Cakepie Rewarder

Security Audit Report

January 16, 2024

Contents

1 Introduction

1.1 About Cakepie Rewarder

Developed by `Magpie`, `Cakepie` is a DeFi platform developed atop `PancakeSwap`. The audited `Cakepie Rewarder` introduces a comprehensive set of reward mechanisms, empowering users to earn various reward tokens. These mechanisms are thoughtfully designed to enhance user engagement and incentivize participation within the system.

1.2 Source Code

The following source code was reviewed during the audit:

- https://github.com/magpiexyz/cakepie_contract/pull/37
- Commit ID: 81789fc

2 Overall Assessment

This report has been compiled to identify issues and vulnerabilities within the `Cakepie Rewarder` project. Throughout this audit, we identified a total of 3 issues spanning various severity levels. By employing auxiliary tool techniques to supplement our thorough manual code review, we have discovered the following findings.

Severity	Count	Acknowledged	Won't Do	Addressed
Critical	-	-	-	-
High	-	-	-	-
Medium	-	-	-	-
Low	2	1	-	1
Informational	1	-	-	1
Undetermined	-	-	-	-

3 Vulnerability Summary

3.1 Overview

Click on an issue to jump to it, or scroll down to see them all.

L-1 Potential Risks Associated with Centralization

~~L-2~~ Revisited Logic of MasterCakepie::pendingTokens()/allPendingTokens()

~~I-1~~ Meaningful Events for Key Operations

3.2 Security Level Reference

In web3 smart contract audits, vulnerabilities are typically classified into different severity levels based on the potential impact they can have on the security and functionality of the contract. Here are the definitions for critical-severity, high-severity, medium-severity, and low-severity vulnerabilities:

Severity	Description
C-X (Critical)	A severe security flaw with immediate and significant negative consequences. It poses high risks, such as unauthorized access, financial losses, or complete disruption of functionality. Requires immediate attention and remediation.
H-X (High)	Significant security issues that can lead to substantial risks. Although not as severe as critical vulnerabilities, they can still result in unauthorized access, manipulation of contract state, or financial losses. Prompt remediation is necessary.
M-X (Medium)	Moderately impactful security weaknesses that require attention and remediation. They may lead to limited unauthorized access, minor financial losses, or potential disruptions to functionality.
L-X (Low)	Minor security issues with limited impact. While they may not pose significant risks, it is still recommended to address them to maintain a robust and secure smart contract.
I-X (Informational)	Warnings and things to keep in mind when operating the protocol. No immediate action required.
U-X (Undetermined)	Identified security flaw requiring further investigation. Severity and impact need to be determined. Additional assessment and analysis are necessary.

3.3 Vulnerability Details

[L-1] Potential Risks Associated with Centralization

Target	Category	IMPACT	LIKELIHOOD	STATUS
Multiple Contracts	Security	Low	Low	Acknowledged

In the `Cakepie Rewarder` implementation, the existence of a privileged `owner` account introduces centralization risks, as it holds significant control and authority over critical operations governing the protocol. In the following, we show the representative function potentially affected by the privileges associated with the privileged account.

StreamRewarder

```
function setRewardQueuerStatus(address _rewardQueuer, bool status) external
    onlyOwner {
    isRewardQueuer[_rewardQueuer] = status;

    emit QueuerStatusUpdated(_rewardQueuer, status);
}

function setMasterCakepie(address _masterCakepie) external onlyOwner {
    address oldMasterCakepie = masterCakepie;
    masterCakepie = _masterCakepie;

    emit MasterCakepieUpdated(oldMasterCakepie, _masterCakepie);
}
```

Remediation To mitigate the identified issue, it is recommended to introduce multi-sig mechanism to undertake the role of the privileged account. Moreover, it is advisable to implement timelocks to govern all modifications to the privileged operations.

Response By Team This issue has been confirmed by the team and the multi-sig mechanism will be used to mitigate it.

