智能合约安全审计报告

目录

1 前言

慢雾安全团队于 2021 年 04 月 01 日，收到 Booster 团队对 BoosterProtocol 安全审计的申请，根据项目特点慢雾安全团队制定如下审计方案。

慢雾安全团队将采用“白盒为主，黑灰为辅”的策略，以最贴近真实攻击的方式，对项目进行安全审计。

慢雾科技 DeFi 项目测试方法：

黑盒测试	站在外部从攻击者角度进行安全测试。
灰盒测试	通过脚本工具对代码模块进行安全测试，观察内部运行状态，挖掘弱点。
白盒测试	基于项目的源代码，进行脆弱性分析和漏洞挖掘。

慢雾科技 DeFi 漏洞风险等级：

严重漏洞	严重漏洞会对项目的安全造成重大影响，强烈建议修复严重漏洞。
高危漏洞	高危漏洞会影响项目的正常运行，强烈建议修复高危漏洞。
中危漏洞	中危漏洞会影响项目的运行，建议修复中危漏洞。
低危漏洞	低危漏洞可能在特定场景中会影响项目的业务操作，建议项目方自行评估和考虑这些问题是否需要修复。
弱点	理论上存在安全隐患，但工程上极难复现。
增强建议	编码或架构存在更好的实践方法。

2. 审计方法

慢雾安全团队智能合约安全审计流程包含两个步骤:

- 使用开源或内部自动化分析的工具对合约代码中常见的安全漏洞进行扫描和测试。
- 人工审计代码的安全问题，通过人工分析合约代码，发现代码中潜在的安全问题。

如下是合约代码审计过程中慢雾安全团队会重点审查的漏洞列表:

（其他未知安全漏洞不包含在本次审计责任范围）

- 重入攻击
- 重放攻击
- 重排攻击
- 短地址攻击
- 拒绝服务攻击
- 交易顺序依赖
- 条件竞争攻击
- 权限控制攻击
- 整数上溢/下溢攻击
- 时间戳依赖攻击
- Gas 使用，Gas 限制和循环
- 冗余的回调函数
- 不安全的接口使用
- 函数状态变量的显式可见性
- 逻辑缺陷
- 未声明的存储指针
- 算术精度误差
- tx.origin 身份验证
- 假充值漏洞
- 变量覆盖

3. 项目背景

3.1 项目介绍

Booster 是 DeFi 的一站式服务平台，用户可在 Booster 完成存款、借款、杠杆挖矿、跨链挖矿等一些列 DeFi 行为。

项目官网地址:

https://booster.farm/

审计版本代码:

https://github.com/boosterfarm/boosterProtocol/tree/946b15629c410d706856584f3aa04001d6a

55bd2

已提供的文档：

https://docs.booster.farm/

修复版本代码：

https://github.com/boosterfarm/boosterProtocol/tree/1d9fc8692cae442b94ec091e866ee5194e0dd3aa

由于修复版本中的代码包含项目方新增和迭代的功能代码，慢雾安全团队仅对发现的安全问题对应的修复代码进行了 review，新增代码和迭代修改的代码本次审计并未涉及。

4. 代码概述

4.1 合约可见性分析

在审计过程中，慢雾安全团队对核心合约的可见性进行分析，结果如下:

ActionCompPools			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
<Receive Ether>	External	Payable	-
poolLength	External	-	-
getPoolInfo	External	-	-
getPoolIndex	External	-	-
add	External	Can Modify State	onlyOwner
setRewardMaxPerBlock	External	Can Modify State	onlyOwner
setAutoUpdate	External	Can Modify State	onlyOwner
setAutoClaim	External	Can Modify State	onlyOwner
setRewardRestricted	External	Can Modify State	onlyOwner
setBooDev	External	Can Modify State	-
getBlocksReward	Public	-	-
pendingRewards	Public	-	-
pendingRewards	Public	-	-

totalRewards	Public	-	-
massUpdatePools	Public	Can Modify State	-
updatePool	Public	Can Modify State	-
onAcionIn	External	Can Modify State	-
onAcionOut	External	Can Modify State	-
onAcionClaim	External	Can Modify State	-
onAcionEmergency	External	Can Modify State	-
onAcionUpdate	External	Can Modify State	-
mintRewards	External	Can Modify State	-
deposit	Internal	Can Modify State	-
withdraw	Internal	Can Modify State	-
claimIds	External	Can Modify State	-
claim	Public	Can Modify State	-
_claim	Internal	Can Modify State	-
emergencyWithdraw	Internal	Can Modify State	-
safeTokenTransfer	Internal	Can Modify State	-

ActionPools			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
<Receive Ether>	External	Payable	-
poolLength	External	-	-
getPoolInfo	External	-	-
getPoolIndex	External	-	-
add	External	Can Modify State	onlyOwner
setRewardMaxPerBlock	External	Can Modify State	onlyOwner
setAutoUpdate	External	Can Modify State	onlyOwner
setAutoClaim	External	Can Modify State	onlyOwner
setRewardRestricted	External	Can Modify State	onlyOwner
setBooDev	External	Can Modify State	-
getBlocksReward	Public	-	-
pendingRewards	Public	-	-
totalRewards	Public	-	-
massUpdatePools	Public	Can Modify State	-

updatePool	Public	Can Modify State	-
onAcionIn	External	Can Modify State	-
onAcionOut	External	Can Modify State	-
onAcionClaim	External	Can Modify State	-
onAcionEmergency	External	Can Modify State	-
onAcionUpdate	External	Can Modify State	-
mintRewards	External	Can Modify State	-
deposit	Internal	Can Modify State	-
withdraw	Internal	Can Modify State	-
claimIds	External	Can Modify State	-
claim	Public	Can Modify State	-
_claim	Internal	Can Modify State	-
emergencyWithdraw	Internal	Can Modify State	-
safesub	Internal	-	-
safeTokenTransfer	Internal	Can Modify State	-

