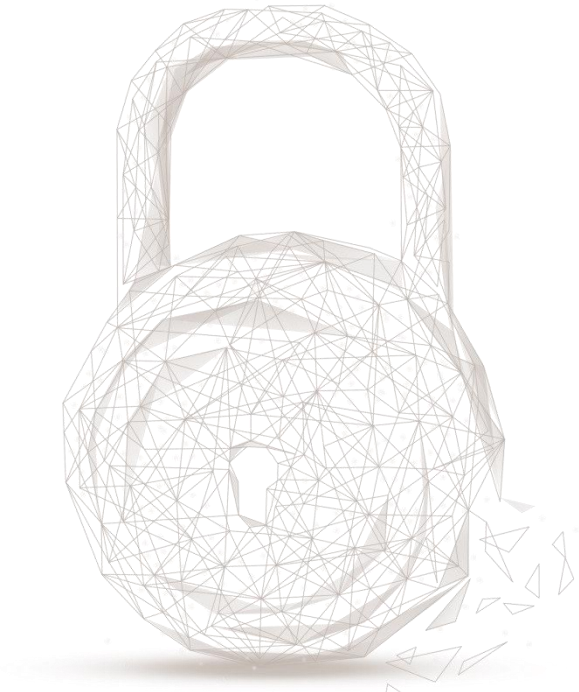
成都链安
BEOSIN

智能合约安全审计报告



审计编号：202009101736

报告查询名称：MFI

审计合约信息：

合约名称	审计合约地址	审计合约链接地址
MoonPoolCRTReward	TMKYuA8JuJJwrqtrwcayRHADXTYDZNveFn	https://tronscan.org/#/contract/TMKYuA8JuJJwrqtrwcayRHADXTYDZNveFn/code
MoonPoolJSTReward	TVuvY19L6BjiST9bNQrYAasNE6wJCXocnq	https://tronscan.org/#/contract/TVuvY19L6BjiST9bNQrYAasNE6wJCXocnq/code
MoonPoolPearlReward	TUrkZvNhNb9vtT7cD9rchbvFD3CYuyW2n4	https://tronscan.org/#/contract/TUrkZvNhNb9vtT7cD9rchbvFD3CYuyW2n4/code
MoonPoolTAIReward	TQJCpKfhjRXmXLSdNVQbNaVrJcoNTE1CcH	https://tronscan.org/#/contract/TQJCpKfhjRXmXLSdNVQbNaVrJcoNTE1CcH/code
MoonPoolUSDJReward	TWgbZoeDrSPcPF2B6ye291iLenhimpwaZj	https://tronscan.org/#/contract/TWgbZoeDrSPcPF2B6ye291iLenhimpwaZj/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

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	编译器版本安全审计	通过
		弃用项审计	通过
		冗余代码审计	通过
		require/assert 使用审计	通过
		gas 消耗审计	通过
2	通用漏洞审计	整型溢出审计	通过
		重入攻击审计	通过
		伪随机数生成审计	通过
		交易顺序依赖审计	通过

		拒绝服务攻击审计	通过
		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
		tx.origin 使用安全审计	通过
		重放攻击审计	通过
		变量覆盖审计	通过
3	业务审计	业务逻辑审计	通过
		业务实现审计	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MFI流动性挖矿项目智能合约代码规范性、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流动性挖矿项目智能合约通过所有检测项，合约审计结果为通过。**以下为本合约详细审计信息。

代码规范审计

1. 编译器版本安全审计

老版本的编译器可能会导致各种已知的安全问题，建议开发者在代码中指定合约代码采用最新的编译器版本，并消除编译器告警。

- **安全建议：无**
- **审计结果：通过**

2. 弃用项审计

Solidity智能合约开发语言处于快速迭代中，部分关键字已被新版本的编译器弃用，如throw、years等，为了消除其可能导致的隐患，合约开发者不应该使用当前编译器版本已弃用的关键字。

- **安全建议：无**
- **审计结果：通过**

3. 冗余代码审计

智能合约中的冗余代码会降低代码可读性，并可能需要消耗更多的gas用于合约部署，建议消除冗余代码。

- **安全建议：无**
- **审计结果：通过**

4. require/assert 使用审计

Solidity使用状态恢复异常来处理错误。这种机制将会撤消对当前调用(及其所有子调用)中的状态所做的所有更改，并向调用者标记错误。函数assert和require可用于检查条件并在条件不满足时抛出异常。assert函数只能用于测试内部错误，并检查非变量。require函数用于确认条件有效性，例如输入变量，或合约状态变量是否满足条件，或验证外部合约调用的返回值。

- **安全建议：无**
- **审计结果：通过**

5. gas 消耗审计

波场虚拟机执行合约代码需要消耗gas，当gas不足时，代码执行会抛出out of gas异常，并撤销所有状态变更。合约开发者需要控制代码的gas消耗，避免因为gas不足导致函数执行一直失败。

- **安全建议：无**
- **审计结果：通过**

通用漏洞审计

1. 整型溢出审计

整型溢出是很多语言都存在的安全问题，它们在智能合约中尤其危险。Solidity最多能处理256位的数字(2**256-1)，最大数字增加1会溢出得到0。同样，当数字为uint类型时，0减去1会下溢得到最大数字值。溢出情况会导致不正确的结果，特别是如果其可能的结果未被预期，可能会影响程序的可靠性和安全性。

