Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.07.04, the SlowMist security team received the PancakeSwap team's security audit application for PancakeSwap Pottery, 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/chefrabbid/pancake-contracts/tree/feature/PAN-397-pottery/projects/pottery

commit: fd7e112fa8743d11ba135330f3024fd638d70977

Fix version

https://github.com/chefrabbid/pancake-contracts/tree/feature/PAN-397-pottery/projects/pottery

commit: 8a41752eee1e40b5e541e33e3c97639c810529f1

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Missing event record	Others	Suggestion	Fixed
N2	Business logic problem	Design Logic Audit	Suggestion	Fixed
N3	Risk of excessive authority	Authority Control Vulnerability	Low	Confirmed

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:

PancakeSwapPotteryDraw			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
init	External	Can Modify State	onlyOwner
generatePottery	Public	Can Modify State	onlyOwner
startDraw	External	Can Modify State	onlyKeeperOrOwner
forceRequestDraw	External	Can Modify State	onlyOwner
closeDraw	External	Can Modify State	onlyKeeperOrOwner
claimReward	External	Can Modify State	-
timeToDraw	Public	-	-
rngFulfillRandomWords	Public	-	-
getWinners	External	-	-
getDraw	External	-	-
getPot	External	-	-
getNumOfDraw	External	-	-
getNumOfWinner	External	-	-
getPotteryPeriod	External	-	-
getTreasury	External	-	-
setVaultFactory	Public	Can Modify State	onlyOwner
setKeeper	Public	Can Modify State	onlyOwner

PancakeSwapPotteryDraw			
setTreasury	Public	Can Modify State	onlyOwner
setClaimFee	Public	Can Modify State	onlyOwner

PancakeSwapPotteryVault			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Share
_beforeTokenTransfer	Internal	Can Modify State	-
deposit	External	Can Modify State	-
mint	External	Can Modify State	-
withdraw	External	Can Modify State	-
redeem	External	Can Modify State	-
asset	Public	-	-
totalAssets	Public	-	-
maxDeposit	External	-	-
maxMint	External	-	-
maxWithdraw	External	-	-
maxRedeem	External	-	-
convertToShares	Public	-	-
convertToAssets	Public	-	-
previewDeposit	Public	-	-

PancakeSwapPotteryVault			
previewMint	Public	-	-
previewWithdraw	Public	-	-
previewRedeem	Public	-	-
lockCake	External	Can Modify State	onlyKeeperOrOwner
unlockCake	External	Can Modify State	onlyKeeperOrOwner
draw	External	-	-
getNumberOfTickets	External	-	-
getLockTime	External	-	-
passLockTime	Public	-	-
getStatus	Public	-	-
generateUserId	Public	-	-
setKeeper	External	Can Modify State	onlyOwner

Share			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
name	Public	-	-
symbol	Public	-	-
decimals	Public	-	-
totalSupply	Public	-	-

Share			
balanceOf	Public	-	-
transfer	Public	Can Modify State	-
allowance	Public	-	-
approve	Public	Can Modify State	-
transferFrom	Public	Can Modify State	-
increaseAllowance	Public	Can Modify State	-
decreaseAllowance	Public	Can Modify State	-
_transfer	Internal	Can Modify State	-
_mint	Internal	Can Modify State	-
_burn	Internal	Can Modify State	-
_approve	Internal	Can Modify State	-
_beforeTokenTransfer	Internal	Can Modify State	-
_afterTokenTransfer	Internal	Can Modify State	-

PotteryVaultFactory			
Function Name	Visibility	Mutability	Modifiers
generateVault	External	Can Modify State	-

PotteryKeeper			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-

PotteryKeeper			
getActiveVaults	External	-	-
checkUpkeep	External	-	-
performUpkeep	External	Can Modify State	onlyKeeperRegistry
addActiveVault	External	Can Modify State	onlyPotteryDrawOrOwner
removeActiveVault	External	Can Modify State	onlyPotteryDrawOrOwner
popActiveVault	Internal	Can Modify State	-
setKeeperRegistry	Public	Can Modify State	onlyOwner
setPotteryDraw	Public	Can Modify State	onlyOwner