BOOTimelock			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
quota	Public	-	-
unlock	External	Can Modify State	-

BOOToken			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20Capped ERC20
setMintWhitelist	External	Can Modify State	onlyOwner
mint	External	Can Modify State	-
burn	External	Can Modify State	onlyOwner

SafeBoxCToken			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	SafeBoxCTokenImpl
getCATPoolInfo	External	-	-

getCATUserAmount	External	-	-
getSource	External	-	-
setBlacklist	External	Can Modify State	onlyOwner
setCompAcionPool	Public	Can Modify State	onlyOwner
setBuyback	Public	Can Modify State	onlyOwner
setBorrowLimitRate	External	Can Modify State	onlyOwner
setBorrowMinAmount	External	Can Modify State	onlyOwner
setEmergencyRepay	External	Can Modify State	onlyOwner
setEmergencyWithdraw	External	Can Modify State	onlyOwner
setOptimalUtilizationRate	External	Can Modify State	onlyOwner
setStableRateSlope	External	Can Modify State	onlyOwner
supplyRatePerBlock	External	-	-
borrowRatePerBlock	External	-	-
borrowInfoLength	External	-	-
getBorrowInfo	External	-	-
getBorrowFactorPrewiew	Public	-	-
getBorrowFactor	Public	Can Modify State	-
_getBorrowFactor	Public	-	-
getBorrowTotal	Public	-	-
getDepositTotal	Public	-	-
getBaseTokenPerLPToken	Public	-	-
pendingSupplyAmount	External	-	-
pendingBorrowAmount	Public	-	-
pendingBorrowRewards	Public	-	-
deposit	External	Can Modify State	nonReentrant
_deposit	Internal	Can Modify State	-
withdraw	External	Can Modify State	nonReentrant
_withdraw	Internal	Can Modify State	-
claim	External	Can Modify State	nonReentrant
_claim	Internal	Can Modify State	-
getBorrowId	Public	-	-
getBorrowId	External	Can Modify State	onlyBank
borrow	External	Can Modify State	onlyBank
_borrow	Internal	Can Modify State	-

repay	External	Can Modify State	-
_repay	Internal	Can Modify State	-
emergencyWithdraw	External	Can Modify State	nonReentrant
emergencyRepay	External	Can Modify State	nonReentrant
update	Public	Can Modify State	-
_update	Public	Can Modify State	-
mintDonate	Public	Can Modify State	nonReentrant
tokenSafeTransfer	Internal	Can Modify State	-

SafeBoxCTokenETH			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	SafeBoxCTokenImplETH
getCATPoolInfo	External	-	-
getCATUserAmount	External	-	-
getSource	External	-	-
setBlacklist	External	Can Modify State	onlyOwner
setCompAcionPool	Public	Can Modify State	onlyOwner
setBuyback	Public	Can Modify State	onlyOwner
setBorrowLimitRate	External	Can Modify State	onlyOwner
setBorrowMinAmount	External	Can Modify State	onlyOwner
setEmergencyRepay	External	Can Modify State	onlyOwner
setEmergencyWithdraw	External	Can Modify State	onlyOwner
setOptimalUtilizationRate	External	Can Modify State	onlyOwner
setStableRateSlope	External	Can Modify State	onlyOwner
supplyRatePerBlock	External	-	-
borrowRatePerBlock	External	-	-
borrowInfoLength	External	-	-
getBorrowInfo	External	-	-
getBorrowFactorPrewiew	Public	-	-
getBorrowFactor	Public	Can Modify State	-
_getBorrowFactor	Public	-	-
getBorrowTotal	Public	-	-
getDepositTotal	Public	-	-
getBaseTokenPerLPToken	Public	-	-

pendingSupplyAmount	External	-	-
pendingBorrowAmount	Public	-	-
pendingBorrowRewards	Public	-	-
deposit	External	Can Modify State	nonReentrant
_deposit	Internal	Can Modify State	-
withdraw	External	Can Modify State	nonReentrant
_withdraw	Internal	Can Modify State	-
claim	External	Can Modify State	nonReentrant
_claim	Internal	Can Modify State	-
getBorrowId	Public	-	-
getBorrowId	External	Can Modify State	onlyBank
borrow	External	Can Modify State	onlyBank
_borrow	Internal	Can Modify State	-
repay	External	Can Modify State	-
_repay	Internal	Can Modify State	-
emergencyWithdraw	External	Can Modify State	nonReentrant
emergencyRepay	External	Can Modify State	nonReentrant
update	Public	Can Modify State	-
_update	Public	Can Modify State	-
mintDonate	Public	Can Modify State	nonReentrant
tokenSafeTransfer	Internal	Can Modify State	-

SafeBoxCTokenImpl			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20
baseToken	Public	–	–
ctokenSupplyRatePerBlock	Public	–	–
ctokenBorrowRatePerBlock	Public	–	–
call_balanceOf	Public	–	–
call_balanceOfCToken_this	Public	–	–
call_balanceOfBaseToken_this	Public	Can Modify State	–
call_borrowBalanceCurrent_this	Public	Can Modify State	–
getBaseTokenPerCToken	Public	–	–
ctokenDeposit	Internal	Can Modify State	–

ctokenWithdraw	Internal	Can Modify State	–
ctokenClaim	Internal	Can Modify State	–
ctokenBorrow	Internal	Can Modify State	–
ctokenRepayBorrow	Internal	Can Modify State	–

SafeBoxCTokenImplETH			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20
<Receive Ether>	External	Payable	-
baseToken	Public	-	-
ctokenSupplyRatePerBlock	Public	-	-
ctokenBorrowRatePerBlock	Public	-	-
call_balanceOf	Public	-	-
call_balanceOfCToken_this	Public	-	-
call_balanceOfBaseToken_this	Public	Can Modify State	-
call_borrowBalanceCurrent_this	Public	Can Modify State	-
getBaseTokenPerCToken	Public	-	-
ctokenDeposit	Internal	Can Modify State	-
ctokenWithdraw	Internal	Can Modify State	-
ctokenClaim	Internal	Can Modify State	-
ctokenBorrow	Internal	Can Modify State	-
ctokenRepayBorrow	Internal	Can Modify State	-