- **安全建议：无**
- **审计结果：通过**

2. 重入攻击审计

重入漏洞是最典型的波场智能合约漏洞，该漏洞原因是Solidity中的call.value()函数在被用来发送TRX的时候会消耗它接收到的所有gas，当调用call.value()函数发送TRX的逻辑顺序存在错误时，就会存在重入攻击的风险。

- **安全建议：无**
- **审计结果：通过**

3. 伪随机数生成审计

智能合约中可能会使用到随机数，在solidity下常见的是用block区块信息作为随机因子生成，但是这样使用是不安全的，区块信息是可以被矿工控制或被攻击者在交易时获取到，这类随机数在一定程度上是可预测或可碰撞的，比较典型的例子就是fomo3d的airdrop随机数可以被碰撞。

- **安全建议：无**
- **审计结果：通过**

4. 交易顺序依赖审计

在波场的交易打包执行过程中，面对相同难度的交易时，矿工往往会选择gas费用高的优先打包，因此用户可以指定更高的gas费用，使自己的交易优先被打包执行。

- **安全建议：无**
- **审计结果：通过**

5. 拒绝服务攻击审计

拒绝服务攻击，即Denial of Service，可以使目标无法提供正常的服务。在波场智能合约中也会存在此类问题，由于智能合约的不可更改性，该类攻击可能使得合约永远无法恢复正常工作状态。导致智能合约拒绝服务的原因有很多种，包括在作为交易接收方时的恶意revert、代码设计缺陷导致gas耗尽等等。

- **安全建议：无**
- **审计结果：通过**

6. 函数调用权限审计

智能合约如果存在高权限功能，如：铸币、自毁、change owner等，需要对函数调用做权限限制，避免权限泄露导致的安全问题。

- **安全建议：无**
- **审计结果：通过**

7. call/delegatecall安全审计

Solidity中提供了call/delegatecall函数来进行函数调用，如果使用不当，会造成call注入漏洞，例如call的参数如果可控，则可以控制本合约进行越权操作或调用其他合约的危险函数。

- **安全建议：无**
- **审计结果：通过**

8. 返回值安全审计

在Solidity中存在transfer()、send()、call.value()等方法中，transfer转账失败交易会回滚，而send和call.value转账失败会return false，如果未对返回做正确判断，则可能会执行到未预期的逻辑；另外在ERC20 Token的transfer/transferFrom功能实现中，也要避免转账失败return false的情况，以免造成假充值漏洞。

- **安全建议：无**
- **审计结果：通过**

9. tx.origin使用安全审计

在波场智能合约的复杂调用中，tx.origin表示交易的初始创建者地址，如果使用tx.origin进行权限判断，可能会出现错误；另外，如果合约需要判断调用方是否为合约地址时则需要使用tx.origin，不能使用extcodesize。

- **安全建议：无**
- **审计结果：通过**

10. 重放攻击审计

重放攻击是指如果两份合约使用了相同的代码实现，并且身份鉴权在传参中，当用户在向一份合约中执行一笔交易，交易信息可以被复制并且向另一份合约重放执行该笔交易。

- **安全建议：无**
- **审计结果：通过**

11. 变量覆盖审计

波场存在着复杂的变量类型，例如结构体、动态数组等，如果使用不当，对其赋值后，可能导致覆盖已有状态变量的值，造成合约执行逻辑异常。

- **安全建议：无**
- **审计结果：通过**

业务审计

1. 抵押初始化

- **业务描述：**合约的“抵押-奖励”模式需要初始化相关参数（奖励比例rewardRate、首次更新时间lastUpdateTime、阶段完成时间periodFinish），通过指定的奖励分配管理员地址

rewardDistribution调用notifyRewardAmount函数，输入初始用于计算奖励比例的奖励数值reward，初始化抵押与奖励相关参数。

- **相关函数：**notifyRewardAmount、rewardPerToken、lastTimeRewardApplicable
- **安全建议：**无
- **审计结果：**通过

2. **抵押代币**

- **业务描述：**合约实现了stake函数用于抵押y代币，用户预先授权该合约地址，通过调用y合约中的transferFrom函数，合约地址代理用户将指定数量的y代币转至本合约地址；该函数限制用户仅可在“抵押-奖励”模式开启（到达指定时间）后进行调用；每次调用该函数抵押代币时通过修饰器updateReward更新奖励相关数据；以及每次调用通过修饰器checkhalve检查是否到达阶段完成时间，并进行奖励减半操作和奖励比例与阶段完成时间的更新。
- **相关函数：**stake、rewardPerToken、lastTimeRewardApplicable、earned、balanceOf
- **安全建议：**无
- **审计结果：**通过

3. **提取抵押代币**

- **业务描述：**合约实现了withdraw函数用于提取已抵押的y代币，通过调用y合约中的transfer函数，合约地址将指定数量的y代币转至函数调用者（用户）地址；该函数限制用户仅可在“抵押-奖励”模式开启（到达指定时间）后进行调用；每次调用该函数抵押代币时通过修饰器updateReward更新奖励相关数据；以及每次调用通过修饰器checkhalve检查是否到达阶段完成时间，并进行奖励减半操作和奖励比例与阶段完成时间的更新。
- **相关函数：**withdraw、rewardPerToken、lastTimeRewardApplicable、earned、balanceOf
- **安全建议：**无
- **审计结果：**通过

