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 2023.03.16, the SlowMist security team received the team's security audit application for PancakeSwap v3 Phase2, 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.
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
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

Serial Number	Audit Class	Audit Subclass
		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

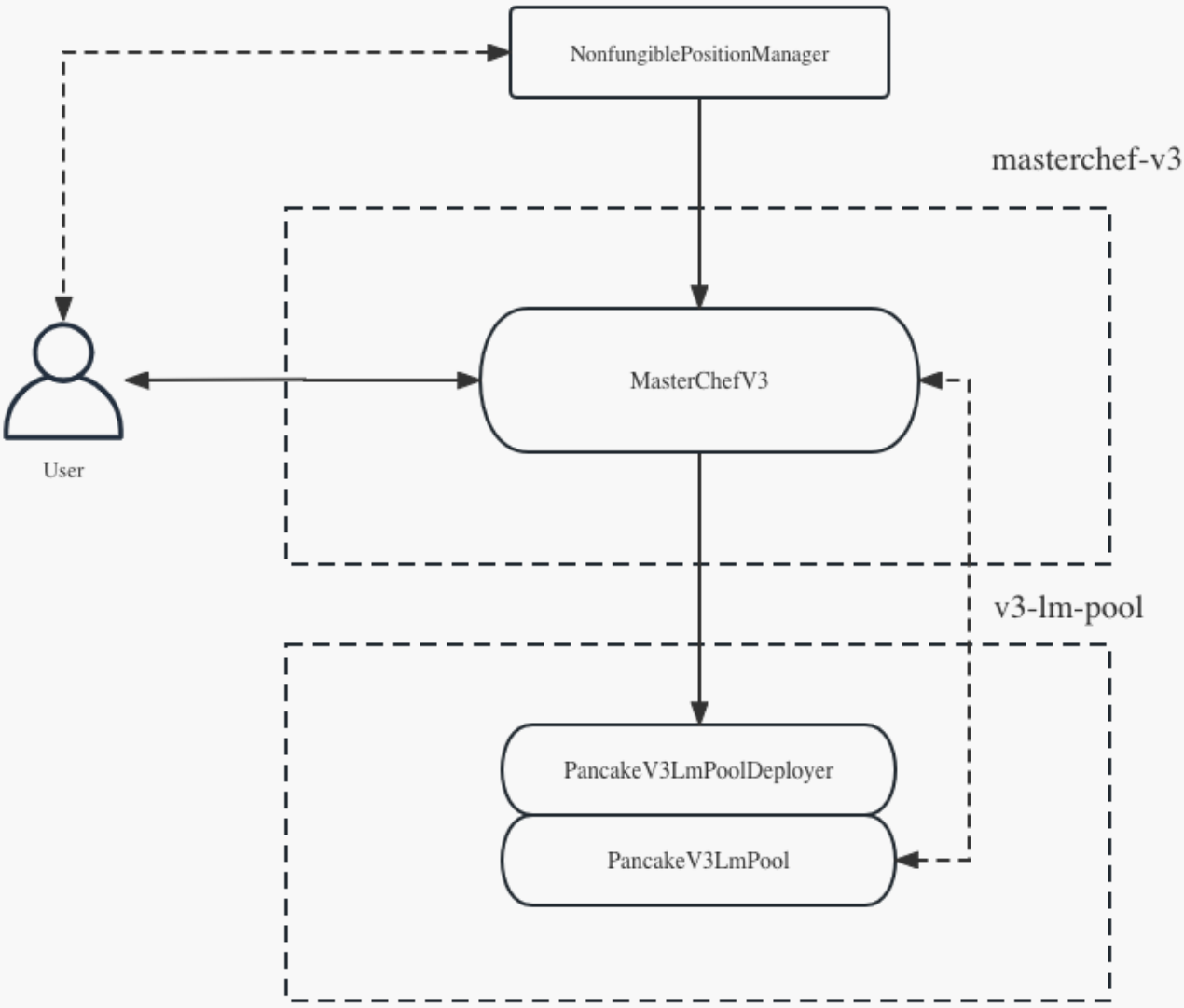
3.1 Project Introduction

PancakeSwap V3 Phase2 audit section is divided into two parts: MasterChefV3, and V3LMPool.