SafeBoxFilDa			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	SafeBoxCToken
update	Public	Can Modify State	-
setFildaDepositPool	Public	Can Modify State	onlyOwner
setFildaBorrowPool	Public	Can Modify State	onlyOwner
checkFildaPool	Internal	-	-
deposit	External	Can Modify State	nonReentrant
withdraw	External	Can Modify State	nonReentrant
borrow	External	Can Modify State	onlyBank

repay	External	Can Modify State	-
updatetoken	Public	Can Modify State	-
claim	External	Can Modify State	nonReentrant

StrategyMDex			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getSource	External	-	-
poolLength	External	-	-
getCATPoolInfo	External	-	-
getCATUserAmount	External	-	-
getPoolInfo	External	-	-
getPoolCollateralToken	External	-	-
getPoolIpToken	External	-	-
getBaseToken	External	-	-
getBorrowInfo	External	-	-
getTokenBalance_this	Internal	-	-
addPool	Public	Can Modify State	onlyOwner
resetApprove	Public	Can Modify State	onlyOwner
setCompAcionPool	External	Can Modify State	onlyOwner
setSConfig	External	Can Modify State	onlyOwner
setBuyback	External	Can Modify State	onlyOwner
setMiniRewardAmount	External	Can Modify State	onlyOwner
pendingRewards	Public	-	-
pendingLPAmount	Public	-	-
getBorrowAmount	Public	-	-
getBorrowAmountInBaseToken	Public	-	-
getDepositAmount	External	-	-
massUpdatePools	External	Can Modify State	-
updatePool	Public	Can Modify State	-
depositLPToken	Public	Can Modify State	onlyBank
deposit	Public	Can Modify State	onlyBank
makeBorrowBaseToken	Internal	Can Modify State	-
makeBalanceOptimalLiquidity	Internal	Can Modify State	-

makeBalanceOptimalLiquidityByAmount	Internal	Can Modify State	-
makeLiquidityAndDeposit	Internal	Can Modify State	-
makeLiquidityAndDepositByAmount	Internal	Can Modify State	-
withdrawLPToken	External	Can Modify State	onlyBank
withdraw	Public	Can Modify State	onlyBank
_withdraw	Internal	Can Modify State	-
makeWithdrawCalcAmount	Public	-	-
makeWithdrawRemoveLiquidity	Internal	Can Modify State	-
repayBorrow	Public	Can Modify State	onlyBank
emergencyWithdraw	External	Can Modify State	onlyBank
_emergencyWithdraw	Internal	Can Modify State	-
liquidation	External	Can Modify State	onlyBank
makeExtraRewards	External	Can Modify State	-

SafeBoxFilDaETH			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	SafeBoxCTokenETH
update	Public	Can Modify State	-
setFildaDepositPool	Public	Can Modify State	onlyOwner
setFildaBorrowPool	Public	Can Modify State	onlyOwner
checkFildaPool	Internal	-	-
deposit	External	Can Modify State	nonReentrant
withdraw	External	Can Modify State	nonReentrant
borrow	External	Can Modify State	onlyBank
repay	External	Can Modify State	-
updatetoken	Public	Can Modify State	-
claim	External	Can Modify State	nonReentrant

StrategyConfig			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setFeeGather	External	Can Modify State	onlyOwner
setReservedGather	External	Can Modify State	onlyOwner

getBorrowFactor	Public	-	-
checkBorrowAndLiquidation	Internal	Can Modify State	-
setBorrowFactor	External	Can Modify State	onlyOwner
getLiquidationFactor	Public	-	-
setLiquidationFactor	External	Can Modify State	onlyOwner
getFarmPoolFactor	External	-	-
setFarmPoolFactor	External	Can Modify State	onlyOwner
getDepositFee	External	-	-
setDepositFee	External	Can Modify State	onlyOwner
getWithdrawFee	External	-	-
setWithdrawFee	External	Can Modify State	onlyOwner
getRefundFee	External	-	-
setRefundFee	External	Can Modify State	onlyOwner
getClaimFee	External	-	-
setClaimFee	External	Can Modify State	onlyOwner
getLiquidationFee	External	-	-
setLiquidationFee	External	Can Modify State	onlyOwner

StrategyMDexPools			
Function Name	Visibility	Mutability	Modifiers
poolDepositToken	Public	-	-
poolRewardToken	Public	-	-
poolPending	Public	-	-
poolTokenApprove	Internal	Can Modify State	-
poolDeposit	Internal	Can Modify State	-
poolWithdraw	Internal	Can Modify State	-
poolClaim	Internal	Can Modify State	-

StrategyUtils			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setSConfig	External	Can Modify State	onlyOwner
makeDepositFee	External	Can Modify State	onlyOwner

makeFeeTransfer	Internal	Can Modify State	-
makeFeeTransferByValue	Internal	Can Modify State	-
makeWithdrawRewardFee	External	Can Modify State	onlyOwner
makeRefundFee	External	Can Modify State	onlyOwner
makeLiquidationFee	External	Can Modify State	onlyOwner
checkAddPoolLimit	External	-	-
checkDepositLimit	External	-	-
checkSlippageLimit	External	-	-
checkBorrowLimit	Public	-	-
checkBorrowGetHoldAmount	Internal	-	-
checkLiquidationLimit	External	-	-
makeRepay	External	Can Modify State	onlyOwner
getBorrowAmount	External	-	-
getBorrowAmount	Internal	-	-
getBorrowAmountInBaseToken	External	-	-
calcBorrowAmountInBaseToken	Public	-	-
transferFromAllToken	Public	Can Modify State	onlyOwner
transferFromToken	Public	Can Modify State	onlyOwner
optimalDepositAmount	Public	-	-
_optimalDepositA	Internal	-	-
getLPToken2TokenAmount	Public	-	-
getAmountOut	Public	-	-
getAmountIn	Public	-	-
getTokenOut	Public	Can Modify State	onlyOwner
getTokenIn	Public	Can Modify State	onlyOwner
getTokenInTo	Internal	Can Modify State	-
getMdexExtraReward	Public	Can Modify State	-