[L-2] Revisited Logic of MasterCakepie::pendingTokens()/allPendingTokens()

Target	Category	IMPACT	LIKELIHOOD	STATUS
MasterCakepie.sol	Business Logic	Low	Low	Addressed

The `MasterCakepie` contract is designed to implement an incentive mechanism that encourages users to stake supported assets. As a result of this staking activity, participants receive rewards in the form of various tokens. Specifically, the `pendingTokens()` function is utilized by users to query the pending rewards associated with the specified `_stakingToken` and `_rewardToken`.

Upon scrutinizing its logic, we've identified that the current implementation neglects the potential existence of two distinct rewarders (i.e., the current rewarder and the legacy rewarder) in specific scenarios. In such instances, the accuracy of the returned pending rewards may be compromised.

MasterCakepie::pendingTokens()

```
function pendingTokens(
    address _stakingToken,
    address _user,
    address _rewardToken
)
    external
    view
    returns (
        uint256 pendingCakepie,
        address bonusTokenAddress,
        string memory bonusTokenSymbol,
        uint256 pendingBonusToken
    )
{
    PoolInfo storage pool = tokenToPoolInfo[_stakingToken];
    pendingCakepie = _calCakepieReward(_stakingToken, _user);

    // If it's a multiple reward farm, we return info about the specific bonus token
    if (address(pool.rewarder) != address(0) && _rewardToken != address(0)) {
        (bonusTokenAddress, bonusTokenSymbol) = (
            _rewardToken,
            IERC20Metadata(_rewardToken).symbol()
        );
        pendingBonusToken = IBaseRewardPool(pool.rewarder).earned(_user, _rewardToken);
    }
}
```

Remediation Consider the potential existence of two distinct rewarders during the pending rewards calculation in the `pendingTokens()/allPendingTokens()` functions.

[I-1] Meaningful Events for Key Operations

Target	Category	IMPACT	LIKELIHOOD	STATUS
MasterCakepie.sol	Coding Practices	N/A	N/A	Addressed

The `event` feature is vital for capturing runtime dynamics in a contract. Upon emission, `events` store transaction arguments in logs, supplying external analytics and reporting tools with crucial information. They play a pivotal role in scenarios like modifying system-wide parameters or handling token operations.

However, in our examination of protocol dynamics, we observed that certain privileged routines lack meaningful events to document their changes. We highlight the representative routines below.

MasterCakepie

```
function updateWhitelistedAllocManager(address _account, bool _allowed) external
    onlyOwner {
    AllocationManagers[_account] = _allowed;
}

function setPancakeV3Helper(address _pancakeV3Helper) external onlyOwner {
    pancakeV3Helper = IPancakeV3Helper(_pancakeV3Helper);
}

function setLegacyRewarder(address _stakingToken, address _legacyRewarder)
    external onlyOwner {
    legacyRewarders[_stakingToken] = _legacyRewarder;
}
```

Remediation Ensure the proper emission of meaningful events containing accurate information to promptly reflect state changes.

4 Appendix

4.1 About AstraSec

AstraSec is a blockchain security company that serves to provide high-quality auditing services for blockchain-based protocols. With a team of blockchain specialists, AstraSec maintains a strong commitment to excellence and client satisfaction. The audit team members have extensive audit experience for various famous DeFi projects. AstraSec's comprehensive approach and deep blockchain understanding make it a trusted partner for the clients.

4.2 Disclaimer

The information provided in this audit report is for reference only and does not constitute any legal, financial, or investment advice. Any views, suggestions, or conclusions in the audit report are based on the limited information and conditions obtained during the audit process and may be subject to unknown risks and uncertainties. While we make every effort to ensure the accuracy and completeness of the audit report, we are not responsible for any errors or omissions in the report.

We recommend users to carefully consider the information in the audit report based on their own independent judgment and professional advice before making any decisions. We are not responsible for the consequences of the use of the audit report, including but not limited to any losses or damages resulting from reliance on the audit report.

This audit report is for reference only and should not be considered a substitute for legal documents or contracts.

4.3 Contact

Name	AstraSec Team
Phone	+86 176 2267 4194
Email	contact@astrasec.ai
Twitter	https://twitter.com/AstraSecAI