4. **领取抵押奖励**

- **业务描述：**合约实现了getReward函数用于领取抵押奖励（MFI代币），通过调用MFI合约中的transfer函数，合约地址将指定数量（用户的全部抵押奖励）的MFI代币转至函数调用者（用户）地址；该函数在用户提取奖励时，向治理管理员地址governance铸币本次提取奖励数量的10%；该函数限制用户仅可在“抵押-奖励”模式开启（到达指定时间）后进行调用；每次调用该函数抵押代币时通过修饰器updateReward更新奖励相关数据；以及每次调用通过修饰器checkhalve检查是否到达阶段完成时间，并进行奖励减半操作和奖励比例与阶段完成时间的更新。
- **相关函数：**getReward、rewardPerToken、lastTimeRewardApplicable、earned、balanceOf

- **安全建议：**无
- **审计结果：**通过

5. **退出抵押奖励参与**

- **业务描述：**合约实现了exit函数用于调用者退出抵押奖励参与，调用withdraw函数提取全部已抵押的y代币，调用getReward函数领取完调用者的抵押奖励，结束“抵押-奖励”模式参与，用户地址由于其已抵押y代币数量为空，无法获得新的抵押奖励。
- **相关函数：**exit、withdraw、getReward、rewardPerToken、lastTimeRewardApplicable、earned、balanceOf
- **安全建议：**无
- **审计结果：**通过

6. **奖励相关数据查询功能**

- **业务描述：**合约用户可通过调用lastTimeRewardApplicable函数查询当前时间戳与阶段完成时间中最早的时间戳；调用rewardPerToken函数可查询每个抵押代币可获得的抵押奖励；调用earned函数可查询指定地址所获取的总抵押奖励。
- **相关函数：**lastTimeRewardApplicable、rewardPerToken、earned
- **安全建议：**无
- **审计结果：**通过

合约源代码审计注释：

在本项目中依据同一代码架构实现了5个流动性挖矿奖池智能合约，分别为MoonPoolCRTReward、MoonPoolJSTReward、MoonPoolPearlReward、MoonPoolTAIReward和MoonPoolUSDJReward，这五个智能合约实现代码在同一位置存在差异，具体差异部分在表格1中列出，以下合约源代码审计注释为MoonPoolCRTReward合约内容。

合约名称	合约实现代码差异	备注
MoonPoolCRTReward	**// 成都链安 // 596 行合约名称与 600 行抵押流动性 y 代币合约地址差异** `contract LPTokenCRTWrapper {` `using SafeMath for uint256;` `using SafeERC20 for IERC20;` `IERC20 public y =`	抵押流动性 y 代币合约地址：0x41fdcfd438d2f94fa3acb0891f18128e27032ddc87 初始用于计算奖励比例的 initreward 值为 750*1e18

	IERC20(0x41fdcfd438d2f94fa3acb0891f18128e27032ddc87); // Stake Token address **// 成都链安 // 626 行合约名称与 630 行初始代币奖励 initreward 存在差异** contract MoonPoolCRTReward is LPTokenCRTWrapper, IRewardDistributionRecipient { IERC20 public mfi = IERC20(0x4177fd35cfd3349cb1ca14b9e71b5b7fd373331878); // MFI Token address uint256 public constant DURATION = 2 days; uint256 public initreward = 750*1e18;	
MoonPoolJSTReward	**// 成都链安 // 596 行合约名称与 600 行抵押流动性 y 代币合约地址差异** contract LPTokenJSTWrapper { using SafeMath for uint256; using SafeERC20 for IERC20; IERC20 public y = IERC20(0x4118fd0626daf3af02389aef3ed87db9c33f638ffa); // Stake Token address **// 成都链安 // 626 行合约名称与 630 行初始代币奖励 initreward 存在差异** contract MoonPoolJSTReward is LPTokenJSTWrapper, IRewardDistributionRecipient { IERC20 public mfi = IERC20(0x4177fd35cfd3349cb1ca14b9e71b5b7fd373331878); // MFI Token address uint256 public constant DURATION = 2 days; uint256 public initreward = 250*1e18;	抵押流动性 y 代币合约地址：0x4118fd0626daf3af02389aef3ed87db9c33f638ffa 初始用于计算奖励比例的 initreward 值为 250*1e18

MoonPoolPearlReward	**// 成都链安 // 596 行合约名称与 600 行抵押流动性 y 代币合约地址差异** contract LPTokenPearlWrapper { using SafeMath for uint256; using SafeERC20 for IERC20; IERC20 public y = IERC20(0x4148c125e0d3c626842bf7180c85e79f97ae524e91); // Stake Token address **// 成都链安 // 626 行合约名称与 630 行初始代币奖励 initreward 存在差异** contract MoonPoolPearlReward is LPTokenPearlWrapper, IRewardDistributionRecipient { IERC20 public mfi = IERC20(0x4177fd35cfd3349cb1ca14b9e71b5b7fd373331878); // MFI Token address uint256 public constant DURATION = 2 days; uint256 public initreward = 750*1e18;	抵押流动性 y 代币合约地址：0x4148c125e0d3c626842bf7180c85e79f97ae524e91 初始用于计算奖励比例的 initreward 值为 750*1e18
MoonPoolTAIReward	**// 成都链安 // 596 行合约名称与 600 行抵押流动性 y 代币合约地址差异** contract LPTokenTAIWrapper { using SafeMath for uint256; using SafeERC20 for IERC20; IERC20 public y = IERC20(0x41af2c205a7e44f79f680d149d339b733f6d34b6d5); // Stake Token address **// 成都链安 // 626 行合约名称与 630 行初始代币奖励 initreward 存在差异**	抵押流动性 y 代币合约地址：0x41af2c205a7e44f79f680d149d339b733f6d34b6d5 初始用于计算奖励比例的 initreward 值为 750*1e18