RandomNumberGenerator			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	VRFConsumerBaseV2
setKeyHash	External	Can Modify State	onlyOwner
setSubId	External	Can Modify State	onlyOwner
setPotteryDraw	External	Can Modify State	onlyOwner
setCallBackGasLimit	External	Can Modify State	onlyOwner
getRandomWords	External	-	-
getLatestRequestId	External	-	-
fulfillRequest	External	-	-
requestRandomWords	External	Can Modify State	onlyPotteryDraw

RandomNumberGenerator			
fulfillRandomWords	Internal	Can Modify State	-

Ownable			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
owner	Public	-	-
renounceOwnership	Public	Can Modify State	onlyOwner
transferOwnership	Public	Can Modify State	onlyOwner
_transferOwnership	Internal	Can Modify State	-

Context			
Function Name	Visibility	Mutability	Modifiers
_msgSender	Internal	-	-
_msgData	Internal	-	-

KeeperCompatible			
Function Name	Visibility	Mutability	Modifiers

KeeperBase			
Function Name	Visibility	Mutability	Modifiers
preventExecution	Internal	-	-

4.3 Vulnerability Summary

[N1] [Suggestion] Missing event record

Category: Others

Content

When the sensitive parameters of the contract are modified, the corresponding events are not recorded, which is not conducive to the supervision of the community and users.

Code location:projects/pottery/contracts/RandomNumberGenerator.sol #L61-86

```
    function setKeyHash(bytes32 _keyHash) external onlyOwner {
        require(_keyHash != bytes32(0), "zero bytes");
        keyHash = _keyHash;
    }

    /**
     * @notice Change the subscription id
     * @param _subId: new subscription id
     */
    function setSubId(uint64 _subId) external onlyOwner {
        s_subscriptionId = _subId;
    }

    /**
     * @notice Set the address for the PancakeSwapLottery
     * @param _potteryDraw: address of the PancakeSwap lottery
     */
    function setPotteryDraw(address _potteryDraw) external onlyOwner {
        require(_potteryDraw != address(0), "zero address");
        potteryDraw = _potteryDraw;
    }

    function setCallBackGasLimit(uint32 _gasLimit) external onlyOwner {
        require(_gasLimit > 100000, "zero gas limit");
        callbackGasLimit = _gasLimit;
    }
```

Solution

It is recommended to add corresponding event records.

Status

Fixed

[N2] [Suggestion] Business logic problem

Category: Design Logic Audit

Content

The contract does not check whether the incoming address and ID exist. If the wrong data is passed in in the actual operation, it will lead to waste of resources.

