Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2024.08.19, the SlowMist security team received the PingPong team's security audit application for PingPong - MiningShareFactory, 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
		Function Return Value Security Audit
		External Call Function Security Audit

Serial Number	Audit Class	Audit Subclass
7	Security Design 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

The MiningShareFactory contract is designed to manage the creation and distribution of mining revenue share Non-Fungible Tokens (NFTs). These NFTs represent ownership stakes in different mining rounds, allowing holders to claim a portion of the revenue generated by those rounds.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Lack of non-zero	Others	Suggestion	Fixed

NO	Title	Category	Level	Status
	address check			
N2	Missing the event records	Others	Suggestion	Fixed
N3	Missing ERC20 return value check	Others	Suggestion	Fixed
N4	Redundant code	Others	Information	Acknowledged
N5	roundId lacks an upper limit restriction	Integer Overflow and Underflow Vulnerability	Suggestion	Acknowledged
N6	Lack of Time Check when creating a round	Integer Overflow and Underflow Vulnerability	Suggestion	Fixed
N7	Risk of excessive authority	Authority Control Vulnerability Audit	Medium	Acknowledged

4 Code Overview

4.1 Contracts Description

Audit Version:

https://github.com/pingpong-build/pingpong-contracts/blob/main/src/MiningShareFactory.sol

commit: f2652f89f60821bd646b207bc3e89dfd30790545

Fixed Version:

https://github.com/pingpong-build/pingpong-contracts/blob/main/src/MiningPassFactory.sol

commit: 18fa437b0d7c179c0fab74ee273281fc94608bc5

The main network address of the contract is as follows:

MiningPassFactory: https://polygonscan.com/address/0x793c94c1e45f9bb5801e08898ff16743a8561bcd

4.2 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

MiningShareFactory			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC721
setFundCollector	External	Can Modify State	onlyRole
createRound	External	Can Modify State	onlyRole
setWhitelist	External	Can Modify State	onlyRole
setBaseURI	External	Can Modify State	onlyRole
mint	External	Can Modify State	nonReentrant
batchMint	External	Can Modify State	nonReentrant
_mintShares	Internal	Can Modify State	-
getRoundIdFromShareId	Public	-	-
tokenURI	Public	-	-
supportsInterface	Public	-	-

4.3 Vulnerability Summary

[N1] [Suggestion] Lack of non-zero address check

Category: Others

Content

In the MiningShareFactory contract, when setting critical address variables in contracts, there is a lack of checks for non-zero addresses. If key address variables are mistakenly set to zero addresses, it could potentially lead to unexpected errors, such as token loss.

Code Location:

src/MiningShareFactory.sol#L79-94

```
    constructor(address _usdtToken, address _fundCollector) ERC721("Mining Share",
"MS") {
```

```
        usdtToken = IERC20(_usdtToken);
        fundCollector = _fundCollector;

        ...
    }

    ...
    function setFundCollector(address _newCollector) external
onlyRole(DEFAULT_ADMIN_ROLE) {
        fundCollector = _newCollector;
        emit FundCollectorUpdated(_newCollector);
    }
```

Solution

It is recommended that an address non-zero check should be added.

Status

Fixed

[N2] [Suggestion] Missing the event records

Category: Others

Content

In the MiningShareFactory contracts, the operator role can modify some sensitive parameters, but there are no event logs in these functions.

Code Location:

src/MiningShareFactory.sol#L128-139

```
    function setWhitelist(uint256 _roundId, address[] calldata _addresses) external
onlyRole(OPERATOR_ROLE) {
        Round storage round = rounds[_roundId];
        for (uint256 i = 0; i < _addresses.length; i++) {
            round.whitelist[_addresses[i]] = true;
        }
    }

    function setBaseURI(string memory _baseURI) external onlyRole(OPERATOR_ROLE) {
        baseURI = _baseURI;
    }
```

Solution

It is recommended to record events when sensitive parameters are modified for self-inspection or community review.

Status

Fixed

[N3] [Suggestion] Missing ERC20 return value check

Category: Others

Content

The `ERC20.transferFrom()` function returns a boolean value indicating success. This parameter needs to be checked for success. Some tokens do not revert if the transfer failed but return false instead.

The `_mintShares` function does not check the return value of this function.

Code Location:

src/MiningShareFactory.sol#L171

```
    function _mintShares(uint256 _roundId, uint256 _quantity) internal {
        ...

        usdtToken.transferFrom(msg.sender, fundCollector, totalCost);

        ...
    }
```

Solution

It is recommended to use SafeERC20 library or add a check of the return value.

Status

Fixed

[N4] [Information] Redundant code

Category: Others

Content

In the MiningShareFactory contracts, The parameters roundType and miningDays are not used in the specific business logic.

Code Location:

src/MiningShareFactory.sol#L103-123

```
    struct Round {
        uint256 roundType;        // Type of the round
        uint256 totalShares;      // Total number of shares available in this round
        uint256 pricePerShare;    // Price per share in USDT
        uint256 startTime;        // Start time of the round
        uint256 endTime;          // End time of the round
        uint256 whitelistEndTime; // End time for whitelist minting
        mapping(address => bool) whitelist; // Whitelist of addresses
        uint256 mintedCount;      // Number of shares minted in this round
        uint256 miningDays;       // The number of days for which this round's revenue
is allocated
    }

    function createRound(
        uint256 _roundType,
        uint256 _totalShares,
        uint256 _pricePerShare,
        uint256 _startTime,
        uint256 _endTime,
        uint256 _whitelistEndTime,
        uint256 _miningDays
    ) external onlyRole(OPERATOR_ROLE) {
        roundCount++;
        Round storage newRound = rounds[roundCount];
        newRound.roundType = _roundType;

        ...

        newRound.miningDays = _miningDays;

        ...
    }
```

Solution

N/A

Status

Acknowledged; The project team's response: These parameters will be considered for other business use cases.

