Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

2 Audit Methodology

3 Project Overview

3.1 Project Introduction

3.2 Vulnerability Information

4 Code Overview

4.1 Contracts Description

4.2 Visibility Description

4.3 Vulnerability Summary

5 Audit Result

6 Statement

1 Executive Summary

On 2022.09.28, the SlowMist security team received the Cada finance team's security audit application for Cada finance, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project team should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.

Level	Description
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.

Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

Serial Number	Audit Class	Audit Subclass
1	Overflow Audit	-
2	Reentrancy Attack Audit	-
3	Replay Attack Audit	-
4	Flashloan Attack Audit	-
5	Race Conditions Audit	Reordering Attack Audit
6	Permission Vulnerability Audit	Access Control Audit
		Excessive Authority Audit

Serial Number	Audit Class	Audit Subclass
7	Security Design Audit	External Module Safe Use Audit
		Compiler Version Security Audit
		Hard-coded Address Security Audit
		Fallback Function Safe Use Audit
		Show Coding Security Audit
		Function Return Value Security Audit
		External Call Function Security Audit
		Block data Dependence Security Audit
		tx.origin Authentication Security Audit
8	Denial of Service Audit	-
9	Gas Optimization Audit	-
10	Design Logic Audit	-
11	Variable Coverage Vulnerability Audit	-
12	"False Top-up" Vulnerability Audit	-
13	Scoping and Declarations Audit	-
14	Malicious Event Log Audit	-
15	Arithmetic Accuracy Deviation Audit	-
16	Uninitialized Storage Pointer Audit	-

3 Project Overview

3.1 Project Introduction

Audit Version

https://github.com/cadafinance/cadafinance-contract commit:

172ed16cebc62a56c63248e4b4a20f1bc5413c0f

Fixed Version

https://github.com/cadafinance/cadafinance-contract commit:

552199b084b7a4178193bdadl78fd83c4167f4ba

3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	Event logging logic error	Design Logic Audit	Suggestion	Fixed
N2	Missing event record	Others	Suggestion	Fixed
N3	Gas optimization	Gas Optimization Audit	Suggestion	Fixed
N4	Business logic is not clear	Design Logic Audit	Low	Fixed
N5	Business logic error	Design Logic Audit	Suggestion	Ignored
N6	mint is not authenticated	Authority Control Vulnerability	High	Fixed
N7	Preemptive initialization issues	Others	Suggestion	Confirmed
N8	Lack of functions to withdraw excess token assets	Others	Suggestion	Confirmed

NO	Title	Category	Level	Status
N9	allowance can't reset	Others	Suggestion	Confirmed

4 Code Overview

4.1 Contracts Description

The main network address of the contract is as follows:

The code was deployed to the mainnet.

4.2 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

GaugeController			
Function Name	Visibility	Mutability	Modifiers
__GaugeController_init	External	Can Modify State	initializer
gaugeTypes	External	-	-
_getTypeWeight	Internal	Can Modify State	-
_getSum	Internal	Can Modify State	-
_getTotal	Internal	Can Modify State	-
_getWeight	Internal	Can Modify State	-
addGauge	External	Can Modify State	onlyOwner
checkpoint	External	Can Modify State	-

GaugeController			
checkpointGauge	External	Can Modify State	-
_gaugeRelativeWeight	Internal	-	-
gaugeRelativeWeight	External	-	-
gaugeRelativeWeightWrite	External	Can Modify State	-
_changeTypeWeight	Internal	Can Modify State	-
addType	External	Can Modify State	onlyOwner
changeTypeWeight	External	Can Modify State	onlyOwner
_changeGaugeWeight	Internal	Can Modify State	-
changeGaugeWeight	External	Can Modify State	onlyOwner
voteForGaugeWeights	External	Can Modify State	-
getGaugeWeight	External	-	-
getTypeWeight	External	-	-
getTotalWeight	External	-	-
getWeightsSumPerType	External	-	-
getGaugeList	External	-	-

