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 2022.03.28, the SlowMist security team received the team's security audit application for Pancakeswap-CakePool, 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:

- Reentrancy Vulnerability
- Replay Vulnerability
- Reordering Vulnerability
- Short Address Vulnerability
- Denial of Service Vulnerability
- Transaction Ordering Dependence Vulnerability
- Race Conditions Vulnerability
- Authority Control Vulnerability
- Integer Overflow and Underflow Vulnerability
- TimeStamp Dependence Vulnerability
- Uninitialized Storage Pointers Vulnerability
- Arithmetic Accuracy Deviation Vulnerability
- tx.origin Authentication Vulnerability

- "False top-up" Vulnerability
- Variable Coverage Vulnerability
- Gas Optimization Audit
- Malicious Event Log Audit
- Redundant Fallback Function Audit
- Unsafe External Call Audit
- Explicit Visibility of Functions State Variables Audit
- Design Logic Audit
- Scoping and Declarations Audit

3 Project Overview

3.1 Project Introduction

Project address:

https://github.com/ChefSnoopy/pancake-contracts/blob/master/projects/cake-pool/contracts/CakePool.sol

Audit Version:

052a0afe87bf8144345d6c5962da3a79be03603c

Fixed Version:

22472d8365609dce3ec67056fb814fb23b0bd9c6

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	Medium	Fixed
N2	Missing event records	Others	Suggestion	Fixed

4 Code Overview

4.1 Contracts Description

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:

CakePool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
init	External	Can Modify State	-
updateBoostContractInfo	Internal	Can Modify State	-
updateUserShare	Internal	Can Modify State	-
unlock	External	Can Modify State	onlyOperatorOrCakeOwner whenNotPaused

CakePool			
deposit	External	Can Modify State	whenNotPaused
depositOperation	Internal	Can Modify State	-
withdraw	Public	Can Modify State	-
withdrawAll	External	Can Modify State	-
harvest	Internal	Can Modify State	-
setAdmin	External	Can Modify State	onlyOwner
setTreasury	External	Can Modify State	onlyOwner
setOperator	External	Can Modify State	onlyOwner
setBoostContract	External	Can Modify State	onlyAdmin
setFreeFeeUser	External	Can Modify State	onlyAdmin
setPerformanceFee	External	Can Modify State	onlyAdmin
setPerformanceFeeContract	External	Can Modify State	onlyAdmin
setWithdrawFee	External	Can Modify State	onlyAdmin
setWithdrawFeeContract	External	Can Modify State	onlyAdmin
setWithdrawFeePeriod	External	Can Modify State	onlyAdmin
setMaxLockDuration	External	Can Modify State	onlyAdmin

CakePool			
setDurationFactor	External	Can Modify State	onlyAdmin
setDurationFactorOverdue	External	Can Modify State	onlyAdmin
setUnlockFreeDuration	External	Can Modify State	onlyAdmin
setBoostWeight	External	Can Modify State	onlyAdmin
inCaseTokensGetStuck	External	Can Modify State	onlyAdmin
pause	External	Can Modify State	onlyAdmin whenNotPaused
unpause	External	Can Modify State	onlyAdmin whenPaused
calculateTotalPendingCakeRewards	External	-	-
getPricePerFullShare	External	-	-
available	Public	-	-
balanceOf	Public	-	-
_isContract	Internal	-	-

4.3 Vulnerability Summary

[N1] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability

Content

In the cakePool contract, the owner role can set the DURATION_FACTOR and DURATION_FACTOR_OVERDUE by calling the setDurationFactor and setDurationFactorOverdue. If these values are set too large or too small, this may affect the calculation of user.shares when calling the deposit and withdraw function.

Code location:

cake-pool/contracts/CakePool#552-561

```
    function setDurationFactor(uint256 _durationFactor) external onlyAdmin {
        require(_durationFactor > 0, "DURATION_FACTOR cannot be zero");
        DURATION_FACTOR = _durationFactor;
    }

    /**
     * @notice Set DURATION_FACTOR_OVERDUE
     * @dev Only callable by the contract admin.
     */
    function setDurationFactorOverdue(uint256 _durationFactorOverdue) external 
onlyAdmin {
        require(_durationFactorOverdue > 0, "DURATION_FACTOR_OVERDUE cannot be 
zero");
        DURATION_FACTOR_OVERDUE = _durationFactorOverdue;
    }
```

Solution

It is recommended to use a time lock mechanism or community governance to restrict.

Status

Fixed

[N2] [Suggestion] Missing event records

Category: Others

Content

In the cakePool contract, the owner role can set admin, treasury, operator, boostContract, freeFeeUsers, performanceFee, performanceFeeContract, withdrawFee, withdrawFeeContract, withdrawFeePeriod, MAX_LOCK_DURATION, DURATION_FACTOR, DURATION_FACTOR_OVERDUE, UNLOCK_FREE_DURATION and

BOOST_WEIGHT by calling the setAdmin, setTreasury, setOperator, setBoostContract, setFreeFeeUser, setPerformanceFee, setPerformanceFeeContract, setWithdrawFee, setWithdrawFeeContract, setWithdrawFeePeriod, setMaxLockDuration, setDurationFactor, setDurationFactorOverdue, setUnlockFreeDuration and setBoostWeight. but there no event logging is preformed.