TenBankHall			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setBlacklist	External	Can Modify State	onlyOwner
setEmergencyEnabled	External	Can Modify State	onlyOwner
boxesLength	External	-	-

addBox	External	Can Modify State	onlyOwner
setBoxListed	External	Can Modify State	onlyOwner
strategyInfoLength	External	-	-
strategyIsListed	External	-	-
setStrategyListed	External	Can Modify State	onlyOwner
addStrategy	External	Can Modify State	onlyOwner
depositLPToken	Public	Can Modify State	nonReentrant
deposit	Public	Can Modify State	nonReentrant
withdrawLPToken	External	Can Modify State	nonReentrant
withdraw	External	Can Modify State	nonReentrant
emergencyWithdraw	External	Can Modify State	nonReentrant
liquidation	External	Can Modify State	nonReentrant
getBorrowAmount	External	-	-
getDepositAmount	External	-	-
makeBorrowFrom	External	Can Modify State	-
<Receive Ether>	External	Payable	-

4.2 合约信息

合约已经部署到 Heco 主网上，如下是已部署的合约名称及合约地址。

Contract Name	Contract Address
BOOTimelock	0xA59FFeF22b29a08D1bB0f39Bb211cefB163be2A4
BOOTimelock	0x79F17bd4a2DD6363aF0c752ba8c024304934fbA9
BOOPools	0xBa92b862ac310D42A8a3DE613dcE917d0d63D98c
ActionPools	0xf80af22dfE727842110AE295a08CDc9b4344430F
SafeBoxFilDa	0x8aee98bC67777d220bD5DBE2d3ECb22d765dCD91
SafeBoxFilDa	0xDD51428f162dcd92264b510D05B7c8bD276416Ba
SafeBoxFilDaETH	0x53A83C2d5D3725dAe285EC85B58dD564d586F6b7
SafeBoxFilDa	0x0e908182AA6989be3Fe452DcF625127873f9231e
SafeBoxFilDa	0x485E75ed3083CC1C3016D08eA049538b24094620
StrategyConfig	0x89F0AD04A3AA20187DD3777Bef7Acb66D036Da14
StrategyMDex	0x9AF4fa09C600598256fCF29DE1ca241A1677d4E1
TenBankHall	0xa61A4F9275eF62d2C076B0933F8A9418CeC8c670
BuybackBooToken	0xc1116948C1b3Befee21a05120dDa6a2e4dcE98f2
TokenOracle	0xb6Cd3fD256f357AD25dfF3520aE5256d27F26158
PriceCheckerLPToken	0x0b129Fe80e1b1FB00C60E14c312E99FCAEF01359

TenMath	0xC818D8a1860B73a70897448Af0667C994005233E
TenMath	0x5383912D31c7F250EcF0466E4552c1d7188ad5a4
StrategyUtils	0x10Ff27409928d8A6Ce2EB07f010c03e3AB4EdE2d

4.3 代码审计

4.3.1 严重漏洞

4.3.1.1 闪电贷攻击风险

计算 LP 价格的时候采用的是根据 LP 可以兑换的单边 token 数量乘 2 来进行计算的，如果攻击者者采用大资金或者使用闪电贷来改变池子中 token 的数量，此时采用单边 token 数量乘 2 会导致 LP 的价格被攻击者操控，从而进行套利。

- contracts/strategies/StrategyUtils.sol

```
function getLPToken2TokenAmount(address _lpToken, address _baseToken, uint256 _lpTokenAmount)
        public view returns (uint256 amount) {
    (uint256 a, uint256 b, ) = IMdexPair(_lpToken).getReserves();
    address token0 = IMdexPair(_lpToken).token0();
    address token1 = IMdexPair(_lpToken).token1();
    if(token0 == _baseToken) {
        amount = _lpTokenAmount.mul(a).div(ERC20(_lpToken).totalSupply()).mul(2);
    }else if(token1 == _baseToken) {
        amount = _lpTokenAmount.mul(b).div(ERC20(_lpToken).totalSupply()).mul(2);
    }
    else{
        require(false, 'unsupport baseToken not in pairs');
    }
}
```

makeExtraRewards 函数会调用 buyback 函数，这里存在通过闪电贷操纵代币回购价格的问题。

- contracts/strategies/StrategyMDex.sol

```
function makeExtraRewards() external {
    if(address(buyback) == address(0)) {
        return ;
    }
```

```
        (address mdxToken, uint256 value) = utils.getMdexExtraReward();
        uint256 fee = value.mul(3e6).div(1e9);
        IERC20(mdxToken).transfer(msg.sender, fee);
        IERC20(mdxToken).approve(address(buyback), value.sub(fee));
        buyback.buyback(mdxToken, value.sub(fee));
    }
```

- contracts/utils/BuybackBooToken.sol

```
function buyback(address _token, uint256 _value) public override returns (uint256 value) {
        uint256 decimals = uint256(ERC20(_token).decimals());
        if(_value < (10**decimals.div(4))) {
            return 0;
        }

        ......
        return buybackIn(_token, path, _value);
    }

    function buybackIn(address _token, address[] memory _path, uint256 _value) internal returns (uint256 value) {
        IERC20(_token).approve(address(router), _value);
        uint256[] memory result = router.swapExactTokensForTokens(_value, 0, _path, address(this), block.timestamp.add(60));
        if(result.length == 0) {
            return 0;
        }
        uint256 valueOut = TenMath.min(result[result.length-1],
                                       IERC20(booToken).balanceOf(address(this)));
        burnSource[_token] = burnSource[_token].add(_value);
        burnAmount[_token] = burnAmount[_token].add(valueOut);
        IERC20(booToken).transfer(lockedAddr, valueOut);
    }
```