Code location:projects/pottery/contracts/PancakeSwapPotteryDraw.sol #L149-213

```
    function startDraw(address _vault) external override onlyKeeperOrOwner {
        Pottery.Pot storage pot = pots[_vault];
        require(pot.numOfDraw < NUM_OF_DRAW, "over draw limit");
        require(timeToDraw(_vault), "too early to draw");
        if (pot.startDraw) {
            Pottery.Draw memory draw = draws[pot.lastDrawId];
            require(draw.closeDrawTime != 0, "last draw has not closed");
        }
        uint256 prize = pot.totalPrize / NUM_OF_DRAW;
        uint256 requestId = rng.requestRandomWords(NUM_OF_WINNER, _vault);
        uint256 drawId = draws.length;
        draws.push(
            Pottery.Draw({
                requestId: requestId,
                vault: PancakeSwapPotteryVault(_vault),
                startDrawTime: block.timestamp,
                closeDrawTime: 0,
                winners: new address[](NUM_OF_WINNER),
                prize: prize
            })
        );

        pot.lastDrawId = drawId;
        if (!pot.startDraw) pot.startDraw = true;
        emit StartDraw(drawId, _vault, requestId, prize, block.timestamp,
msg.sender);
    }

    function forceRequestDraw(address _vault) external override onlyOwner {
        Pottery.Pot storage pot = pots[_vault];
        Pottery.Draw storage draw = draws[pot.lastDrawId];
        require(address(draw.vault) != address(0), "draw not exist");
        require(draw.startDrawTime != 0 && draw.closeDrawTime == 0, "draw has
closed");
        require(!rng.fulfillRequest(draw.requestId), "request has fulfilled");
        uint256 requestId = rng.requestRandomWords(NUM_OF_WINNER, _vault);

        draw.requestId = requestId;

        emit StartDraw(pot.lastDrawId, _vault, requestId, draw.prize,
block.timestamp, msg.sender);
    }

    function closeDraw(uint256 _drawId) external override onlyKeeperOrOwner {
        Pottery.Draw storage draw = draws[_drawId];
        require(draw.startDrawTime != 0, "draw has not started");
        require(draw.closeDrawTime == 0, "draw has closed");
        draw.closeDrawTime = block.timestamp;

        require(draw.requestId == rng.getLatestRequestId(address(draw.vault)),
"requestId not match");
        require(rng.fulfillRequest(draw.requestId), "rng request not fulfill");
        uint256[] memory randomWords = rng.getRandomWords(draw.requestId);
        require(randomWords.length == NUM_OF_WINNER, "winning number not match");
        address[] memory winners = draw.vault.draw(randomWords);
        require(winners.length == NUM_OF_WINNER, "winners not match");
        uint256 eachWinnerPrize = draw.prize / NUM_OF_WINNER;
        for (uint256 i = 0; i < NUM_OF_WINNER; i++) {
            draw.winners[i] = winners[i];
            userInfos[winners[i]].reward += eachWinnerPrize;
            userInfos[winners[i]].winCount += 1;
        }

        Pottery.Pot storage pot = pots[address(draw.vault)];
        pot.numOfDraw += 1;

        emit CloseDraw(_drawId, address(draw.vault), draw.requestId, draw.winners,
block.timestamp, msg.sender);
    }
```

Solution

It is recommended to verify the authenticity of incoming data.

Status

Fixed

[N3] [Low] Risk of excessive authority

Category: Authority Control Vulnerability

Content

The Owner has the right to modify the address of the contract to any address.

Code location:projects/pottery/contracts/PancakeSwapPotteryDraw.sol #L271-290

```
    function setVaultFactory(address _factory) public onlyOwner {
        require(_factory != address(0), "zero address");
        vaultFactory = IPotteryVaultFactory(_factory);

        emit SetVaultFactory(msg.sender, _factory);
    }

    function setKeeper(address _keeper) public onlyOwner {
        require(_keeper != address(0), "zero address");
        keeper = _keeper;

        emit SetKeeper(msg.sender, _keeper);
    }

    function setTreasury(address _treasury) public onlyOwner {
        require(_treasury != address(0), "zero address");
        treasury = _treasury;

        emit SetTreasury(msg.sender, _treasury);
    }
```

Code location:projects/pottery/contracts/PancakeSwapPotteryVault.sol #L279-284

```
    function setKeeper(address _keeper) external onlyOwner {
        require(_keeper != address(0), "zero address");
        keeper = _keeper;

        emit SetKeeper(msg.sender, _keeper);
    }
```

Code location:projects/pottery/contracts/PotteryKeeper.sol #L120-132

```
    function setKeeperRegistry(address _registry) public onlyOwner {
        require(_registry != address(0), "zero address");
        keeperRegistry = _registry;

        emit SetKeeperRegistry(_registry, msg.sender);
    }

    function setPotteryDraw(address _potteryDraw) public onlyOwner {
        require(_potteryDraw != address(0), "zero address");
        potteryDraw = IPancakeSwapPotteryDraw(_potteryDraw);

        emit SetPotteryDraw(_potteryDraw, msg.sender);
    }
```

Solution

It is recommended to hand over permissions to timelock management, at least multi-signature should be used.

Status

Confirmed; The project team will use multi-signature wallet for Owner.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002207110002	SlowMist Security Team	2022.07.04 - 2022.07.11	Low Risk

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 low risk, and 2 suggestion vulnerabilities. And 1 low risk were confirmed; 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.