LiquidityGauge			
Function Name	Visibility	Mutability	Modifiers
__LiquidityGauge_init	External	Can Modify State	initializer
integrateCheckpoint	External	-	-

LiquidityGauge			
_updateLiquidityLimit	Internal	Can Modify State	-
_checkpointRewards	Internal	Can Modify State	-
_checkpoint	Internal	Can Modify State	-
userCheckpoint	External	Can Modify State	-
claimableTokens	External	Can Modify State	-
claimedReward	External	-	-
claimableReward	External	-	-
setRewardsReceiver	External	Can Modify State	-
claimRewards	External	Can Modify State	nonReentrant
kick	External	Can Modify State	-
deposit	External	Can Modify State	nonReentrant
withdraw	External	Can Modify State	nonReentrant
_transfer2	Internal	Can Modify State	-
transfer	Public	Can Modify State	nonReentrant
addReward	External	Can Modify State	onlyOwner
setRewardDistributor	External	Can Modify State	-
depositRewardToken	External	Can Modify State	nonReentrant
setKilled	External	Can Modify State	onlyOwner

Minter			
Function Name	Visibility	Mutability	Modifiers
__Minter_init	External	Can Modify State	initializer
_mintFor	Internal	Can Modify State	-
mint	External	Can Modify State	nonReentrant
mintMany	External	Can Modify State	nonReentrant
mintFor	External	Can Modify State	nonReentrant
toggleApproveMint	External	Can Modify State	-

Allowlist			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getPoolAccountLimit	External	-	-
getPoolCap	External	-	-
isAccountVerified	External	-	-
verifyAddress	External	Can Modify State	-
setPoolAccountLimit	External	Can Modify State	onlyOwner
setPoolCap	External	Can Modify State	onlyOwner
updateMerkleRoot	External	Can Modify State	onlyOwner

LPTokenGuarded			
Function Name	Visibility	Mutability	Modifiers

LPTokenGuarded			
<Constructor>	Public	Can Modify State	BEP20
mint	External	Can Modify State	onlyOwner
_beforeTokenTransfer	Internal	Can Modify State	-

OwnerPausable			
Function Name	Visibility	Mutability	Modifiers
pause	External	Can Modify State	onlyOwner
unpause	External	Can Modify State	onlyOwner

SwapGuarded			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	OwnerPausable ReentrancyGuard
getA	External	-	-
getAPrecise	External	-	-
getToken	Public	-	-
getTokenIndex	External	-	-
getAllowlist	External	-	-
getDepositTimestamp	External	-	-
getTokenBalance	External	-	-
getVirtualPrice	External	-	-
calculateSwap	External	-	-

SwapGuarded			
calculateTokenAmount	External	-	-
calculateRemoveLiquidity	External	-	-
calculateRemoveLiquidityOneToken	External	-	-
calculateCurrentWithdrawFee	External	-	-
getAdminBalance	External	-	-
swap	External	Can Modify State	nonReentrant whenNotPaused deadlineCheck
addLiquidity	External	Can Modify State	nonReentrant whenNotPaused deadlineCheck
removeLiquidity	External	Can Modify State	nonReentrant deadlineCheck
removeLiquidityOneToken	External	Can Modify State	nonReentrant whenNotPaused deadlineCheck
removeLiquidityImbalance	External	Can Modify State	nonReentrant whenNotPaused deadlineCheck
updateUserWithdrawFee	External	Can Modify State	-
withdrawAdminFees	External	Can Modify State	onlyOwner
setAdminFee	External	Can Modify State	onlyOwner
setSwapFee	External	Can Modify State	onlyOwner
setDefaultWithdrawFee	External	Can Modify State	onlyOwner
rampA	External	Can Modify State	onlyOwner
stopRampA	External	Can Modify State	onlyOwner

SwapGuarded			
disableGuard	External	Can Modify State	onlyOwner
isGuarded	External	-	-

