SLOWMIST

Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.03.11, the SlowMist security team received the PancakeSwap team's security audit application for MasterChef v2, 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

Audit Version:

https://github.com/chefcooper/pancake-contracts/blob/dev/MasterChefV2/projects/masterchef/v2/contracts/MasterChefV2.sol

commit: 677b150db5a5f2eccd9bd425af526663f581bc22

Fixed Version:

https://github.com/chefcooper/pancake-contracts/blob/dev/MasterChefV2/projects/masterchef/v2/contracts/MasterChefV2.sol

commit: 58102f2adf4ffa2da77c4301d80af315a3f653e6

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Redundant logic issue	Others	Suggestion	Ignored
N2	Permission control issue	Authority Control Vulnerability	Medium	Fixed
N3	The number of pendingCakeToBurn is not checked	Design Logic Audit	Low	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:

MasterChefV2			
Function Name	Visibility	Mutability	Modifiers

MasterChefV2			
<Constructor>	Public	Can Modify State	-
init	External	Can Modify State	-
poolLength	Public	-	-
add	External	Can Modify State	onlyOwner
set	External	Can Modify State	onlyOwner
pendingCake	External	-	-
massUpdatePools	Public	Can Modify State	-
cakePerBlock	Public	-	-
cakePerBlockToBurn	Public	-	-
updatePool	Public	Can Modify State	-
deposit	External	Can Modify State	nonReentrant
withdraw	External	Can Modify State	nonReentrant
settlePendingCake	Internal	Can Modify State	-
harvestFromMasterChef	Public	Can Modify State	-
emergencyWithdraw	External	Can Modify State	nonReentrant
burnCake	Public	Can Modify State	onlyOwner
updateCakeRate	External	Can Modify State	onlyOwner
updateBurnAdmin	External	Can Modify State	onlyOwner
updateWhiteList	External	Can Modify State	onlyOwner
updateBoostContract	External	Can Modify State	onlyOwner

MasterChefV2			
updateBoostMultiplier	External	Can Modify State	onlyBoostContract nonReentrant
getBoostMultiplier	Public	-	-

4.3 Vulnerability Summary

[N1] [Suggestion] Redundant logic issue

Category: Others

Content

In the add function, the owner can add new pools. It will first check whether the number of newly added lpTokens in the contract is greater than or equal to 0. But in fact, the number of lpTokens in the contract will be greater than or equal to 0 in any case, so this check is redundant.

Code location:

```
function add(
        uint256 _allocPoint,
        IBEP20 _lpToken,
        bool _isRegular,
        bool _withUpdate
    ) external onlyOwner {
        require(_lpToken.balanceOf(address(this)) >= 0, "None BEP20 tokens");

        ...

        emit AddPool(lpToken.length.sub(1), _allocPoint, _lpToken, _isRegular);
    }
```

Solution

It is recommended to remove redundant logic.

Status

Ignored; After communicating with the project team, the project team stated that this is an expected design to avoid non-token lpTokens from being added to the pool.

[N2] [Medium] Permission control issue

Category: Authority Control Vulnerability

Content

If any user holds the dummy token, the user can stake the dummy token to the MasterChef v1 contract through the init function, which will cause the lastBurnedBlock parameter to be updated unexpectedly, and finally lead to an error in the calculation of the number of CAKE tokens waiting to be burned.

Code location:

```
    function init(IBEP20 dummyToken) external {
        uint256 balance = dummyToken.balanceOf(msg.sender);
        require(balance != 0, "MasterChefV2: Balance must exceed 0");
        dummyToken.safeTransferFrom(msg.sender, address(this), balance);
        dummyToken.approve(address(MASTER_CHEF), balance);
        MASTER_CHEF.deposit(MASTER_PID, balance);
        // MCV2 start to earn CAKE reward from current block in MCV1 pool
        lastBurnedBlock = block.number;
        emit Init();
    }
```

Solution

It is recommended to perform permission control on the init function.

Status

Fixed; Fixed in commit 58102f2adf4ffa2da77c4301d80af315a3f653e6

[N3] [Low] The number of pendingCakeToBurn is not checked

Category: Design Logic Audit

Content

In the burnCake function, if the number of CAKE tokens in the contract is less than pendingCakeToBurn, it will harvest CAKE tokens from MasterChef v1 via the harvestFromMasterChef function. But it does not check if the balance of CAKE tokens in the contract after harvesting is greater than or equal to pendingCakeToBurn.

Code location:

```
function burnCake(bool _withUpdate) public onlyOwner {
    if (_withUpdate) {
        massUpdatePools();
    }

    uint256 multiplier = block.number.sub(lastBurnedBlock);
    uint256 _pendingCakeToBurn = multiplier.mul(cakePerBlockToBurn());
    lastBurnedBlock = block.number;

    if (_pendingCakeToBurn > 0) {
        if (CAKE.balanceOf(address(this)) < _pendingCakeToBurn) {
            harvestFromMasterChef();
        }
        CAKE.safeTransfer(burnAdmin, _pendingCakeToBurn);
    }
}
```

Solution

If it is not designed as expected, it is recommended to check the CAKE token balance in the contract again after the harvestFromMasterChef operation.

Status

Fixed; Fixed in commit 58102f2adf4ffa2da77c4301d80af315a3f653e6

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002203170002	SlowMist Security Team	2022.03.11 - 2022.03.17	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 low risk, 1 suggestion. And 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