[N5] [Suggestion] roundId lacks an upper limit restriction

Category: Integer Overflow and Underflow Vulnerability

Content

In the MiningShareFactory contracts, the `roundId` is incremented continuously when the `createRound` function adds new rounds, and it is involved in the calculation of the `shareId` in the `_mintShares` function. Since the calculation for the `shareId` uses bitwise operations, there might be an overflow issue when the `roundId` becomes sufficiently large, potentially causing the calculated `shareId` to exceed expectations.

Code Location:

src/MiningShareFactory.sol#L112

```
    function createRound(
        uint256 _roundType,
        uint256 _totalShares,
        uint256 _pricePerShare,
        uint256 _startTime,
        uint256 _endTime,
        uint256 _whitelistEndTime,
        uint256 _miningDays
    ) external onlyRole(OPERATOR_ROLE) {
        roundCount++;
        Round storage newRound = rounds[roundCount];

        ...
    }
```

Solution

Although it may be difficult in practice to increment the roundId to a value high enough to cause an overflow, it is recommended for safety to impose an upper limit on the roundId.

Status

Acknowledged;

The project team's response: After the project is actually running, the number of rounds might not be too high, so we are not considering this issue for now.

[N6] [Suggestion] Lack of Time Check when creating a round

Category: Integer Overflow and Underflow Vulnerability

Content

In the MiningShareFactory contracts, when the operator role calls the createRound function to create a new round, the start time, end time, and whitelist sale end time for that round are set. However, there's a lack of validation for these three parameters. If the start time is greater than the end time, or if the whitelist sale end time exceeds the range between the start and end times, or if the end time is less than the current time, it could potentially lead to unexpected errors.

Code Location:

src/MiningShareFactory.sol#L103-123

```
    function createRound(
        uint256 _roundType,
        uint256 _totalShares,
        uint256 _pricePerShare,
        uint256 _startTime,
        uint256 _endTime,
        uint256 _whitelistEndTime,
        uint256 _miningDays
    ) external onlyRole(OPERATOR_ROLE) {
        ...

        newRound.startTime = _startTime;
        newRound.endTime = _endTime;

        ...

        newRound.whitelistEndTime = _whitelistEndTime;

        ...
    }
```

Solution

It is recommended to add checks for these three time parameters to ensure that the start time is less than the end time, the end time is not less than the current time, and the whitelist sale time falls between the start and end times.

Status

Fixed

[N7] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

In the MiningShareFactory contracts, the admin role can invoke the setFundCollector function to set the address for receiving payment tokens during NFT minting. The operator role can create a new minting round by calling the createRound function and setting relevant information for the round, such as the NFT unit price, total quantity to be minted in the round, whitelist addresses, etc. If the private key for a core role is lost, it may result in a loss of funds for the contract.

Code Location:

src/MiningShareFactory.sol

```
    function setFundCollector(address _newCollector) external
onlyRole(DEFAULT_ADMIN_ROLE) {
        ...
    }

    function createRound(
        uint256 _roundType,
        uint256 _totalShares,
        uint256 _pricePerShare,
        uint256 _startTime,
        uint256 _endTime,
        uint256 _whitelistEndTime,
        uint256 _miningDays
    ) external onlyRole(OPERATOR_ROLE) {
        ...
    }
    function setWhitelist(uint256 _roundId, address[] calldata _addresses) external
onlyRole(OPERATOR_ROLE) {
        ...
    }

    function setBaseURI(string memory _baseURI) external onlyRole(OPERATOR_ROLE) {
        ...
    }
```

Solution

In the short term, during the early stages of the project, the protocol may need to frequently set various parameters to ensure the stable operation of the protocol. Therefore, transferring the owner ownership to a multisig management

can effectively solve the single-point risk, but it cannot mitigate the excessive privilege risk. In the long run, after the protocol stabilizes, transferring the owner ownership to community governance and executing through a timelock can effectively mitigate the excessive privilege risk and increase the community users' trust in the protocol.

Status

Acknowledged;

The project team's response: For the initial phase of the project's operation, we will temporarily maintain the existing permissions. Once the project runs smoothly later on, management will be transferred to a multi-signature address.

Update: The permissions of the admin and operator roles have been transferred to be managed by a multisig contract. Here are the tx transactions for the permission transfers:

https://polygonscan.com/tx/0x17a8a207a4f3463e5d50c2c55cfa160fc6368e6328a7216f52a303c493fef8a2

https://polygonscan.com/tx/0x19f98af0e50744445995d6a9a77a6765088dc030b34fdf6c771d3f5bd66b78a7

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002408190001	SlowMist Security Team	2024.08.19 - 2024.08.19	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 medium risk, 5 suggestion vulnerabilities and 1 information. All the findings were fixed and acknowledged. The project has been deployed on the mainnet, and the permissions of the core roles have been transferred to be managed by the multisig contract.

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