SwapUtilsGuarded			
Function Name	Visibility	Mutability	Modifiers
getA	External	-	-
_getA	Internal	-	-
getAPrecise	External	-	-
_getAPrecise	Internal	-	-
getDepositTimestamp	External	-	-
calculateWithdrawOneToken	Public	-	-
calculateWithdrawOneTokenDY	Internal	-	-
getYD	Internal	-	-
getD	Internal	-	-
getD	Internal	-	-
_xp	Internal	-	-
_xp	Internal	-	-
_xp	Internal	-	-
getVirtualPrice	External	-	-
getY	Internal	-	-

SwapUtilsGuarded			
calculateSwap	External	-	-
_calculateSwap	Internal	-	-
calculateRemoveLiquidity	External	-	-
_calculateRemoveLiquidity	Internal	-	-
calculateCurrentWithdrawFee	Public	-	-
calculateTokenAmount	External	-	-
getAdminBalance	External	-	-
_feePerToken	Internal	-	-
swap	External	Can Modify State	-
addLiquidity	External	Can Modify State	-
updateUserWithdrawFee	External	Can Modify State	-
_updateUserWithdrawFee	Internal	Can Modify State	-
removeLiquidity	External	Can Modify State	-
removeLiquidityOneToken	External	Can Modify State	-
removeLiquidityImbalance	Public	Can Modify State	-
withdrawAdminFees	External	Can Modify State	-
setAdminFee	External	Can Modify State	-
setSwapFee	External	Can Modify State	-
setDefaultWithdrawFee	External	Can Modify State	-
rampA	External	Can Modify State	-

SwapUtilsGuarded			
stopRampA	External	Can Modify State	-

BaseBoringBatchable			
Function Name	Visibility	Mutability	Modifiers
_getRevertMsg	Internal	-	-
batch	External	Payable	-

Cloner			
Function Name	Visibility	Mutability	Modifiers
clone	External	Can Modify State	-

GenericBEP20			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	BEP20
mint	External	Can Modify State	-
decimals	Public	-	-

MetaSwap			
Function Name	Visibility	Mutability	Modifiers
getVirtualPrice	External	-	-
calculateSwap	External	-	-
calculateSwapUnderlying	External	-	-

MetaSwap			
calculateTokenAmount	External	-	-
calculateRemoveLiquidityOneToken	External	-	-
initialize	Public	Can Modify State	initializer
initializeMetaSwap	External	Can Modify State	initializer
swap	External	Can Modify State	nonReentrant whenNotPaused deadlineCheck
swapUnderlying	External	Can Modify State	nonReentrant whenNotPaused deadlineCheck
addLiquidity	External	Can Modify State	nonReentrant whenNotPaused deadlineCheck
removeLiquidityOneToken	External	Can Modify State	nonReentrant whenNotPaused deadlineCheck
removeLiquidityImbalance	External	Can Modify State	nonReentrant whenNotPaused deadlineCheck

MetaSwapDeposit			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer
swap	External	Can Modify State	nonReentrant
addLiquidity	External	Can Modify State	nonReentrant
removeLiquidity	External	Can Modify State	nonReentrant
removeLiquidityOneToken	External	Can Modify State	nonReentrant
removeLiquidityImbalance	External	Can Modify State	nonReentrant

MetaSwapDeposit			
calculateTokenAmount	External	-	-
calculateRemoveLiquidity	External	-	-
calculateRemoveLiquidityOneToken	External	-	-
getToken	External	-	-
calculateSwap	External	-	-

AmplificationUtils			
Function Name	Visibility	Mutability	Modifiers
getA	External	-	-
getAPrecise	External	-	-
_getAPrecise	Internal	-	-
rampA	External	Can Modify State	-
stopRampA	External	Can Modify State	-

LPToken			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	initializer
mint	External	Can Modify State	onlyOwner
_beforeTokenTransfer	Internal	Can Modify State	-