Code location:

cake-pool/contracts/CakePool#443-586

```
    function setAdmin(address _admin) external onlyOwner {
        require(_admin != address(0), "Cannot be zero address");
        admin = _admin;
    }

    /**
     * @notice Set treasury address
     * @dev Only callable by the contract owner.
     */
    function setTreasury(address _treasury) external onlyOwner {
        require(_treasury != address(0), "Cannot be zero address");
        treasury = _treasury;
    }

    /**
     * @notice Set operator address
     * @dev Callable by the contract owner.
     */
    function setOperator(address _operator) external onlyOwner {
        require(_operator != address(0), "Cannot be zero address");
        operator = _operator;
    }

    /**
     * @notice Set Boost Contract address
     * @dev Callable by the contract admin.
     */
    function setBoostContract(address _boostContract) external onlyAdmin {
        require(_boostContract != address(0), "Cannot be zero address");
        boostContract = _boostContract;
    }
    /**
     * @notice Set free fee address
     * @dev Only callable by the contract admin.
     * @param _user: User address
     * @param _free: true:free false:not free
     */
    function setFreeFeeUser(address _user, bool _free) external onlyAdmin {
        require(_user != address(0), "Cannot be zero address");
        freeFeeUsers[_user] = _free;
    }

    /**
     * @notice Set performance fee
     * @dev Only callable by the contract admin.
     */
    function setPerformanceFee(uint256 _performanceFee) external onlyAdmin {
        require(_performanceFee <= MAX_PERFORMANCE_FEE, "performanceFee cannot be
more than MAX_PERFORMANCE_FEE");
        performanceFee = _performanceFee;
    }

    /**
     * @notice Set performance fee for contract
     * @dev Only callable by the contract admin.
     */
    function setPerformanceFeeContract(uint256 _performanceFeeContract) external
onlyAdmin {
        require(
            _performanceFeeContract <= MAX_PERFORMANCE_FEE,
            "performanceFee cannot be more than MAX_PERFORMANCE_FEE"
        );
        performanceFeeContract = _performanceFeeContract;
    }

    /**
     * @notice Set withdraw fee
     * @dev Only callable by the contract admin.
     */
    function setWithdrawFee(uint256 _withdrawFee) external onlyAdmin {
        require(_withdrawFee <= MAX_WITHDRAW_FEE, "withdrawFee cannot be more than
MAX_WITHDRAW_FEE");
        withdrawFee = _withdrawFee;
    }
    /**
     * @notice Set withdraw fee for contract
     * @dev Only callable by the contract admin.
     */
    function setWithdrawFeeContract(uint256 _withdrawFeeContract) external onlyAdmin
{
        require(_withdrawFeeContract <= MAX_WITHDRAW_FEE, "withdrawFee cannot be more
than MAX_WITHDRAW_FEE");
        withdrawFeeContract = _withdrawFeeContract;
    }

    /**
     * @notice Set withdraw fee period
     * @dev Only callable by the contract admin.
     */
    function setWithdrawFeePeriod(uint256 _withdrawFeePeriod) external onlyAdmin {
        require(
            _withdrawFeePeriod <= MAX_WITHDRAW_FEE_PERIOD,
            "withdrawFeePeriod cannot be more than MAX_WITHDRAW_FEE_PERIOD"
        );
        withdrawFeePeriod = _withdrawFeePeriod;
    }

    /**
     * @notice Set MAX_LOCK_DURATION
     * @dev Only callable by the contract admin.
     */
    function setMaxLockDuration(uint256 _maxLockDuration) external onlyAdmin {
        require(
            _maxLockDuration <= MAX_LOCK_DURATION_LIMIT,
            "MAX_LOCK_DURATION cannot be more than MAX_LOCK_DURATION_LIMIT"
        );
        MAX_LOCK_DURATION = _maxLockDuration;
    }

    /**
     * @notice Set DURATION_FACTOR
     * @dev Only callable by the contract admin.
     */
    function setDurationFactor(uint256 _durationFactor) external onlyAdmin {
        require(_durationFactor > 0, "DURATION_FACTOR cannot be zero");
        DURATION_FACTOR = _durationFactor;
    }
    /**
     * @notice Set DURATION_FACTOR_OVERDUE
     * @dev Only callable by the contract admin.
     */
    function setDurationFactorOverdue(uint256 _durationFactorOverdue) external
onlyAdmin {
        require(_durationFactorOverdue > 0, "DURATION_FACTOR_OVERDUE cannot be
zero");
        DURATION_FACTOR_OVERDUE = _durationFactorOverdue;
    }

    /**
     * @notice Set UNLOCK_FREE_DURATION
     * @dev Only callable by the contract admin.
     */
    function setUnlockFreeDuration(uint256 _unlockFreeDuration) external onlyAdmin {
        require(_unlockFreeDuration > 0, "UNLOCK_FREE_DURATION cannot be zero");
        UNLOCK_FREE_DURATION = _unlockFreeDuration;
    }

    /**
     * @notice Set BOOST_WEIGHT
     * @dev Only callable by the contract admin.
     */
    function setBoostWeight(uint256 _boostWeight) external onlyAdmin {
        require(_boostWeight <= BOOST_WEIGHT_LIMIT, "BOOST_WEIGHT cannot be more than
BOOST_WEIGHT_LIMIT");
        BOOST_WEIGHT = _boostWeight;
    }
```

Solution

It is recommended to record events when modifying sensitive parameters.

Status

Fixed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002204010002	SlowMist Security Team	2022.03.28 - 2022.04.01	Passed

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 suggestion vulnerability. All 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.