修复状态：该问题在 commit: 1d9fc8692cae442b94ec091e866ee5194e0dd3aa 版本代码中进行了修复。

4.3.2 高危漏洞

4.3.2.1 返回值检查缺失

safeTokenTransfer 没有对转账的返回值进行判断，当 _token 采用类似 if ... else... 的写法会有假充值问

题。建议对_token.transfer(_to, value);的返回值进行判断，并且检查外部_token 合约的安全性。

- contracts/pools/ActionPools.sol，contracts/pools/ActionCompPools.sol

```
function safeTokenTransfer(IERC20 _token, address _to, uint256 _amount) internal returns (uint256 value) {
        uint256 balance = _token.balanceOf(address(this));
        value = _amount > balance ? balance : _amount;
        if ( value > 0 ) {
            _token.transfer(_to, value);
        }
    }
```

修复状态: 该问题已经在 commit:1d9fc8692cae442b94ec091e866ee5194e0dd3aa 版本中的代码修改为 safeTransfer，同时经过与项目方的沟通反馈，项目方在添加 Pool 的时候会保证 rewardToken 的安全性并且不会有假充值的问题。

4.3.3 中危漏洞

4.3.3.1 权限过大问题

Owner 用户添加 Pool 的权限，存在权限过大问题，Owner 角色可以自己添加 Pool 偷挖薅奖励。建议将 Owner 设置为 timelock 合约或采用治理的方式进行管理。

- contracts/pools/ActionPools.sol，contracts/pools/ActionCompPools.sol

```
function add(address _callFrom, uint256 _callId,
                address _rewardToken, uint256 _maxPerBlock) external onlyOwner {

        (address lpToken,, uint256 totalPoints,) =
                    ICompActionTrigger(_callFrom).getCATPoolInfo(_callId);
        require(lpToken != address(0) && totalPoints >= 0, 'pool not right');
        poolInfo.push(PoolInfo({
            callFrom: _callFrom,
            callId: _callId,
            rewardToken: IERC20(_rewardToken),
            rewardMaxPerBlock: _maxPerBlock,
            lastRewardBlock: block.number,
            lastRewardTotal: 0,
            lastRewardClosed: 0,
```

```
            poolTotalRewards: 0,
            autoUpdate: true,
            autoClaim: false
        }));

        eventSources[_callFrom] = true;
        poolIndex[_callFrom][_callId].push(poolInfo.length.sub(1));

        emit AddPool(poolInfo.length.sub(1), _callFrom, _callId, _rewardToken, _maxPerBlock);
    }
```

Owner 可以修改 poolInfo 的参数，这会影响挖矿的奖励，存在权限过大问题，建议将 Owner 权限移交给 timelock 合约或社区治理合约，并使用事件对更改操作进行记录，便于社区用户进行审查。

- contracts/pools/ActionPools.sol，contracts/pools/ActionCompPools.sol

```
function setRewardMaxPerBlock(uint256 _pid, uint256 _maxPerBlock) external onlyOwner {
        poolInfo[_pid].rewardMaxPerBlock = _maxPerBlock;
        emit SetRewardMaxPerBlock(_pid, _maxPerBlock);
    }

    function setAutoUpdate(uint256 _pid, bool _set) external onlyOwner {
        poolInfo[_pid].autoUpdate = _set;
    }

    function setAutoClaim(uint256 _pid, bool _set) external onlyOwner {
        poolInfo[_pid].autoClaim = _set;
    }

    function setRewardRestricted(address _hacker, uint256 _rate) external onlyOwner {
        require(_rate <= 1e9, 'max is 1e9');
        rewardRestricted[_hacker] = _rate;
        emit SetRewardRestricted(_hacker, _rate);
    }

    function setBooDev(address _boodev) external {
        require(msg.sender == boodev, 'prev dev only');
        boodev = _boodev;
    }
```

Owner 拥有转走合约中资产的权限，存在权限过大的问题。经过与项目方沟通反馈，Owner 权限会移交到 StrategyMDex 合约，不会设置为普通地址。

- contracts/strategies/StrategyUtils.sol

```
function getMdexExtraReward() public virtual returns (address token, uint256 rewards) {
        IMdexHecoSwapPool swappool = IMdexHecoSwapPool(0x7373c42502874C88954bDd6D50b53061F018422e);
        token = swappool.mdx();
        uint256 uBalanceBefore = IERC20(token).balanceOf(address(this));
        swappool.takerWithdraw();
        uint256 uBalanceAfter = IERC20(token).balanceOf(address(this));
        rewards = uBalanceAfter.sub(uBalanceBefore);
        transferFromAllToken(address(this), msg.sender, token, token);
    }
```

- contracts/strategies/StrategyUtils.sol

```
function transferFromAllToken(address _from, address _to, address _token0, address _token1)
         public onlyOwner {

        transferFromToken(_from, _to, _token0);
        transferFromToken(_from, _to, _token1);
    }
```

修复状态: StrategyUtils 合约的权限过大问题已修复，StrategyUtils 合约由 StrategyMDex 合约创建，

StrategyUtils 合约的 Owner 是 0x9AF4fa09C600598256fCF29DE1ca241A1677d4E1 。

详情参考：

https://hecoinfo.com/address/0x9AF4fa09C600598256fCF29DE1ca241A1677d4E1#code

https://hecoinfo.com/address/0x10Ff27409928d8A6Ce2EB07f010c03e3AB4EdE2d#code

其它合约的权限过大问题暂未修复。

4.3.3.2 黑名单机制可以被绕过

Owner 可以设置黑名单列表，黑名单中的地址不能进行充值。

- contracts/TenBankHall.sol

```
function setBlacklist(address _account, bool _newset) external onlyOwner {
        blacklist[_account] = _newset;
        emit SetBlacklist(_account, _newset);
    }
```