	contract MoonPoolTAIReward is LPTokenTAIWrapper, IRewardDistributionRecipient { IERC20 public mfi = IERC20(0x4177fd35cfd3349cb1ca14b9e71b5b7fd373331878); // MFI Token address uint256 public constant DURATION = 2 days; uint256 public initreward = 750*1e18;	
MoonPoolUSDJReward	**// 成都链安 // 596 行合约名称与 600 行抵押流动性 y 代币合约地址差异** contract LPTokenUSDJWrapper { using SafeMath for uint256; using SafeERC20 for IERC20; IERC20 public y = IERC20(0x41834295921a488d9d42b4b3021ed1a3c39fb0f03e); // Stake Token address **// 成都链安 // 626 行合约名称与 630 行初始代币奖励 initreward 存在差异** contract MoonPoolUSDJReward is LPTokenUSDJWrapper, IRewardDistributionRecipient { IERC20 public mfi = IERC20(0x4177fd35cfd3349cb1ca14b9e71b5b7fd373331878); // MFI Token address uint256 public constant DURATION = 2 days; uint256 public initreward = 250*1e18;	抵押流动性 y 代币合约地址：0x41834295921a488d9d42b4b3021ed1a3c39fb0f03e 初始用于计算奖励比例的 initreward 值为 250*1e18