OwnerPausableUpgradeable

OwnerPausableUpgradeable			
Function Name	Visibility	Mutability	Modifiers
__OwnerPausable_init	Internal	Can Modify State	initializer
pause	External	Can Modify State	onlyOwner
unpause	External	Can Modify State	onlyOwner

RetroactiveVesting			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
verifyAndClaimReward	External	Can Modify State	-
claimReward	External	Can Modify State	-
_claimReward	Internal	Can Modify State	-
vestedAmount	External	-	-
_vestedAmount	Internal	-	-

SimpleGovernance			
Function Name	Visibility	Mutability	Modifiers
changeGovernance	External	Can Modify State	onlyGovernance
acceptGovernance	External	Can Modify State	-

cada			
Function Name	Visibility	Mutability	Modifiers

CADA			
<Constructor>	Public	Can Modify State	BEP20 BEP20Permit
setMinter	External	Can Modify State	onlyGovernance
_updateMiningParameters	Private	Can Modify State	-
updateMiningParameters	External	Can Modify State	-
startEpochTimeWrite	External	Can Modify State	-
futureEpochTimeWrite	External	Can Modify State	-
_availableSupply	Internal	-	-
availableSupply	External	-	-
mintableInTimeframe	External	-	-
mint	External	Can Modify State	-
burn	External	Can Modify State	-
deployNewVestingContract	Public	Can Modify State	onlyGovernance
enableTransfer	External	Can Modify State	-
addToAllowedList	External	Can Modify State	onlyGovernance
removeFromAllowedList	External	Can Modify State	onlyGovernance
_beforeTokenTransfer	Internal	Can Modify State	-
rescueTokens	External	Can Modify State	onlyGovernance

StakeableTokenWrapper			
Function Name	Visibility	Mutability	Modifiers

StakeableTokenWrapper			
<Constructor>	Public	Can Modify State	-
balanceOf	External	-	-
stake	External	Can Modify State	-
withdraw	External	Can Modify State	-

Swap			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
getA	External	-	-
getAPrecise	External	-	-
getToken	Public	-	-
getTokenIndex	Public	-	-
getTokenBalance	External	-	-
getVirtualPrice	External	-	-
calculateSwap	External	-	-
calculateTokenAmount	External	-	-
calculateRemoveLiquidity	External	-	-
calculateRemoveLiquidityOneToken	External	-	-
getAdminBalance	External	-	-
swap	External	Can Modify State	nonReentrant whenNotPaused deadlineCheck

Swap			
addLiquidity	External	Can Modify State	nonReentrant whenNotPaused deadlineCheck
removeLiquidity	External	Can Modify State	nonReentrant deadlineCheck
removeLiquidityOneToken	External	Can Modify State	nonReentrant whenNotPaused deadlineCheck
removeLiquidityImbalance	External	Can Modify State	nonReentrant whenNotPaused deadlineCheck
withdrawAdminFees	External	Can Modify State	onlyOwner
setAdminFee	External	Can Modify State	onlyOwner
setSwapFee	External	Can Modify State	onlyOwner
rampA	External	Can Modify State	onlyOwner
stopRampA	External	Can Modify State	onlyOwner

SwapDeployer			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
clone	External	Can Modify State	-
_clone	Internal	Can Modify State	-
deploy	External	Can Modify State	-
deployMetaSwap	External	Can Modify State	-

SwapMigrator

SwapMigrator			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
migrateUSDPool	External	Can Modify State	-
rescue	External	Can Modify State	-

SwapUtils			
Function Name	Visibility	Mutability	Modifiers
_getAPrecise	Internal	-	-
calculateWithdrawOneToken	External	-	-
_calculateWithdrawOneToken	Internal	-	-
calculateWithdrawOneTokenDY	Internal	-	-
getYD	Internal	-	-
getD	Internal	-	-
_xp	Internal	-	-
_xp	Internal	-	-
getVirtualPrice	External	-	-
getY	Internal	-	-
calculateSwap	External	-	-
_calculateSwap	Internal	-	-
calculateRemoveLiquidity	External	-	-