但是这里的黑名单机制存在缺陷，因为黑名单的地址可以将资产转移给其他不在黑名单中的地址从而绕过检查。建议如果要采用黑名单的机制的话，要在入金和出金的函数中都要进行检查，并且需要快速识别定位黑客地址，并及时加入黑名单这样能提高黑客作恶的成本。

同理：contracts/pools/BOOPools.sol, contracts/safebox/SafeBoxCTokenETH.sol 合约中的黑名单机制也是如此。

- contracts/TenBankHall.sol

```
function depositLPToken(uint256 _sid, uint256 _amount, uint256 _bid, uint256 _bAmount, uint256 _desirePrice, uint256 _slippage)
        public nonReentrant returns (uint256 lpAmount) {
    require(strategyInfo[_sid].isListed, 'not listed');
    require(!blacklist[msg.sender], 'address in blacklist');

    address lpToken = strategyInfo[_sid].iLink.getPoolLpToken(strategyInfo[_sid].pid);
    IERC20(lpToken).safeTransferFrom(msg.sender, address(strategyInfo[_sid].iLink), _amount);

    address boxitem = address(0);
    if(_bAmount > 0) {
        boxitem = address(boxInfo[_bid]);
    }
    return strategyInfo[_sid].iLink.depositLPToken(strategyInfo[_sid].pid, msg.sender, boxitem, _bAmount, _desirePrice, _slippage);
  }

  function deposit(uint256 _sid, uint256[] memory _amount, uint256 _bid, uint256 _bAmount, uint256 _desirePrice, uint256 _slippage)
        public nonReentrant returns (uint256 lpAmount) {
    require(strategyInfo[_sid].isListed, 'not listed');
    require(!blacklist[msg.sender], 'address in blacklist');

    address[] memory collateralToken = strategyInfo[_sid].iLink.getPoolCollateralToken(strategyInfo[_sid].pid);
    require(collateralToken.length == _amount.length, '_amount length error');

    for(uint256 u = 0; u < collateralToken.length; u ++) {
        if(_amount[u] > 0) {
            IERC20(collateralToken[u]).safeTransferFrom(msg.sender, address(strategyInfo[_sid].iLink), _amount[u]);
        }
    }
```

```
        address boxitem = address(0);
        if(_bAmount > 0) {
            boxitem = address(boxInfo[_bid]);
        }
        return strategyInfo[_sid].iLink.deposit(strategyInfo[_sid].pid, msg.sender, boxitem, _bAmount, _desirePrice, _slippage);
    }
```

修复状态: 已修复,经过于项目方的沟通和反馈,该业务设计是用于避免恶意合约进行薅羊毛获利,在 commit: 1d9fc8692cae442b94ec091e866ee5194e0dd3aa 版本中的代码黑名单机制改为了仅可以添加合约地址到黑名单的地址列表中，并且限制黑名单中的地址进行入金和出金。

4.3.4 低危漏洞

4.3.4.1 业务逻辑不明确

使用了 3 个 if 来判断 reward 的数量，这里 pool.poolTotalRewards 和 balance 的大小判断会有多种情况，存在业务逻辑上的不明确的问题，建议要根据实际的业务需求明确判断 pool.poolTotalRewards 和 balance 两个变量的大小。

- contracts/pools/ActionCompPools.sol

```
function getBlocksReward(uint256 _pid, uint256 _from, uint256 _to) public view returns (uint256 value) {
        require(_from <= _to, 'getBlocksReward error');
        PoolInfo storage pool = poolInfo[_pid];
        value = pool.rewardMaxPerBlock.mul(_to.sub(_from));
        if( address(pool.rewardToken) == address(booToken)) {
            return value;
        }
        uint256 balance = pool.rewardToken.balanceOf(address(this));
        if( pool.lastRewardClosed > balance ||
            pool.lastRewardClosed > pool.poolTotalRewards) {
            return 0;
        }
        if( pool.lastRewardClosed.add(value) > balance) {
            value = balance.sub(pool.lastRewardClosed);
        }
        if( pool.lastRewardClosed.add(value) > pool.poolTotalRewards) {
            value = pool.poolTotalRewards.sub(pool.lastRewardClosed);
```

```
        }
    }
```

修复状态：该问题在 commit: 1d9fc8692cae442b94ec091e866ee5194e0dd3aa 版本代码中进行了修复。

4.3.4.2 三明治攻击风险

在调用 swapExactTokensForTokens 的时候没有进行滑点限制，amountOutMin 为 0，存在三明治攻击的风险，建议在调用 swapExactTokensForTokens 函数之前增加滑点的检查。

- contracts/strategies/StrategyUtils.sol

```
function getTokenInTo(address _toAddress, address _tokenIn, uint256 _amountIn, address _tokenOut)
            internal virtual returns (uint256 value) {
        if(_tokenIn == _tokenOut) {
            value = _amountIn;
            return value;
        }
        address[] memory path = new address[](2);
        path[0] = _tokenIn;
        path[1] = _tokenOut;
        uint256 amountOutMin = 0;
        IERC20(_tokenIn).approve(address(router), uint256(-1));
        require(IERC20(_tokenIn).balanceOf(address(this)) >= _amountIn, 'getTokenInTo not amount in');
        uint256[] memory result = router.swapExactTokensForTokens(_amountIn, amountOutMin, path, _toAddress,
block.timestamp.add(60));
        if(result.length == 0) {
            value = 0;
        } else {
            value = result[result.length-1];
        }
    }
```

- contracts/utils/BuybackBooToken.sol

```
function buybackIn(address _token, address[] memory _path, uint256 _value) internal returns (uint256 value) {
        IERC20(_token).approve(address(router), _value);
        uint256[] memory result = router.swapExactTokensForTokens(_value, 0, _path, address(this), block.timestamp.add(60));
        if(result.length == 0) {
            return 0;
```

```
        }
        uint256 valueOut = TenMath.min(result[result.length-1],
                                IERC20(booToken).balanceOf(address(this)));
        burnSource[_token] = burnSource[_token].add(_value);
        burnAmount[_token] = burnAmount[_token].add(valueOut);
        IERC20(booToken).transfer(lockedAddr, valueOut);
    }
```