表格 1 智能合约代码差异标注

```
/*
   ____            __   __        __   _
  / __/__ __ ___  / /_ / /  ___  / /_ (_)__ __
 _\ \ / // // _ \/ __// _ \/ -_)/ __// / \ \ /
/___/ \_, //_//_/\__//_//_/\__/ \__//_/ /_\_\
     /___/

* Synthetix: MFIReward.sol
*
* https://moonpool.finance/
*
*
* MIT License
* ===========
*
* Copyright (c) 2020 Synthetix
*
* Permission is hereby granted, free of charge, to any person obtaining a copy
* of this software and associated documentation files (the "Software"), to deal
* in the Software without restriction, including without limitation the rights
* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
* copies of the Software, and to permit persons to whom the Software is
* furnished to do so, subject to the following conditions:
*
* The above copyright notice and this permission notice shall be included in all
* copies or substantial portions of the Software.
*
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
*/

// File: @openzeppelin/contracts/math/Math.sol

pragma solidity ^0.5.0; // 成都链安 // 建议固定编译器版本

/**
 * @dev Standard math utilities missing in the Solidity language.
 */
library Math {
    /**
     * @dev Returns the largest of two numbers.
     */
    function max(uint256 a, uint256 b) internal pure returns (uint256) {
        return a >= b ? a : b;
```

```
    }

    /**
     * @dev Returns the smallest of two numbers.
     */
    function min(uint256 a, uint256 b) internal pure returns (uint256) {
        return a < b ? a : b;
    }

    /**
     * @dev Returns the average of two numbers. The result is rounded towards
     * zero.
     */
    function average(uint256 a, uint256 b) internal pure returns (uint256) {
        // (a + b) / 2 can overflow, so we distribute
        return (a / 2) + (b / 2) + ((a % 2 + b % 2) / 2);
    }
}

// File: @openzeppelin/contracts/math/SafeMath.sol

pragma solidity ^0.5.0; // 成都链安 // 建议固定编译器版本

/**
 * @dev Wrappers over Solidity's arithmetic operations with added overflow
 * checks.
 *
 * Arithmetic operations in Solidity wrap on overflow. This can easily result
 * in bugs, because programmers usually assume that an overflow raises an
 * error, which is the standard behavior in high level programming languages.
 * `SafeMath` restores this intuition by reverting the transaction when an
 * operation overflows.
 *
 * Using this library instead of the unchecked operations eliminates an entire
 * class of bugs, so it's recommended to use it always.
 */
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) {
        return sub(a, b, "SafeMath: subtraction overflow");
    }

    /**
     * @dev Returns the subtraction of two unsigned integers, reverting with custom
message on
     * overflow (when the result is negative).
     *
     * Counterpart to Solidity's `-` operator.
     *
     * Requirements:
     * - Subtraction cannot overflow.
     *
     * _Available since v2.4.0._
     */
    function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns
(uint256) {
        require(b <= a, errorMessage);
        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) {
        return div(a, b, "SafeMath: division by zero");
    }

    /**
     * @dev Returns the integer division of two unsigned integers. Reverts with custom
message 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.
     *
     * _Available since v2.4.0._
     */
    function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns
(uint256) {
        // Solidity only automatically asserts when dividing by 0
        require(b > 0, errorMessage);
        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) {
        return mod(a, b, "SafeMath: modulo by zero");
    }

    /**
     * @dev Returns the remainder of dividing two unsigned integers. (unsigned integer
modulo),
     * Reverts with custom message 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.
     *
     * _Available since v2.4.0._
     */
    function mod(uint256 a, uint256 b, string memory errorMessage) internal pure returns
(uint256) {
        require(b != 0, errorMessage);
        return a % b;
    }
}

// File: @openzeppelin/contracts/GSN/Context.sol

pragma solidity ^0.5.0; // 成都链安 // 建议固定编译器版本

/*
 * @dev Provides information about the current execution context, including the
 * sender of the transaction and its data. While these are generally available
```

```
 * via msg.sender and msg.data, they should not be accessed in such a direct
 * manner, since when dealing with GSN meta-transactions the account sending and
 * paying for execution may not be the actual sender (as far as an application
 * is concerned).
 *
 * This contract is only required for intermediate, library-like contracts.
 */
contract Context {
    // Empty internal constructor, to prevent people from mistakenly deploying
    // an instance of this contract, which should be used via inheritance.
    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;
    }
    // 成都链安 // 内部函数_msgData，返回调用数据
    function _msgData() internal view returns (bytes memory) {
        this; // silence state mutability warning without generating bytecode - see
https://github.com/ethereum/solidity/issues/2691
        return msg.data;
    }
}

// File: @openzeppelin/contracts/ownership/Ownable.sol

pragma solidity ^0.5.0; // 成都链安 // 建议固定编译器版本

/**
 * @dev Contract module which provides a basic access control mechanism, where
 * there is an account (an owner) that can be granted exclusive access to
 * specific functions.
 *
 * This module is used through inheritance. It will make available the modifier
 * `onlyOwner`, which can be applied to your functions to restrict their use to
 * the owner.
 */
contract Ownable is Context {
    address private _owner; // 成都链安 // 声明变量_owner，存储合约所有者地址

    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);
// 成都链安 // 声明合约所有者权限转移事件

    /**
     * @dev Initializes the contract setting the deployer as the initial owner.
     */
    constructor () internal {
        _owner = _msgSender();
```

```
        emit OwnershipTransferred(address(0), _owner); // 成都链安 // 触发
OwnershipTransferred 事件
    }

    /**
     * @dev Returns the address of the current owner.
     */
    function owner() public view returns (address) {
        return _owner;
    }

    /**
     * @dev Throws if called by any account other than the owner.
     */
    modifier onlyOwner() {
        require(isOwner(), "Ownable: caller is not the owner"); // 成都链安 // 修饰器,
要求被修饰函数的调用者必须为合约所有者
        _;
    }

    /**
     * @dev Returns true if the caller is the current owner.
     */
    function isOwner() public view returns (bool) {
        return _msgSender() == _owner;
    }

    /**
     * @dev Leaves the contract without owner. It will not be possible to call
     * `onlyOwner` functions anymore. Can only be called by the current owner.
     *
     * NOTE: Renouncing ownership will leave the contract without an owner,
     * thereby removing any functionality that is only available to the owner.
     */
    function renounceOwnership() public onlyOwner {
        emit OwnershipTransferred(_owner, address(0)); // 成都链安 // 触发
OwnershipTransferred 事件
        _owner = address(0); // 成都链安 // 转移合约所有者权限至零地址
    }

    /**
     * @dev Transfers ownership of the contract to a new account (`newOwner`).
     * Can only be called by the current owner.
     */
    function transferOwnership(address newOwner) public onlyOwner {
        _transferOwnership(newOwner);
    }
```

```
    /**
     * @dev Transfers ownership of the contract to a new account (`newOwner`).
     */
    function _transferOwnership(address newOwner) internal {
        require(newOwner != address(0), "Ownable: new owner is the zero address"); // 成都链安 // newOwner 非零地址检查，避免合约所有者权限丢失
        emit OwnershipTransferred(_owner, newOwner); // 成都链安 // 触发 OwnershipTransferred 事件
        _owner = newOwner; // 成都链安 // 转移合约所有者权限至 newOwner
    }
}

// File: @openzeppelin/contracts/token/ERC20/IERC20.sol

pragma solidity ^0.5.0; // 成都链安 // 建议固定编译器版本

/**
 * @dev Interface of the ERC20 standard as defined in the EIP. Does not include
 * the optional functions; to access them see {ERC20Detailed}.
 */
interface IERC20 {
    // 成都链安 // 定义 ERC20 Token 标准要求的接口函数
    /**
     * @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);
    function mint(address account, uint amount) external;

    /**
     * @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);
    // 成都链安 // 声明代币转账事件与代币授权事件
    /**
     * @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);

    /**
     * @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);
}

// File: @openzeppelin/contracts/utils/Address.sol
```

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

/**
 * @dev Collection of functions related to the address type
 */
library Address {
    /**
     * @dev Returns true if `account` is a contract.
     *
     * This test is non-exhaustive, and there may be false-negatives: during the
     * execution of a contract's constructor, its address will be reported as
     * not containing a contract.
     *
     * IMPORTANT: It is unsafe to assume that an address for which this
     * function returns false is an externally-owned account (EOA) and not a
     * contract.
     */
    function isContract(address account) internal view returns (bool) {
        // This method relies in extcodesize, which returns 0 for contracts in
        // construction, since the code is only stored at the end of the
        // constructor execution.