In the MasterChefV3 part, users can deposit their farm pool LP NFT in the pool for staking and get the CAKE tokens as rewards.

The MasterChefV3 contract will deploy the LmPool the calculate the reward. And the users can increase or decrease the liquidity and collect the fee through MasterChefV3. If the user wants to quit the staking. He can withdraw all his liquidity and tokens first, and then burn his LP NFT. The owner role of the MasterChefV3

contract can set emergency, the receiver contract, the LMPoolDeployer contract, set operator address, add a new pool, set period duration, update farm boost contract address, update the given pool's CAKE allocation point, and update the cake reward for the liquidity mining pool.



3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Risk of excessive authority	Authority Control Vulnerability Audit	Medium	Acknowledged
N2	LP token locking issue	Design Logic Audit	Low	Acknowledged

NO	Title	Category	Level	Status
N3	Cast truncation issue	Arithmetic Accuracy Deviation Vulnerability	Suggestion	Acknowledged

4 Code Overview

4.1 Contracts Description

Codebase:

Audit Version

https://github.com/pancakeswap/pancake-v3

commit: 4378b6c90ec9ddd07ba6a94d127badca734e2d83

- /masterchef-v3
- /v3-lm-pool

Fixed Version

https://github.com/pancakeswap/pancake-v3

commit: a85cf83d3e7f7e9346e53f11614b770cec586cc1

- /masterchef-v3
- /v3-lm-pool

The main network address of the contract is as follows:

The code was not 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:

Multicall			
Function Name	Visibility	Mutability	Modifiers
multicall	Public	Payable	-

Enumberable			
Function Name	Visibility	Mutability	Modifiers
tokenOfOwnerByIndex	Public	-	-
balanceOf	Public	-	-
addToken	Internal	Can Modify State	-
removeToken	Internal	Can Modify State	-
_addTokenToOwnerEnumeration	Private	Can Modify State	-
_removeTokenFromOwnerEnumeration	Private	Can Modify State	-

MasterChefV3			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getLatestPeriodInfoByPid	Public	-	-
getLatestPeriodInfo	Public	-	-
pendingCake	External	-	-
setEmergency	External	Can Modify State	onlyOwner
setReceiver	External	Can Modify State	onlyOwner
setLMPoolDeployer	External	Can Modify State	onlyOwner
add	External	Can Modify State	onlyOwner
set	External	Can Modify State	onlyOwner onlyValidPid
onERC721Received	External	Can Modify State	-

MasterChefV3			
harvest	External	Can Modify State	nonReentrant
harvestOperation	Internal	Can Modify State	-
withdraw	External	Can Modify State	nonReentrant
updateLiquidity	External	Can Modify State	nonReentrant
updateBoostMultiplier	External	Can Modify State	onlyBoostContract
updateLiquidityOperation	Internal	Can Modify State	-
increaseLiquidity	External	Can Modify State	nonReentrant
decreaseLiquidity	External	Can Modify State	nonReentrant
collect	External	Can Modify State	nonReentrant
burn	External	Can Modify State	nonReentrant
upkeep	External	Can Modify State	onlyReceiver
massUpdatePools	Internal	Can Modify State	-
updatePools	External	Can Modify State	onlyOwnerOrOperator
setOperator	External	Can Modify State	onlyOwner
setPeriodDuration	External	Can Modify State	onlyOwner
updateFarmBoostContract	External	Can Modify State	onlyOwner
_safeTransfer	Internal	Can Modify State	-

PancakeV3LmPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
accumulateReward	External	Can Modify State	onlyPoolOrMasterChef
crossLmTick	External	Can Modify State	onlyPool

