Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.05.23, the SlowMist security team received the team's security audit application for ROTL, 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:

ray_contract_eth_new.zip

55116efd0f22d3c62e6674250c96b822efcd62f1d7ceaff2a9c1e2538e3e0aa0

ray_contract_eth 0526.zip

1652becd2cceea18aff3d89bfdeb348a3412a94cd642b9641fe507ebba5e200d

Fixed version:

https://github.com/dominusrotl/rotl-contract

commit: af8d0c310071767a8b7a9d57ed89fb8f4fadbfcc

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Excessive authority issue	Authority Control Vulnerability	Medium	Fixed
N2	Pausable is not implemented	Others	Low	Fixed
N3	Missing event record	Others	Suggestion	Fixed
N4	Dev address setting enhancement suggestion	Others	Suggestion	Ignored
N5	Variable not used	Others	Suggestion	Fixed

4 Code Overview

4.1 Contracts Description

The main network address of the contract is as follows:

https://etherscan.io/address/0x18affb2a5ead3fae42e34668871ec9b5e5e713e0

https://etherscan.io/address/0xa2e470b334777c7b6a0d8a066e4c0368695453d9

4.2 Visibility Description

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

ROTLMint			
Function Name	Visibility	Mutability	Modifiers
setAddress	External	Can Modify State	onlyOwner
setMerkleRoot	External	Can Modify State	onlyOwner
setRound	External	Can Modify State	onlyOwner
setRoundInfo	External	Can Modify State	onlyOwner
isEnable	External	-	-
__isEnable	Internal	-	-
getRemainCount	External	-	-
__getRemainCount	Internal	-	-
getCurrentRoundInfo	External	-	-
__getCurrentRoundInfo	Internal	-	-
getCurrentRound	External	-	-
mint	External	Payable	whenNotPaused

ROTLMint			
__mint	Internal	Can Modify State	-
withdraw	External	Can Modify State	onlyOwner

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	-	-

ROTL			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC721A
mint	External	Payable	onlyRole
addMinter	External	Can Modify State	onlyRole

ROTL			
setBaseTokenURI	External	Can Modify State	onlyRole
_baseURI	Internal	-	-
supportsInterface	Public	-	-

ERC721A			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
_startTokenId	Internal	-	-
totalSupply	Public	-	-
_totalMinted	Internal	-	-
supportsInterface	Public	-	-
balanceOf	Public	-	-
_numberMinted	Internal	-	-
_numberBurned	Internal	-	-
_getAux	Internal	-	-
_setAux	Internal	Can Modify State	-
_ownershipOf	Internal	-	-
ownerOf	Public	-	-
name	Public	-	-
symbol	Public	-	-

ERC721A			
tokenURI	Public	-	-
_baseURI	Internal	-	-
approve	Public	Can Modify State	-
getApproved	Public	-	-
setApprovalForAll	Public	Can Modify State	-
isApprovedForAll	Public	-	-
transferFrom	Public	Can Modify State	-
safeTransferFrom	Public	Can Modify State	-
safeTransferFrom	Public	Can Modify State	-
_exists	Internal	-	-
_safeMint	Internal	Can Modify State	-
_safeMint	Internal	Can Modify State	-
_mint	Internal	Can Modify State	-
_transfer	Private	Can Modify State	-
_burn	Internal	Can Modify State	-
_burn	Internal	Can Modify State	-
_checkContractOnERC721Received	Private	Can Modify State	-
_beforeTokenTransfers	Internal	Can Modify State	-
_afterTokenTransfers	Internal	Can Modify State	-
_msgSenderERC721A	Internal	-	-

ERC721A			
_toString	Internal	-	-

AccessControlEnumerable			
Function Name	Visibility	Mutability	Modifiers
supportsInterface	Public	-	-
getRoleMember	Public	-	-
getRoleMemberCount	Public	-	-
_grantRole	Internal	Can Modify State	-
_revokeRole	Internal	Can Modify State	-

AccessControl			
Function Name	Visibility	Mutability	Modifiers
supportsInterface	Public	-	-
hasRole	Public	-	-
_checkRole	Internal	-	-
_checkRole	Internal	-	-
getRoleAdmin	Public	-	-
grantRole	Public	Can Modify State	onlyRole
revokeRole	Public	Can Modify State	onlyRole
renounceRole	Public	Can Modify State	-
_setupRole	Internal	Can Modify State	-

AccessControl			
_setRoleAdmin	Internal	Can Modify State	-
_grantRole	Internal	Can Modify State	-
_revokeRole	Internal	Can Modify State	-

ERC165			
Function Name	Visibility	Mutability	Modifiers
supportsInterface	Public	-	-

Pausable			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
paused	Public	-	-
_pause	Internal	Can Modify State	whenNotPaused
_unpause	Internal	Can Modify State	whenPaused

4.3 Vulnerability Summary

[N1] [Medium] Excessive authority issue

Category: Authority Control Vulnerability

Content

In the ROTL contract, the DEFAULT_ADMIN_ROLE can set the minter role, the minter role can mint ERC721A tokens arbitrarily and the minter role is entitled to free mint without going through each rounds.

Code location:

ROTL.sol#L18-25

```
    function mint(address to, uint256 quantity) external payable
onlyRole(MINTER_ROLE) {
        // _safeMint's second argument now takes in a quantity, not a tokenId.
        _safeMint(to, quantity);
    }

    function addMinter(address minter) external onlyRole(DEFAULT_ADMIN_ROLE) {
        _setupRole(MINTER_ROLE, minter);
    }
```

Solution

It is recommended to limit the total amount of the underlying mint contract.

Status

Fixed; After communication with the project team, they expressed that permissions will added to Multisig Owner after Minting and delete existing permissions. And after the fix, the ROTL contract limits the total supply of the NFT.

[N2] [Low] Pausable is not implemented

Category: Others

Content

In the ROLTMint contract, it heritates the Pausable contract, but there is no pause and unpause function implemented. That means the value ot the _paused is false and can not be changed. Which will impact the __isEnable function and whenNotPaused modifier.

Code location:

ROTLMint.sol#59-65, 92

```
    function isEnable() external view returns (bool) {
        return __isEnable();
    }
    function __isEnable() internal view returns (bool) {
        return 0 < __getRemainCount() && !paused();
    }

    function mint(uint256 round, uint256 count, bytes32[] calldata merkleProof)
external payable whenNotPaused {
......
    }
```

Solution

It’s recommended to complete and implement the pause and unpause functions.

Status

Fixed

[N3] [Suggestion] Missing event record

Category: Others

Content

1.In the ROTLMint contract, the owner role can set the _nft, _merkleRoot, _currentRound, price, maxCount, onceMaxCount, addressMaxCount, and startBlock values through the setAddress, setMerkleRoot, setRound, and setRoundInfo functions. But there are no no events logging performed.

Code location:

ROTLMint.sol#L29-57

```
    function setAddress(address kip17) external onlyOwner {
        _nft = ROTL(kip17);
    }

    function setMerkleRoot(bytes32 root) external onlyOwner {
        _merkleRoot = root;
    }

    function setRound(
        uint256 round
    ) external onlyOwner {
        _currentRound = round;
    }

    function setRoundInfo(
        uint256 round,
        uint256 price,
        uint256 maxCount,
        uint256 onceMaxCount,
        uint256 addressMaxCount,
        uint256 startBlock
    ) external onlyOwner {
        Round storage v = _round[round];
        v._price = price;
        v._maxCount = maxCount;
        v._onceMaxCount = onceMaxCount;
        v._addressMaxCount = addressMaxCount;
        v._startBlock = startBlock;
    }
```

2.In the ROTL contract, the DEFAULT_ADMIN_ROLE can set the _baseTokenURI through the setBaseTokenURI function, but there are no no events logging performed.

Code location:

ROTL.sol#L27-29

```
    function setBaseTokenURI(string memory uri) external onlyRole(DEFAULT_ADMIN_ROLE)
  {
        _baseTokenURI = uri;
    }
```

Solution

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

Status

Fixed

[N4] [Suggestion] Dev address setting enhancement suggestion

Category: Others

Content

In the ROTLMint contract, the owner role can withdraw the native token through the withdraw function. If the owner is an EOA address, in a scenario where the private key is leaked, the team's revenue will be stolen.

Code location:

ROTLMint.sol#L121-123

```
    function withdraw(address payable to, uint256 value) external onlyOwner {
        to.transfer(value);
    }
```

Solution

It is recommended to set the development address as a multi-signature contract to avoid the leakage of private keys and the theft of team rewards.

Status

Ignored

[N5] [Suggestion] Variable not used

Category: Others

Content

In the ROLT contract, the contract defined the _mintContractAddress and _revealIndex value. But these two values are not assigned and can not be set.

Code location:

ROTL.sol#L11, 13

```
contract ROTL is AccessControlEnumerable, ERC721A, ERC721ABurnable, ERC721AQueryable
{
    address _mintContractAddress;
    string _baseTokenURI;
    uint256 _revealIndex;
    ......
    }
```

Solution

It is recommended to assign the values in the constructor or add a function to make the values can be changed. And It is also recommended to clarify the business logic implementation, and if it is redundant code, it is recommended to remove it from the contract.

Status

Fixed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002205260002	SlowMist Security Team	2022.05.23 - 2022.05.26	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, 3 suggestion vulnerabilities. And 1 suggestion vulnerabilities were ignored; All other findings were fixed.

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.