SwapUtils			
_calculateRemoveLiquidity	Internal	-	-
calculateTokenAmount	External	-	-
getAdminBalance	External	-	-
_feePerToken	Internal	-	-
swap	External	Can Modify State	-
addLiquidity	External	Can Modify State	-
removeLiquidity	External	Can Modify State	-
removeLiquidityOneToken	External	Can Modify State	-
removeLiquidityImbalance	Public	Can Modify State	-
withdrawAdminFees	External	Can Modify State	-
setAdminFee	External	Can Modify State	-
setSwapFee	External	Can Modify State	-

Vesting			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
release	External	Can Modify State	-
vestedAmount	Public	-	-
changeBeneficiary	External	Can Modify State	onlyGovernance

Vesting			
governance	Public	-	-

VotingEscrow			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getLastUserSlope	External	-	-
userPointHistory__ts	External	-	-
locked__end	External	-	-
_checkpoint	Internal	Can Modify State	-
_depositFor	Internal	Can Modify State	-
checkpoint	External	Can Modify State	-
depositFor	External	Can Modify State	nonReentrant
createLock	External	Can Modify State	nonReentrant
increaseAmount	External	Can Modify State	nonReentrant
increaseUnlockTime	External	Can Modify State	nonReentrant
withdraw	External	Can Modify State	nonReentrant
findBlockEpoch	Internal	-	-
balanceOf	Public	-	-
balanceOf	Public	-	-
balanceOfAt	External	-	-

VotingEscrow			
supplyAt	Internal	-	-
totalSupply	Public	-	-
totalSupply	Public	-	-
totalSupplyAt	External	-	-
emergencyWithdraw	External	Can Modify State	nonReentrant
_penalize	Internal	Can Modify State	-
setPenaltyCollector	External	Can Modify State	onlyOwner
setEarlyWithdrawPenaltyRate	External	Can Modify State	onlyOwner

4.3 Vulnerability Summary

[N1] [Suggestion] Event logging logic error

Category: Design Logic Audit

Content

The AddType event is placed in the if statement, but the type will be added when the addType function is executed.

The code of the event record should be placed outside the if statement.

- contracts/farming/GaugeController.sol#L412

```
function addType(string memory _name, uint256 _weight) external onlyOwner {
    uint256 typeId = nGaugeTypes;
    gaugeTypeNames[typeId] = _name;
    nGaugeTypes = typeId + 1;
    if (_weight != 0) {
        _changeTypeWeight(typeId, _weight);
        emit AddType(_name, typeId);
    }
```

```
    }
```

Solution

It is recommended to put the code for the AddType event logging outside the if statement.

Status

Fixed

[N2] [Suggestion] Missing event record

Category: Others

Content

Owner can modify the value of isKilled, when isKilled is true, it will Stop distributing inflation.

- contracts/farming/LiquidityGaugeV4.sol#L617

```
    function setKilled(bool _isKilled) external onlyOwner {
        isKilled = _isKilled;
    }
```

The toggleApproveMint function does not use events to record the historical information of user authorization, which is not conducive to user review.

- contracts/farming/Minter.sol#92

```
    function toggleApproveMint(address _mintingUser) external {
        allowedToMintFor[_mintingUser][msg.sender] = !allowedToMintFor[_mintingUser]
[msg.sender];
    }
```

The event does not record the change of the old and new values.