        // According to EIP-1052, 0x0 is the value returned for not-yet created accounts
        // and 0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470 is
returned
        // for accounts without code, i.e. `keccak256('')`
        bytes32 codehash;
        bytes32 accountHash =
0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470;
        // solhint-disable-next-line no-inline-assembly
        assembly { codehash := extcodehash(account) }
        return (codehash != 0x0 && codehash != accountHash);
    }

    /**
     * @dev Converts an `address` into `address payable`. Note that this is
     * simply a type cast: the actual underlying value is not changed.
     *
     * _Available since v2.4.0._
     */
    function toPayable(address account) internal pure returns (address payable) {
        return address(uint160(account));
    }

    /**
     * @dev Replacement for Solidity's `transfer`: sends `amount` wei to
     * `recipient`, forwarding all available gas and reverting on errors.
     *
```

```
     * https://eips.ethereum.org/EIPS/eip-1884[EIP1884] increases the gas cost
     * of certain opcodes, possibly making contracts go over the 2300 gas limit
     * imposed by `transfer`, making them unable to receive funds via
     * `transfer`. {sendValue} removes this limitation.
     *
     * https://diligence.consensys.net/posts/2019/09/stop-using-soliditys-transfer-now/[Learn more].
     *
     * IMPORTANT: because control is transferred to `recipient`, care must be
     * taken to not create reentrancy vulnerabilities. Consider using
     * {ReentrancyGuard} or the
     * https://solidity.readthedocs.io/en/v0.5.11/security-considerations.html#use-the-checks-effects-interactions-pattern[checks-effects-interactions pattern].
     *
     * _Available since v2.4.0._
     */
    function sendValue(address payable recipient, uint256 amount) internal {
        require(address(this).balance >= amount, "Address: insufficient balance");

        // solhint-disable-next-line avoid-call-value
        (bool success, ) = recipient.call.value(amount)("");
        require(success, "Address: unable to send value, recipient may have reverted");
    }
}

// File: @openzeppelin/contracts/token/ERC20/SafeERC20.sol

pragma solidity ^0.5.0;



/**
 * @title SafeERC20
 * @dev Wrappers around ERC20 operations that throw on failure (when the token
 * contract returns false). Tokens that return no value (and instead revert or
 * throw on failure) are also supported, non-reverting calls are assumed to be
 * successful.
 * To use this library you can add a `using SafeERC20 for ERC20;` statement to your contract,
 * which allows you to call the safe operations as `token.safeTransfer(...)`, etc.
 */
library SafeERC20 {
    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
    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, uint256 value) internal {
        callOptionalReturn(token, abi.encodeWithSelector(token.transfer.selector, to,
value));
    }

    function safeTransferFrom(IERC20 token, address from, address to, uint256 value)
internal {
        callOptionalReturn(token, abi.encodeWithSelector(token.transferFrom.selector,
from, to, value));
    }

    function safeApprove(IERC20 token, address spender, uint256 value) internal {
        // safeApprove should only be called when setting an initial allowance,
        // or when resetting it to zero. To increase and decrease it, use
        // 'safeIncreaseAllowance' and 'safeDecreaseAllowance'
        // solhint-disable-next-line max-line-length
        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 safeIncreaseAllowance(IERC20 token, address spender, uint256 value)
internal {
        uint256 newAllowance = token.allowance(address(this), spender).add(value);
        callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector,
spender, newAllowance));
    }

    function safeDecreaseAllowance(IERC20 token, address spender, uint256 value)
internal {
        uint256 newAllowance = token.allowance(address(this), spender).sub(value,
"SafeERC20: decreased allowance below zero");
        callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector,
spender, newAllowance));
    }

    /**
     * @dev Imitates a Solidity high-level call (i.e. a regular function call to a
contract), relaxing the requirement
     * on the return value: the return value is optional (but if data is returned, it
must not be false).
     * @param token The token targeted by the call.
     * @param data The call data (encoded using abi.encode or one of its variants).
     */
    function callOptionalReturn(IERC20 token, bytes memory data) private {
        // We need to perform a low level call here, to bypass Solidity's return data
```

```
size checking mechanism, since
        // we're implementing it ourselves.

        // A Solidity high level call has three parts:
        //  1. The target address is checked to verify it contains contract code
        //  2. The call itself is made, and success asserted
        //  3. The return value is decoded, which in turn checks the size of the
returned data.
        // solhint-disable-next-line max-line-length
        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");
        }
    }
}

// File: contracts/IRewardDistributionRecipient.sol

pragma solidity ^0.5.0;



contract IRewardDistributionRecipient is Ownable {
    address rewardDistribution; // 成都链安 // 声明变量 rewardDistribution 存储奖励分配
地址

    function notifyRewardAmount(uint256 reward) external; // 成都链安 // 定义
notifyRewardAmount 函数接口

    modifier onlyRewardDistribution() {
        require(_msgSender() == rewardDistribution, "Caller is not reward
distribution"); // 成都链安 // 修饰器，要求被修饰函数的调用者必须为 rewardDistribution
        _;
    }
    // 成都链安 // 设置 rewardDistribution 地址
    function setRewardDistribution(address _rewardDistribution)
        external
        onlyOwner
    {
        rewardDistribution = _rewardDistribution;
    }
```

```
}

// File: contracts/CurveRewards.sol

pragma solidity ^0.5.0;




contract LPTokenCRTWrapper {
    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
    using SafeERC20 for IERC20; // 成都链安 // 引入 SafeERC20 库，其内部函数用于安全外部 ERC20 合约转账相关操作