- contracts/strategies/StrategyMDex.sol

```
function updatePool(uint256 _pid) public override {
        PoolInfo storage pool = poolInfo[_pid];
        if(pool.lastRewardsBlock == block.number ||
            pool.totalLPReinvest <= 0) {
            pool.lastRewardsBlock = block.number;
            return ;
        }
.....
        uint256 newRewardBase = utils.getTokenIn(rewardToken, newRewards, pool.baseToken);
.....
    }
```

修复状态：该问题在 commit: 1d9fc8692cae442b94ec091e866ee5194e0dd3aa 版本代码中进行了修复。

4.3.4.3 交易重排攻击风险

getAmountOut 和 getAmountIn 是在 添加和移除流动性的时候要用到的，代码中没有进行滑点的检查，存在交易重排的风险，攻击者可以在用户添加和移除流动性之前构造一个失衡的比例，让用户以错误的比例进行添加或移除流动性，等用户执行完操作后，攻击者再将比例还原获利。

参考：https://www.odaily.com/post/5162888

- contracts/strategies/StrategyUtils.sol

```
function getAmountOut(address _tokenIn, address _tokenOut, uint256 _amountOut)
        public virtual view returns (uint256) {
    if(_tokenIn == _tokenOut) {
        return _amountOut;
    }
    address[] memory path = new address[](2);
```

```
        path[0] = _tokenIn;
        path[1] = _tokenOut;
        uint256[] memory result = router.getAmountsIn(_amountOut, path);
        if(result.length == 0) {
            return 0;
        }
        return result[0];
    }

    function getAmountIn(address _tokenIn, uint256 _amountIn, address _tokenOut)
            public virtual view returns (uint256) {
        if(_tokenIn == _tokenOut) {
            return _amountIn;
        }
        address[] memory path = new address[](2);
        path[0] = _tokenIn;
        path[1] = _tokenOut;
        uint256[] memory result = router.getAmountsOut(_amountIn, path);
        if(result.length == 0) {
            return 0;
        }
        return result[result.length-1];
    }
```

修复状态：该问题在 commit: 1d9fc8692cae442b94ec091e866ee5194e0dd3aa 版本代码中通过滑点检查进行了修复。

4.3.5 增强建议

4.3.5.1 存在冗余代码

合约默认是不接收 ETH 的，所以这里使用了 receive 和 revert 是属于冗余的代码。建议可以删除冗余代码来节省部署合约的 Gas 消耗。

- contracts/pools/ActionPools.sol，contracts/pools/ActionCompPools.sol

```
    receive() external payable {
        revert();
```

```
}
```

修复状态：经过与项目方的沟通反馈，该问题不影响实际的业务，暂不修复。

4.3.5.2 Gas 优化问题

getBlocksReward 函数先执行了`pool.rewardToken.balanceOf(address(this));`外部合约调用，然后再进行 if 的判断，这两部分代码没有依赖关系，建议先判断 if 语句`if( address(pool.rewardToken) == address(booToken))` 再执行外部调用，`uint256 balance = pool.rewardToken.balanceOf(address(this)); `这样可以优化 Gas,避免先获取了 balance 然后 if 语句 return 导致白白消耗外部调用的 Gas。

- contracts/pools/ActionPools.sol

```
function getBlocksReward(uint256 _pid, uint256 _from, uint256 _to) public view returns (uint256 value) {
        require(_from <= _to, 'getBlocksReward error');
        PoolInfo storage pool = poolInfo[_pid];
        uint256 balance = pool.rewardToken.balanceOf(address(this));
        value = pool.rewardMaxPerBlock.mul(_to.sub(_from));
        if( address(pool.rewardToken) == address(booToken)) {
            return value;
        }
        if( pool.lastRewardClosed > balance
            || pool.lastRewardClosed > pool.poolTotalRewards) {
                // require(pool.lastRewardClosed > balance, 'rewardClosed > balance');
                // require(pool.lastRewardClosed > pool.poolTotalRewards, 'rewardClosed > poolTotalRewards');
                return 0;
        }
```

修复状态：该问题在 commit: 1d9fc8692cae442b94ec091e866ee5194e0dd3aa 版本中的代码进行了修复。

4.3.5.3 编码规范优化建议

项目中大部分部分 Internal 的函数是采用了以下划线(_)开头进行函数命名的，有一部分函数没有采用下划线

(_)开头的方式进行命名，这部分函数容易在开发上造成可见性的混淆属于编码规范的优化建议，建议可以采用_下划线开头来命名，避免开发上内外部函数的混淆。

Contract Name	Function Name	Visibility
ActionCompPools	deposit	Internal
ActionCompPools	withdraw	Internal
ActionCompPools	emergencyWithdraw	Internal
ActionCompPools	safeTokenTransfer	Internal
ActionPools	deposit	Internal
ActionPools	withdraw	Internal
ActionPools	emergencyWithdraw	Internal
ActionPools	safesub	Internal
ActionPools	safeTokenTransfer	Internal
SafeBoxCTokenImpl	ctokenDeposit	Internal
SafeBoxCTokenImpl	ctokenWithdraw	Internal
SafeBoxCTokenImpl	ctokenClaim	Internal
SafeBoxCTokenImpl	ctokenBorrow	Internal
SafeBoxCTokenImpl	ctokenRepayBorrow	Internal
SafeBoxCToken	tokenSafeTransfer	Internal
SafeBoxCTokenETH	tokenSafeTransfer	Internal
SafeBoxCTokenImplETH	ctokenDeposit	Internal
SafeBoxCTokenImplETH	ctokenWithdraw	Internal
SafeBoxCTokenImplETH	ctokenClaim	Internal
SafeBoxCTokenImplETH	ctokenBorrow	Internal
SafeBoxCTokenImplETH	ctokenRepayBorrow	Internal
SafeBoxFilDa	checkFildaPool	Internal
SafeBoxFilDaETH	checkFildaPool	Internal
StrategyConfig	checkBorrowAndLiquidation	Internal
StrategyMDexPools	poolTokenApprove	Internal
StrategyMDexPools	poolDeposit	Internal
StrategyMDexPools	poolWithdraw	Internal
StrategyMDexPools	poolClaim	Internal
StrategyMDex	getTokenBalance_this	Internal
StrategyMDex	makeBorrowBaseToken	Internal