- contracts/SimpleGovernance.sol#L29-L55

```
function changeGovernance(address newGovernance) external onlyGovernance {
    require(
        newGovernance != governance,
        "governance must be different from current one"
    );
    require(newGovernance != address(0), "governance cannot be empty");
    pendingGovernance = newGovernance;
}

function acceptGovernance() external {
    address _pendingGovernance = pendingGovernance;
    require(
        _pendingGovernance != address(0),
        "changeGovernance must be called first"
    );
    require(
        _msgSender() == _pendingGovernance,
        "only pendingGovernance can accept this role"
    );
    pendingGovernance = address(0);
    governance = _msgSender();
    emit SetGovernance(_msgSender());
}
```

Solution

It is recommended to add events for recording, which is convenient for community users to review.

Status

Fixed

[N3] [Suggestion] Gas optimization

Category: Gas Optimization Audit

Content

The mintFor function uses the method of if to judge. If the condition is not met, the transaction will still be linked to the chain and consume gas. If require is used, the remaining gas will be returned.

- contracts/farming/**Minter**.sol#P62

```
function mintFor(address _gaugeAddr, address _for) external nonReentrant() {
    if(allowedToMintFor[msg.sender][_for]){









        _mintFor(_gaugeAddr, _for);
    }
}
```

Solution

It is recommended to use require instead of if for judgment.

Status

Fixed

[N4] [Low] Business logic is not clear

Category: Design Logic Audit

Content

The constructor function passed in the decimals_ parameter, but it was not used.

- contracts/guarded/LPTokenGuarded.sol#P24

```
constructor(
    string memory name_,
    string memory symbol_,
    uint8 decimals_
) public ERC20(name_, symbol_) {
    // _setupDecimals(decimals_);
    swap = ISwapGuarded(_msgSender());
}
```

Solution

Need to communicate with developers about the business logic here.

Status

Fixed

[N5] [Suggestion] Business logic error

Category: Design Logic Audit

Content

`blockhash(block.number)` is the hash of the current block that cannot be obtained.

- contracts/helper/Multicall2.sol#P64

```
    function blockAndAggregate(Call[] memory calls)
        public
        returns (
            uint256 blockNumber,
            bytes32 blockHash,
            Result[] memory returnData
        )
    {
        (blockNumber, blockHash, returnData) = tryBlockAndAggregate(
            true,
            calls
        );
    }
```

- contracts/helper/Multicall2.sol#L121

```
    function tryBlockAndAggregate(bool requireSuccess, Call[] memory calls)
        public
        returns (
            uint256 blockNumber,
            bytes32 blockHash,
            Result[] memory returnData
        )
    {
```

```
        blockNumber = block.number;
        blockHash = blockhash(block.number);
        returnData = tryAggregate(requireSuccess, calls);
    }
```

Solution

Need to communicate with developers about the business logic here.

Status

Ignored; The project team response: Multicall2 and Multicall will not be used in the actual business of the project.

[N6] [High] mint is not authenticated

Category: Authority Control Vulnerability

Content

Anyone can call mint function to mint coins.

- contracts/helper/GenericERC20.sol#D28-D17

```
    function mint(address recipient, uint256 amount) external {
        require(amount != 0, "amount == 0");
        _mint(recipient, amount);
    }
```

Solution

It is recommended to add permission checking for mint function.

Status

Fixed; The project team response: GenericBEP20 is a test

contract.

[N7] [Suggestion] Preemptive initialization issues Category:

Others

Content

When the function is initialized, the authentication is not performed, and there is an issue that the wrong parameters are input by the preemptive initialization.

- contracts/farming/GaugeController.sol#P37

```
    function __GaugeController_init(address _token, address _veToken)
        external
        initializer
        returns (bool)
    {
        require(_token != address(0), "Minter: !token");
        require(_veToken != address(0), "Minter: !veToken");

        __OwnerPausable_init();

        token = _token;
        votingEscrow = _veToken;
        timeTotal = block.timestamp / WEEK * WEEK;
    }
```