    IERC20 public y = IERC20(0x41fdcfd438d2f94fa3acb0891f18128e27032ddc87); // Stake Token address // 成都链安 // 声明外部 y 合约实例

    uint256 private _totalSupply; // 成都链安 // 声明变量_totalSupply，存储抵押总量
    mapping(address => uint256) private _balances; // 成都链安 // 声明 mapping 变量 _balances，记录指定地址的抵押数量
    // 成都链安 // 抵押总量查询函数，返回抵押总量
    function totalSupply() public view returns (uint256) {
        return _totalSupply;
    }
    // 成都链安 // 抵押数量查询函数，返回指定地址 account 的抵押数量
    function balanceOf(address account) public view returns (uint256) {
        return _balances[account];
    }
    // 成都链安 // 抵押函数
    function stake(uint256 amount) public {
        _totalSupply = _totalSupply.add(amount); // 成都链安 // 更新抵押总量
        _balances[msg.sender] = _balances[msg.sender].add(amount); // 成都链安 // 修改调用者的抵押数量
        y.safeTransferFrom(msg.sender, address(this), amount); // 成都链安 // 通过 IERC20 合约实体所引用的 SafeERC20 库中 safeTransferFrom 函数发起指定函数调用，调用指定代币合约的 transferFrom 函数，代理当前函数调用者转出其指定数量的抵押代币至本合约地址
    }
    // 成都链安 // 提取函数
    function withdraw(uint256 amount) public {
        _totalSupply = _totalSupply.sub(amount); // 成都链安 // 更新抵押总量
        _balances[msg.sender] = _balances[msg.sender].sub(amount); // 成都链安 // 修改调用者的抵押数量
        y.safeTransfer(msg.sender, amount); // 成都链安 // 通过 IERC20 合约实体所引用的 SafeERC20 库中 safeTransfer 函数发起指定函数调用，调用指定代币合约的 transfer 函数，转出指定数量的抵押代币至当前函数调用者
```

```
    }
}

contract MoonPoolCRTReward is LPTokenCRTWrapper, IRewardDistributionRecipient {
    IERC20 public mfi = IERC20(0x4177fd35cfd3349cb1ca14b9e71b5b7fd373331878);  // MFI Token address // 成都链安 // 初始化 MFI 合约实例
    uint256 public constant DURATION = 2 days; // 成都链安 // 声明常量 DURATION，存储奖励计算间隔时间，默认为 2 天

    uint256 public initreward = 750*1e18; // 成都链安 // 声明变量 initreward，存储初始奖励数量
    uint256 public starttime = 1599310800; //Saturday, 5 September 2020 13:00:00 (UTC)
    uint256 public periodFinish = 0; // 成都链安 // 声明变量 periodFinish，存储阶减半段完成时间
    uint256 public rewardRate = 0; // 成都链安 // 声明变量 rewardRate，存储奖励比例
    uint256 public lastUpdateTime; // 成都链安 // 声明变量 lastUpdateTime，存储上次更新时间
    uint256 public rewardPerTokenStored; // 成都链安 // 声明变量 rewardPerTokenStored，存储动态时间间隔内单个抵押代币可获得的奖励代币数量
    address public governance; // 成都链安 // 声明变量 governance，存储治理管理员地址
    mapping(address => uint256) public userRewardPerTokenPaid; // 成都链安 // 声明 mapping 变量 userRewardPerTokenPaid，存储指定地址已支付（计算）的奖励
    mapping(address => uint256) public rewards; // 成都链安 // 声明 mapping 变量 rewards，存储指定地址所获得的抵押奖励总量
    // 成都链安 // 声明奖励相关事件
    event RewardAdded(uint256 reward);
    event Staked(address indexed user, uint256 amount);
    event Withdrawn(address indexed user, uint256 amount);
    event RewardPaid(address indexed user, uint256 reward);
    // 成都链安 // 修饰器，更新奖励相关数据
    modifier updateReward(address account) {
        rewardPerTokenStored = rewardPerToken(); // 成都链安 // 调用内部函数 rewardPerToken，获取特定奖励比例下每个抵押代币可获得的抵押奖励
        lastUpdateTime = lastTimeRewardApplicable(); // 成都链安 // 调用函数 lastTimeRewardApplicable 确定上次更新时间为当前时间戳与阶段完成时间中最早的时间戳
        if (account != address(0)) {
            rewards[account] = earned(account); // 成都链安 // 更新指定地址 account 所获得的抵押奖励总量
            userRewardPerTokenPaid[account] = rewardPerTokenStored; // 成都链安 // 更新指定地址 account 已支付（计算）的奖励
        }
        _;
    }
    // 成都链安 // 构造函数，初始化治理管理员地址
    constructor (address g) public{
      governance = g;
    }
    // 成都链安 // 获取当前时间戳与阶段完成时间中最早的时间戳
```

