducing the
 * total supply.
 *
 * Emits a {Transfer} event with `to` set to the zero address.
 *
 * Requirements
 *
 * - `account` cannot be the zero address.
 * - `account` must have at least `amount` tokens.
 */
// 成都链安 // 冗余代码，建议删除
function _burn(address account, uint256 value) internal {
    require(account != address(0), "ERC20: burn from the zero address");

    _totalSupply = _totalSupply.sub(value);
    _balances[account] = _balances[account].sub(value);
    emit Transfer(account, address(0), value);
}

/**
 * @dev Sets `amount` as the allowance of `spender` over the `owner`s tokens.
 *
 * This is internal function is equivalent to `approve`, and can be used to
 * e.g. set automatic allowances for certain subsystems, etc.
```

```
     *
     * Emits an {Approval} event.
     *
     * Requirements:
     *
     * - `owner` cannot be the zero address.
     * - `spender` cannot be the zero address.
     */
    function _approve(
        address owner,
        address spender,
        uint256 value
    ) 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] = value; // 成都链安 // 设置 owner 对 spender 的授权值为 amount
        emit Approval(owner, spender, value); // 成都链安 // 触发 Approval 事件
    }

    /**
     * @dev Destoys `amount` tokens from `account`.`amount` is then deducted
     * from the caller's allowance.
     *
     * See {_burn} and {_approve}.
     */
    // 成都链安 // 冗余代码，建议删除
    function _burnFrom(address account, uint256 amount) internal {
        _burn(account, amount);
        _approve(
            account,
            msg.sender,
            _allowances[account][msg.sender].sub(amount)
        );
    }

    /**
```

```
    * @dev See {IERC20-transfer}.
    *
    * Requirements:
    *
    * - `recipient` cannot be the zero address.
    * - the caller must have a balance of at least `amount`.
    */
    function internalTransfer(address recipient, uint256 amount)
        public
        returns (bool)
    {
        _transfer(msg.sender, recipient, amount); // 成都链安 // 调用内部函数_transfer 进行代币转账
        return true;
    }

    /**
    * set fee
    */
    function setFeeAddr(address _feeAddr) external onlyOwner returns (address) {
        require(
            _feeAddr != address(0),
            "ERC20: _feeAddr set to the zero address"
        );
        feeAddr = _feeAddr; // 成都链安 // 修改地址变量 feeAddr 的值，仅 owner 可以修改
        return feeAddr;
    }
}
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

    /**
    * @dev Sets the values for `name`, `symbol`, and `decimals`. All three of
    * these values are immutable: they can only be set once during
    * construction.
```

```
 */
// 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;
    _symbol = symbol;
    _decimals = decimals;
}

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

/**
 * @dev Returns the symbol of the token, usually a shorter version of the
 * name.
 */
function symbol() public view returns (string memory) {
    return _symbol;
}

/**
 * @dev Returns the number of decimals used to get its user representation.
 * For example, if `decimals` equals `2`, a balance of `505` tokens should
 * be displayed to a user as `5,05` (`505 / 10 ** 2`).
 *
 * Tokens usually opt for a value of 18, imitating the relationship between
 * Ether and Wei.
 *
 * NOTE: This information is only used for _display_ purposes: it in
 * no way affects any of the arithmetic of the contract, including
 * {IERC20-balanceOf} and {IERC20-transfer}.
 */
function decimals() public view returns (uint8) {
    return _decimals;
}
```

```
}
contract FC is ERC20, ERC20Detailed {
    /**
     * @dev Constructor that gives msg.sender all of existing tokens.
     */
    constructor() public ERC20Detailed("FC", "FC", 6) {
        _mint(msg.sender, 100000000 * (10**uint256(decimals()))); // 成都链安 // 构造函数，向创建合约的地址铸币 1 亿枚
    }
}
// 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
```

成都链安
BEOSIN

官方网址

https://lianantech.com

电子邮箱

vaas@lianantech.com

微信公众号