Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.09.20, the SlowMist security team received the NFTReward team's security audit application for NFTReward, 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

Project address: https://github.com/MultichainDAO/NFTReward

commit: 926d2800ea4aafb93057c98a3dd7fa6d3afc4b1d

Audit scope:

- NFTReward/contracts/Administrable.sol
- NFTReward/contracts/NFTFactory.sol
- NFTReward/contracts/RewardPortal.sol
- NFTReward/contracts/SlowRelease.sol

Fixed Version

Project address: https://github.com/MultichainDAO/NFTReward

commit: 1148509d6993b8f98a514bb8d27784430002418c

Audit scope:

- NFTReward/contracts/Administrable.sol
- NFTReward/contracts/NFTFactory.sol
- NFTReward/contracts/RewardPortal.sol
- NFTReward/contracts/SlowRelease.sol

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

NO	Title	Category	Level	Status
N2	Risk of excessive authority	Authority Control Vulnerability	Low	Fixed
N3	Business Logic Defect Suggestion	Design Logic Audit	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:

Administrable			
Function Name	Visibility	Mutability	Modifiers
setAdmin	Internal	Can Modify State	-
transferAdmin	External	Can Modify State	onlyAdmin
acceptAdmin	External	Can Modify State	-

RewardPortal			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer

RewardPortal			
setFactory_SlowRelease	Public	Can Modify State	onlyOwner
setFactory_VE	Public	Can Modify State	onlyOwner
deployRewardHandler_SlowRelease	Public	Payable	onlyOwner
deployRewardHandler_VEShare	Public	Payable	onlyOwner
claimable	External	-	-
claimReward	External	Can Modify State	-

RewardHandler_SlowRelease			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setReward	External	Can Modify State	onlyAdmin
withdrawReward	External	Can Modify State	onlyAdmin
claimable	Public	-	-
claimReward	External	Can Modify State	-

RewardHandler_Factory_SlowRelease			
Function Name	Visibility	Mutability	Modifiers
getBytecode	Public	-	-
create	Public	Payable	-

4.3 Vulnerability Summary

[N1] [Suggestion] Missing event record

Category: Others

Content

Modifying sensitive parameters in the contract lacks corresponding event records, which is not conducive to the supervision of the community and users.

Code location: NFTReward/contracts/NFTFactory.sol #L42-52

```
    function revokeWhitelist(address[] calldata accounts) external {
        require(owner == msg.sender);
        for (uint256 i = 0; i < accounts.length; i++) {
            whitelist[accounts[i]] == false;
        }
    }

    function transferOwner(address to) external {
        require(owner == msg.sender);
        owner = to;
    }
```

Code location: NFTReward/contracts/RewardPortal.sol #L42-48

```
    function setFactory_SlowRelease(address factory_) public onlyOwner {
        factory_slowRelease = factory_;
    }

    function setFactory_VE(address factory_) public onlyOwner {
        factory_veShare = factory_;
    }
```

Code location: NFTReward/contracts/SlowRelease.sol #L35-39

```
    function setReward(uint256[] calldata tokenIds, uint256 amount, uint256
startTime, uint256 endTime) onlyAdmin external {
        for (uint i = 0; i < tokenIds.length; i++) {
            rewardInfo[tokenIds[i]] = Info(amount, uint64(startTime),
uint64(endTime));
```

```
        }
    }
```

Code location: NFTReward/contracts/SlowRelease.sol #L60-64

```
    function claimReward(uint256 tokenId) override external {
        uint256 amount = claimable(tokenId);
        lastClaimTime[tokenId] = block.timestamp;
        IERC20(rewardToken).transfer(IERC721(nft).ownerOf(tokenId), amount);
    }
```

Code location: NFTReward/contracts/SlowRelease.sol #L73-89

```
    function create(address nft, address rewardToken, uint salt, address admin)
 payable public returns (address) {
        address addr;
        bytes memory bytecode = getBytecode(nft, rewardToken, admin);
        assembly {
            addr := create2(
                callvalue(),
                add(bytecode, 0x20),
                mload(bytecode),
                salt
            )

            if iszero(extcodesize(addr)) {
                revert(0, 0)
            }
        }
        return addr;
    }
```

Solution

It is recommended to add corresponding event records.