PancakeV3LmPool			
updatePosition	External	Can Modify State	onlyMasterChef
getRewardGrowthInside	External	-	-

PancakeV3LmPoolDeployer			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
deploy	External	Can Modify State	onlyMasterChef

MasterChefV3Receiver			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
depositForMasterChefV2Pool	External	Can Modify State	onlyOwner
harvestFromV2	Internal	Can Modify State	-
upkeep	External	Can Modify State	onlyOwnerOrOperator
setOperator	External	Can Modify State	onlyOwner
withdraw	External	Can Modify State	onlyOwner

4.3 Vulnerability Summary

[N1] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

1.The owner role can change the external part contract `FARM_BOOSTER` through the updateFarmBoostContract function and the external part contract can affect the `boostMultiplier`.

Code location:

masterchef-v3/contracts/MasterChefV3.sol#L646-650

```
    function updateFarmBoostContract(address _newFarmBoostContract) external
onlyOwner {
        // farm booster can be zero address when need to remove farm booster
        FARM_BOOSTER = IFarmBooster(_newFarmBoostContract);
        emit UpdateFarmBoostContract(_newFarmBoostContract);
    }
```

2.The owner role can add a pool arbitrarily through the add function, and there is a risk that the Owner can add apool to mine by itself to obtain rewards. When calling the add function to add a pool, the `lastRewardTimestamp` and `totalAllocPoint` will be updated, and related information about the pool will be stored.

Code location:

masterchef-v3/contracts/MasterChefV3.soll#L254-284

```
    function add(uint256 _allocPoint, IPancakeV3Pool _v3Pool, bool _withUpdate)
external onlyOwner {
        if (_withUpdate) massUpdatePools();

        ILMPool lmPool = LMPoolDeployer.deploy(_v3Pool);

        totalAllocPoint += _allocPoint;
        address token0 = _v3Pool.token0();
        address token1 = _v3Pool.token1();
        uint24 fee = _v3Pool.fee();

        …

        v3PoolPid[token0][token1][fee] = poolLength;
        v3PoolAddressPid[address(_v3Pool)] = poolLength;
        emit AddPool(poolLength, _allocPoint, _v3Pool, lmPool);
    }
```

3.The owner role can update the distribution weight of the pool through the set function, and the updated weight will affect the user's mining reward. The owner calls to add and in the set function, when the value of the `_withUpdate` parameter is passed as true, the mining pool will be updated. At this time, the set function will also update the distribution weight of the mining pool. After the update, the subsequent mining will be

calculated according to the new weight. The modification of the reward distribution rate will affect the rewards of users for mining.

Code location:

masterchef-v3/contracts/MasterChefV3.soll#290-302

```
    function set(uint256 _pid, uint256 _allocPoint, bool _withUpdate) external
onlyOwner onlyValidPid(_pid) {
        uint32 currentTime = uint32(block.timestamp);
        PoolInfo storage pool = poolInfo[_pid];
        ILMPool LMPool = ILMPool(pool.v3Pool.lmPool());
        if (address(LMPool) != address(0)) {
            LMPool.accumulateReward(currentTime);
        }

        if (_withUpdate) massUpdatePools();
        totalAllocPoint = totalAllocPoint - pool.allocPoint + _allocPoint;
        pool.allocPoint = _allocPoint;
        emit SetPool(_pid, _allocPoint);
    }
```

Solution

In the short term, transferring owner ownership to multisig contracts is an effective solution to avoid single-point risk. But in the long run, it is a more reasonable solution to implement a privilege separation strategy and set up multiple privileged roles to manage each privileged function separately. And the authority involving user funds should be managed by the community, and the authority involving emergency contract suspension can be managed by the EOA address. This ensures both a quick response to threats and the safety of user funds.

Status

Acknowledged; After communication with the project team, they express that they will use multisig wallet to control the owner.