- contracts/farming/LiquidityGaugeV4.sol#T57

```
function __LiquidityGauge_init(address _lpToken, address _minter, address
_cada, address _veToken, address _gaugeCtrl)
        external
        initializer
    {

        require(_lpToken != address(0), "LiquidityGauge: !lp");
        require(_minter != address(0), "LiquidityGauge: !minter");
       require(_cada != address(0), "LiquidityGauge: !srs");
        require(_veToken != address(0), "LiquidityGauge: !ve");
        require(_gaugeCtrl != address(0), "LiquidityGauge: !gaugeCtrl");

        lpToken = _lpToken;

        string memory symbol = IERC20Extended(_lpToken).symbol();
        __ERC20_init(string(abi.encodePacked("Cada Finance ", symbol, "
Gauge Deposit")), string(abi.encodePacked(symbol, "-gauge")));
```

```
        MINTER = _minter;
        cada = _cada;
        VOTING_ESCROW = _veToken;
        GAUGE_CONTROLLER = _gaugeCtrl;

        periodTimestamp[0] = block.timestamp;
       inflationRate = ICADA(cada).rate();
        futureEpochTime =
       ICADA(cada).futureEpochTimeWrite();
    }
```

- contracts/farming/Minter.sol#L22

```
function __Minter_init(address _token, address _controller)
        external
        initializer
        returns (bool)
    {
        require(_token != address(0), "Minter: !token");
        require(_controller != address(0), "Minter: !controller");

        __OwnerPausable_init();

        token = _token;
        controller = _controller;
        return true;
    }
```

Solution

It is recommended to use the contract to deploy and call the initialization function, or call the initialization function immediately after deployment.

Status

Confirmed; The project team response: The initialization function will be called immediately after the contract is deployed.

[N8] [Suggestion] Lack of functions to withdraw excess token assets

Category: Others

Content

There are token assets in the contract for issuing rewards, but there is no function to reserve extra tokens, so the remaining tokens cannot be taken out.

- contracts/RetroactiveVesting.sol#L116

```
function claimReward(address account) external {
        if (account == address(0)) {
            account = msg.sender;
        }
        require(vestings[account].isVerified, "must verify first");
        _claimReward(account);
    }

    function _claimReward(address account) internal {
        VestingData storage vesting = vestings[account];
        uint256 released = vesting.released;
        uint256 amount = _vestedAmount(
            vesting.totalAmount,
            released,
            startTimestamp,
            DURATION
        );
        uint256 newReleased = amount + released;
        require(
            newReleased < type(uint120).max,
            "newReleased is too big to be cast uint120"
        );
        vesting.released = uint120(newReleased);
        token.safeTransfer(account, amount);

        emit Claimed(account, amount);
    }
```

Solution

It is recommended to add a function for extra redundant token withdrawal.

Status

Confirmed

[N9] [Suggestion] allowance can't reset

Category: Others

Content

No reset allowance function in SwapMigrator contract, If the allowance is used up, the contract cannot be used.

- contracts/SwapMigrator.sol#L45

```
    constructor(MigrationData memory usdData_, address owner_) public {
        // Approve old USD LP Token to be used by the old USD pool
        usdData_.oldPoolLPTokenAddress.approve(
            usdData_.oldPoolAddress,
            MAX_UINT256
        );

        // Approve USD tokens to be used by the new USD pool
        for (uint256 i = 0; i < usdData_.underlyingTokens.length; i++) {
            usdData_.underlyingTokens[i].safeApprove(
                usdData_.newPoolAddress,
                MAX_UINT256
            );
        }

        // Set storage variables
        usdPoolMigrationData = usdData_;
        owner = owner_;
    }
```

Solution

It is recommended to add the function of resetting the allowance.

Status

Confirmed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X76205240002	SlowMist Security Team	2022.09.28 - 2022.10.12	Passed

Summary conclusion: The SlowMist security team uses a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 high risk, 1 low risk, 7 suggestion vulnerabilities. And 3 suggestion vulnerabilities were confirmed; 1 suggestion was ignored; All other findings were fixed. The code was not deployed to the mainnet.

6 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist