Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.03.24, the SlowMist security team received the Laqira team's security audit application for Laqira NFT marketplace, 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/LaqiraProtocol/Laqira-Collectibles-and-NFT-Marketplace

Commit:

24ff02fb2a182fbce8aa0b7f3a02a2678aca5d33

Fixed Version

Project address:

https://github.com/LaqiraProtocol/Laqira-Collectibles-and-NFT-Marketplace

Commit:

aba48fa099dc71ea508791bf41297ab34bd23591

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Gas optimization	Gas Optimization Audit	Suggestion	Ignored
N2	Unused return	Others	Suggestion	Fixed
N3	Missing event record	Others	Suggestion	Fixed
N4	Missing zero address validation	Others	Suggestion	Fixed

4 Code Overview

4.1 Contracts Description

The main network address of the contract is as follows:

LaqiraNFT:

0x6f695510df417C8B87ef94F3342542bA5d24EBBB

RoyaltiesProvider:

0x18216e3d7f03E39e121c3f5b8A0d8652CB4B57eB

ExchangeNFTConfig:

0x0a1996fA0f704B1f62DD77cE8d6ba16B76017b75

ExchangeNFT:

0x344a5ec33f410081D7290d952AfF1184a7a13F45

Proxy:

0x30635f3F336c98a44490909Ff08AF4086c96Bc9D

0xF79CB10D00803D7Bcdb677CC8a652732141A9286

0xF5A4E19ac27a236D08A2ADB695Ea61012599B1F5

0xE4cAE3079DBC4c689468185Dc36B449012375fD8

4.2 Visibility Description

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

ExchangeNFTConfiguration			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
setSettings	External	Can Modify State	onlyOwner
setNftEnables	Public	Can Modify State	onlyOwner

ExchangeNFTConfiguration			
setNftQuoteEnables	Public	Can Modify State	onlyOwner
transferFeeAddress	Public	Can Modify State	-
batchTransferFeeAddress	Public	Can Modify State	-
setFee	Public	Can Modify State	onlyOwner
batchSetFee	Public	Can Modify State	onlyOwner
setFeeBurnAble	Public	Can Modify State	onlyOwner
batchSetFeeBurnAble	Public	Can Modify State	onlyOwner
setRoyaltiesProvider	Public	Can Modify State	onlyOwner
batchSetRoyaltiesProviders	Public	Can Modify State	onlyOwner
setRoyaltiesBurnable	Public	Can Modify State	onlyOwner
batchSetRoyaltiesBurnable	Public	Can Modify State	onlyOwner
addNft	External	Can Modify State	onlyOwner
nftSettings	External	-	-
checkEnableTrade	External	-	-
whenSettings	External	-	-
getNftQuotes	External	-	-

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

ExchangeNFTs			
setConfig	Public	Can Modify State	onlyOwner
getNftQuotes	Public	-	-
batchReadyToSellToken	External	Can Modify State	-
batchReadyToSellTokenTo	Public	Can Modify State	-
readyToSellToken	External	Can Modify State	-
readyToSellToken	External	Can Modify State	-
readyToSellTokenTo	Public	Can Modify State	nonReentrant
batchSetCurrentPrice	External	Can Modify State	-
setCurrentPrice	Public	Can Modify State	nonReentrant
batchBuyToken	External	Can Modify State	-
batchBuyTokenTo	Public	Can Modify State	-
buyToken	External	Payable	-
_settleTrade	Internal	Can Modify State	-
buyTokenTo	Public	Payable	nonReentrant
batchCancelSellToken	External	Can Modify State	-
cancelSellToken	Public	Can Modify State	nonReentrant
batchBidToken	External	Can Modify State	-
batchBidTokenTo	Public	Can Modify State	-
bidToken	External	Payable	-
bidTokenTo	Public	Payable	nonReentrant

ExchangeNFTs			
batchUpdateBidPrice	External	Can Modify State	-
updateBidPrice	Public	Payable	nonReentrant
getBidByTokenIdAndAddress	Internal	-	-
delBidByTokenIdAndIndex	Internal	Can Modify State	-
sellTokenTo	Public	Can Modify State	nonReentrant
batchCancelBidToken	External	Can Modify State	-
cancelBidToken	Public	Can Modify State	nonReentrant
getAskLength	Public	-	-
getAsks	Public	-	-
getAsksByNFT	External	-	-
getAsksByPage	External	-	-
getUserAsks	Public	-	-
getUserAsksByNFT	External	-	-
getBidsLength	External	-	-
getBids	External	-	-
getUserBids	Public	-	-
getUserBidsByNFT	External	-	-