StrategyMDex	makeBalanceOptimalLiquidity	Internal
StrategyMDex	makeBalanceOptimalLiquidityByAmount	Internal
StrategyMDex	makeLiquidityAndDeposit	Internal
StrategyMDex	makeLiquidityAndDepositByAmount	Internal
StrategyMDex	makeWithdrawRemoveLiquidity	Internal
StrategyMDexPools	poolTokenApprove	Internal
StrategyMDexPools	poolDeposit	Internal
StrategyMDexPools	poolWithdraw	Internal
StrategyMDexPools	poolClaim	Internal
StrategyUtils	makeFeeTransfer	Internal
StrategyUtils	makeFeeTransferByValue	Internal
StrategyUtils	checkBorrowGetHoldAmount	Internal
StrategyUtils	getBorrowAmount	Internal
StrategyUtils	getTokenInTo	Internal

修复状态: 暂未修复。

4.3.5.4 boodev 角色增强建议

boodev 角色可以通过 setBooDev 函数更改开发团队奖励的地址 ，建议将 boodev 角色设置为多签地址，并时候用事件记录更改地址的操作，结合链下对事件操作进行监控，避免地址私钥被盗或遗失导致开发者的奖励丢失。

- ccontracts/pools/ActionPools.sol
- ontracts/pools/ActionCompPools.sol

```
function setBooDev(address _boodev) external {
        require(msg.sender == boodev, 'prev dev only');
        boodev = _boodev;
    }
```

修复状态: 暂未修复。

4.3.5.5 函数可见性设置错误

如下合约对应的函数可见性应该是 internal 的但是设置成了 public，建议将如下的函数可见性修改为 internal。

- contracts/safebox/SafeBoxCToken.sol
- contracts/safebox/SafeBoxCTokenETH.sol

```
function _getBorrowFactor(uint256 supplyAmount) public virtual view returns (uint256)
function _update() public
```

修复状态：该问题在 commit: 1d9fc8692cae442b94ec091e866ee5194e0dd3aa 版本中的代码已进行了修复。

4.3.5.6 存在逻辑判断上的冗余代码

uint 类型的变量的取值范围是不会小于 0 的，但是代码上有多处对 uint 类型的变量小于 0 的情况进行判断，属于逻辑判断上的冗余代码。建议开发人员排查代码中其他类似的写法并删除逻辑判断上冗余的代码。

- contracts/strategies/StrategyConfig.sol

```
function checkBorrowAndLiquidation(address _strategy, uint256 _poolid) internal returns (bool bok) {
        uint256 v = getBorrowFactor(_strategy, _poolid);
        if(v <= 0) {
            return true;
        }
        ......
    }
```

- contracts/strategies/StrategyConfig.sol

```
function getLiquidationFactor(address _strategy, uint256 _poolid) public override view returns (uint256 value) {
        value = liquidationFactor[_strategy][_poolid];
        if(value <= 0) {
            value = 8e8;   // 80% for default , 100% will be liquidation
        }
    }
```

- contracts/strategies/StrategyMDex.sol

```
function pendingLPAmount(uint256 _pid, address _account) public override view returns (uint256 value) {
        PoolInfo storage pool = poolInfo[_pid];
        if(pool.totalPoints <= 0) {
            return 0;
        }
.....
    }
```

修复状态：暂未修复。

4.3.5.7 代币兼容性问题

目前项目的设计不兼容通缩和通胀型代币对应的 LP Token 进行挖矿和杠杠借贷，建议对接的时候要对代币进行审查，确保与 BoosterProtocol 项目是兼容的。

修复状态：经过与项目方沟通反馈，项目方在对接 Token 的时候会评估 Token 与 BoosterProtocol 项目是否兼容，仅对接标准的 ERC20 Token 对应的 LP 挖矿业务。

5. 审计结果

5.1 结论

审计结果：中风险

审计编号：0X002104130002

审计日期：2021 年 04 月 13 日

审计团队：慢雾安全团队

总结：慢雾安全团队采用人工结合内部工具对代码进行分析，审计期间发现了 1 个严重漏洞，1 个高危漏洞，2 个中危漏洞，3 个低危漏洞，7 个增强建议。如下是对修复的状态进行汇总：

(1). 严重漏洞，闪电贷攻击风险已进行了修复。

(2). 高危漏洞，返回值检查缺失的风险经过沟通反馈后项目方会保证 rewardToken 不会有假充值的问题。

(3). 中危漏洞权限过大问题由于目前已部署到主网，但是还未将 Owner 权限移交给 timelock，存在权限过大的问题。黑名单机制可被绕过的问题已进行了修复。

(4). 低危漏洞，业务逻辑不明确的问题已进行了修复，三明治攻击风险已进行了修复，交易重排攻击风险已进行了修复。

(5). 增强建议，忽略存在冗余代码的问题，修复了 Gas 优化的增强建议，编码规范优化暂未修复，由于暂未部署到主网 boodev 角色暂未使用多签合约，修复了函数可见性设置错误的问题，逻辑判断上的冗余代码暂未修复，逻辑判断上的冗余代码暂未处理，代币兼容性问题经过与项目方的沟通反馈，项目仅对接标准的 ERC20 Token 对应的 LP 挖矿业务。

6. 声明

慢雾科技
slow mist

官方网址

www.slowmist.com

电子邮箱

team@slowmist.com

微信公众号