```
    function lastTimeRewardApplicable() public view returns (uint256) {
        return Math.min(block.timestamp, periodFinish);
    }
    // 成都链安 // 计算每个代币可获得的抵押奖励函数量
    function rewardPerToken() public view returns (uint256) {
        // 成都链安 // 如果该合约抵押总量为 0，则返回当前奖励比例下每个抵押代币可获得的抵押奖励
        if (totalSupply() == 0) {
            return rewardPerTokenStored;
        }
        return
            rewardPerTokenStored.add(
                lastTimeRewardApplicable()
                    .sub(lastUpdateTime) // 成都链安 // 计算距离上次更新时间后的时间差
                    .mul(rewardRate) // 成都链安 // 根据当前奖励比例计算该段时间获得的抵押奖励数量
                    .mul(1e18) // 成都链安 // 代币精度换算
                    .div(totalSupply())
            );
    }
    // 成都链安 // 奖励查询函数
    function earned(address account) public view returns (uint256) {
        return
            balanceOf(account)
                .mul(rewardPerToken().sub(userRewardPerTokenPaid[account])) // 成都链安 // 指定地址 account 在实际阶段获取的抵押奖励
                .div(1e18) // 成都链安 // 精度换算
                .add(rewards[account]); // 成都链安 // 指定地址 account 所获取的总抵押奖励
    }
    // 成都链安 // 重写 stake 函数，增加修饰器 updateReward，checkhalve 以及 checkStart
    // stake visibility is public as overriding LPTokenWrapper's stake() function
    function stake(uint256 amount) public updateReward(msg.sender) checkhalve checkStart{
        require(amount > 0, "Cannot stake 0");
        super.stake(amount);
        emit Staked(msg.sender, amount);
    }
    // 成都链安 // 重写 withdraw 函数，增加修饰器 updateReward，checkhalve 以及 checkStart
    function withdraw(uint256 amount) public updateReward(msg.sender) checkhalve checkStart{
        require(amount > 0, "Cannot withdraw 0");
        super.withdraw(amount); // 成都链安 // 执行父合约的 withdraw 函数
        emit Withdrawn(msg.sender, amount); // 成都链安 // 触发 Withdrawn 事件
    }
    // 成都链安 // 退出函数，调用者提取全部抵押代币与抵押奖励
    function exit() external {
        withdraw(balanceOf(msg.sender)); // 成都链安 // 调用 withdraw 函数提取全部抵押代
```

```
币
        getReward(); // 成都链安 // 调用 getReward 函数提取抵押奖励
    }
    // 成都链安 // 提取抵押奖励函数
    function getReward() public updateReward(msg.sender) checkhalve checkStart{
        uint256 reward = earned(msg.sender); // 成都链安 // 声明局部变量 reward，获取调用者的抵押奖励
        if (reward > 0) { // 成都链安 // 要求提取抵押奖励不得为 0
            rewards[msg.sender] = 0; // 成都链安 // 清空调用者的抵押奖励
            mfi.safeTransfer(msg.sender, reward); // 成都链安 // 通过 IERC20 合约实体所引用的 SafeERC20 库中 safeTransfer 函数发起指定函数调用，调用指定代币合约的 transfer 函数，转出指定数量的 MFI 代币至当前函数调用者
            mfi.mint(governance, reward.div(10)); // 成都链安 // 调用 MFI 合约的 mint 函数向 governance 地址铸币用户奖励的 10%
            emit RewardPaid(msg.sender, reward); // 成都链安 // 触发 RewardPaid 事件
        }
    }
    // 成都链安 // 修饰器，在当前时间到达阶段完成时间时进行奖励减半操作
    modifier checkhalve(){
        if (block.timestamp >= periodFinish) {
            initreward = initreward.mul(50).div(100); // 成都链安 // 奖励总数减半

            mfi.mint(address(this),initreward); // 成都链安 // 调用 MFI 合约的 mint 函数向该合约地址铸币

            rewardRate = initreward.div(DURATION); // 成都链安 // 更新奖励比例
            periodFinish = block.timestamp.add(DURATION); // 成都链安 // 更新阶段完成时间
            emit RewardAdded(initreward); // 成都链安 // 触发 RewardAdded 事件
        }
        _;
    }
    // 成都链安 // 修饰器，要求被修饰函数仅当时间到达开始时间时可被调用
    modifier checkStart(){
        require(block.timestamp > starttime,"not start");
        _;
    }
    // 成都链安 // 根据指定总奖励数量 reward 计算奖励比例并更新 lastUpdateTime 与 periodFinish
    // 成都链安 // 注意：该函数可由指定地址 rewardDistribution 调用来控制奖励比例与关键时间判断节点
    function notifyRewardAmount(uint256 reward)
        external
        onlyRewardDistribution // 成都链安 // 调用权限检查，要求调用者必须为 rewardDistribution 地址
        updateReward(address(0)) // 成都链安 // 更新每个代币可获得的抵押奖励函数量与 lastUpdateTime
    {
```

```
        if (block.timestamp >= periodFinish) {
            rewardRate = reward.div(DURATION); // 成都链安 // 当时间到达阶段完成时间时，更新奖励比例
        } else { // 成都链安 // 否则计算剩余阶段应得奖励并更新奖励比例
            uint256 remaining = periodFinish.sub(block.timestamp); // 成都链安 // 计算阶段剩余时间
            uint256 leftover = remaining.mul(rewardRate); // 成都链安 // 根据当前奖励比例计算剩余阶段应得抵押奖励
            rewardRate = reward.add(leftover).div(DURATION); // 成都链安 // 更新奖励比例
        }

        mfi.mint(address(this),reward); // 成都链安 // 调用 MFI 合约的 mint 函数向该合约地址铸币

        lastUpdateTime = block.timestamp; // 成都链安 // 更新 lastUpdateTime 为当前时间
        periodFinish = block.timestamp.add(DURATION); // 成都链安 // 更新阶段完成时间
        emit RewardAdded(reward); // 成都链安 // 触发 RewardAdded 事件
    }
}
```

成都链安
BEOSIN

官方网址

https://lianantech.com

电子邮箱

vaas@lianantech.com

微信公众号