[N2] [Low] LP token locking issue

Category: Design Logic Audit

Content

In the MasterChefV3 contract, users will transfer their ERC721 LP tokens for staking to get the CAKE as reward. Users can only call the safeTransferFrom function to transfer their ERC721 LP token in the MasterChefV3

contract to trigger the _checkOnERC721Received hook to let the NonfungiblePositionManager contract call back the onERC721Received function. After this, the positionInfo can be recorded and make the staking effective. If users miss transferring the LP token by using the transferFrom function, the LP tokens will be locked in this contract.

Code location:

masterchef-v3/contracts/MasterChefV3.sol#314

```
    function onERC721Received(address, address _from, uint256 _tokenId, bytes
calldata) external returns (bytes4) {
        if (msg.sender != address(nonfungiblePositionManager)) revert
NotPancakeNFT();
        DepositCache memory cache;
        (
            ,
            ,
            cache.token0,
            cache.token1,
            cache.fee,
            cache.tickLower,
            cache.tickUpper,
            cache.liquidity,
            ,
            ,
            ,

        ) = nonfungiblePositionManager.positions(_tokenId);
        if (cache.liquidity == 0) revert NoLiquidity();
        uint256 pid = v3PoolPid[cache.token0][cache.token1][cache.fee];
        if (pid == 0) revert InvalidNFT();
        PoolInfo memory pool = poolInfo[pid];
        ILMPool LMPool = ILMPool(pool.v3Pool.lmPool());
        if (address(LMPool) == address(0)) revert NoLMPool();

        UserPositionInfo storage positionInfo = userPositionInfos[_tokenId];

        positionInfo.tickLower = cache.tickLower;
        positionInfo.tickUpper = cache.tickUpper;
        positionInfo.user = _from;
        positionInfo.pid = pid;
        updateLiquidityOperation(positionInfo, _tokenId, 0);

        positionInfo.rewardGrowthInside =
LMPool.getRewardGrowthInside(cache.tickLower, cache.tickUpper);
```

```
        // Update Enumberable
        addToken(_from, _tokenId);
        emit Deposit(_from, pid, _tokenId, cache.liquidity, cache.tickLower,
cache.tickUpper);

        return this.onERC721Received.selector;
    }
```

Solution

It's recommended to add a function to transfer the miss transferred LP token which doesn't record the positionInfo, and strict restrictions on the front end. Or use deposit or approve to let the user transfer their LP token

Status

Acknowledged

[N3] [Suggestion] Cast truncation issue

Category: Arithmetic Accuracy Deviation Vulnerability

Content

In the PancakeV3LmPool contract, the Pool or MasterChef will calculate the reward through the accumulateReward function. The uint256 `endTime` is assigned by getLatestPeriodInfo in the MasterChef contract and the endTime is assigned by an uint256 value `latestPeriodEndTime`, then the `endTime` will cast to an uint32 to `endTimestamp`. If the `latestPeriodEndTime` is larger than `type(uin32).max`, there will be a cast truncation issue. And PancakeV3LmPool contract imports the SafeCast contract but doesn't use it for the uin32 cast.

Code location:

v3-lm-pool/contracts/PancakeV3LmPool.sol#64-67

```
    if (lmLiquidity != 0) {
      (uint256 rewardPerSecond, uint256 endTime) =
masterChef.getLatestPeriodInfo(address(pool));

      uint32 endTimestamp = uint32(endTime);
```

Solution

It's recommended to use the SafeCast to prevent the cast truncation issue.

Status

Acknowledged; After communication with the project team, they expressed that the `type(uin32).max` will be Sunday, February 7, 2106 6:28:15 AM

and it is safe.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002303220001	SlowMist Security Team	2023.03.16 - 2023.03.22	Medium Risk

Summary conclusion: The SlowMist security team use a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 medium risk, 1 low risk, 1 suggestion vulnerabilities. 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