Status

Fixed

[N2] [Low] Risk of excessive authority

Category: Authority Control Vulnerability

Content

The owner has the risk of excessive authorization, and can add/delete the whitelist arbitrarily, and the roles in the whitelist have the right to receive NFT.

Code location: NFTReward/contracts/NFTFactory.sol #L34-47

```
    function setWhitelist(address[] calldata accounts) external {
        require(owner == msg.sender);
        for (uint256 i = 0; i < accounts.length; i++) {
            whitelist[accounts[i]] = true;
        }
        emit Whitelist(accounts);
    }

    function revokeWhitelist(address[] calldata accounts) external {
        require(owner == msg.sender);
        for (uint256 i = 0; i < accounts.length; i++) {
            whitelist[accounts[i]] == false;
        }
    }
```

The Owner role has the risk of over-authorization, and the Owner has the right to set the factory_slowRelease and factory_veShare addresses to any address.

Code location: NFTReward/contracts/RewardPortal.sol #L42-48

```
    function setFactory_SlowRelease(address factory_) public onlyOwner {
        factory_slowRelease = factory_;
    }

    function setFactory_VE(address factory_) public onlyOwner {
        factory_veShare = factory_;
    }
```

The Admin role has the risk of over-authorization. The Admin can modify the reward parameter to any value at any time.

Code location: NFTReward/contracts/SlowRelease.sol #L35-39

```
    function setReward(uint256[] calldata tokenIds, uint256 amount, uint256 
startTime, uint256 endTime) onlyAdmin external {
        for (uint i = 0; i < tokenIds.length; i++) {
            rewardInfo[tokenIds[i]] = Info(amount, uint64(startTime), 
uint64(endTime));
        }
    }
```

There is a risk of over-authorization with the Admin role, which can withdraw any number of reward tokens.

Code location: NFTReward/contracts/SlowRelease.sol #L41-44

```
    function withdrawReward(uint256 amount) onlyAdmin external {
        IERC20(rewardToken).transfer(msg.sender, amount);
        emit LogWithdrawReward(amount);
    }
```

Solution

1. It is recommended to transfer roles with excessive authorization risk to multi-signature wallet management.
2. When setting the whitelist, you should check whether the address has received the whitelist airdrop. If you have received it, you cannot set it again.
3. When adjusting the reward parameters, you should check whether the reward parameter settings are valid. The reward parameters cannot be modified during the validity period of the reward parameters.

Status

Fixed; The project team made the following fixes:

1. Added a check on whether the address has received the whitelist airdrop in the setWhitelist function.
2. Removed setFactory_SlowRelease and setFactory_VE functions.

3. In the setReward function, the restriction that the reward parameters are not allowed to be modified while the reward is in effect has been added.
4. Removed withdrawReward function.

[N3] [Suggestion] Business Logic Defect Suggestion

Category: Design Logic Audit

Content

It is recommended to add the acceptOwner logic after calling the transferOwner function. The new Owner needs to accept the permissions manually, which can prevent the loss of permissions due to operational errors.

Code location: NFTReward/contracts/NFTFactory.sol #L49-52

```
function transferOwner(address to) external {
    require(owner == msg.sender);
    owner = to;
}
```

The Owner role can grant the target address createNFT permission through the setCreator function, but there is no function to deny the permission. In extreme cases, it cannot respond and withdraw the permission.

Code location: NFTReward/contracts/NFTFactory.sol #L132-134

```
function setCreator(address creator) external onlyOwner {
    isCreator[creator] = true;
}
```

Solution

1. It is recommended to add acceptOwner logic to prevent the loss of permissions due to operational errors.
2. It is recommended to add revokeCreator logic. In extreme cases, the Owner can respond in time to deny the permission.

Status

Fixed; The project team has added acceptOwner and revokeCreator logic.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002209210001	SlowMist Security Team	2022.09.20 - 2022.09.21	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 low risk and 2 suggestions. All the 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