RoyaltiesProvider			
Function Name	Visibility	Mutability	Modifiers

RoyaltiesProvider			
initialize	Public	Can Modify State	initializer
getRoyalties	External	-	-
setRoyalties	External	Can Modify State	onlyAllowedNFT
setTotalRoyalties	Public	Can Modify State	onlyOwner
setAllowedNFT	Public	Can Modify State	onlyOwner
getAllowedNFT	Public	-	-
getTotalRoyalties	Public	-	-

ERC1967Proxy			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Payable	-
_implementation	Internal	-	-

Proxy			
Function Name	Visibility	Mutability	Modifiers
_delegate	Internal	Can Modify State	-
_implementation	Internal	-	-
_fallback	Internal	Can Modify State	-
<Fallback>	External	Payable	-
<Receive Ether>	External	Payable	-
_beforeFallback	Internal	Can Modify State	-

ERC1967Upgrade			
Function Name	Visibility	Mutability	Modifiers
_getImplementation	Internal	-	-
_setImplementation	Private	Can Modify State	-
_upgradeTo	Internal	Can Modify State	-
_upgradeToAndCall	Internal	Can Modify State	-
_getAdmin	Internal	-	-
_setAdmin	Private	Can Modify State	-
_changeAdmin	Internal	Can Modify State	-

TransparentUpgradeableProxy			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Payable	ERC1967Proxy
admin	External	Can Modify State	ifAdmin
implementation	External	Can Modify State	ifAdmin
changeAdmin	External	Can Modify State	ifAdmin
upgradeTo	External	Can Modify State	ifAdmin
upgradeToAndCall	External	Payable	ifAdmin
_admin	Internal	-	-
_beforeFallback	Internal	Can Modify State	-

LaqiraNFT			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
mint	Public	Payable	-
mintTo	Public	Can Modify State	onlyOwner
burn	Public	Can Modify State	onlyOwner
confirmNFT	Public	Can Modify State	-
rejectNFT	Public	Can Modify State	-
setMintingFeeAmount	Public	Can Modify State	onlyOwner
setAsOperator	Public	Can Modify State	onlyOwner
removeOperator	Public	Can Modify State	onlyOwner
transferAnyBEP20	Public	Can Modify State	onlyOwner
adminWithdrawal	Public	Can Modify State	onlyOwner
setFeeAddress	Public	Can Modify State	onlyOwner
transfer	Public	Can Modify State	-
setRoyaltiesProviderAddress	Public	Can Modify State	onlyOwner
getRoyaltiesProviderAddress	Public	-	-
getFeeAddress	Public	-	-
isOperator	Public	-	-
getPendingRequests	Public	-	-
getRejectedRequests	Public	-	-

LaqiraNFT			
getUserPendingIds	Public	-	-
getUserRejectedIds	Public	-	-
fetchPendingIdDetails	Public	-	-
fetchRejectedIdDetails	Public	-	-
tokenURI	Public	-	-
_baseURI	Internal	-	-
_setTokenURI	Internal	Can Modify State	-
delUintFromArray	Internal	Can Modify State	-

4.3 Vulnerability Summary

[N1] [Suggestion] Gas optimization

Category: Gas Optimization Audit

Content

Using assert will consume the remaining gas when the transaction fails to execute.

Code location:

Laqira-Collectibles-and-NFT-Marketplace/contracts/proxy/ERC1967Proxy.sol #L22-L25

```
    constructor(address _logic, bytes memory _data) payable {
        assert(_IMPLEMENTATION_SLOT ==
bytes32(uint256(keccak256("eip1967.proxy.implementation")) - 1));
        _upgradeToAndCall(_logic, _data, false);
    }
```

Code location:

Laqira-Collectibles-and-NFT-Marketplace/contracts/proxy/TransparentUpgradeableProxy.sol #L34-L41

```
    constructor(
        address _logic,
        address admin_,
        bytes memory _data
    ) payable ERC1967Proxy(_logic, _data) {
        assert(_ADMIN_SLOT == bytes32(uint256(keccak256("eip1967.proxy.admin")) -
1));
        _changeAdmin(admin_);
    }
```

Solution

It is recommended to use require instead of assert to optimize gas.

Status

Ignored; This code is written in the constructor and it will be executed only once by the contract creator. As this code is the original code of OpenZeppelin libraries, so the project team ignored it.

[N2] [Suggestion] Unused return

Category: Others

Content

There is a return value in the setRoyalties function in the RoyaltiesProvider contract, and the function is called here without checking its return value.

Laqira-Collectibles-and-NFT-Marketplace/contracts/LaqiraNFT.sol #L58-L76

```
    function mint(string memory _tokenURI, address[] memory royaltyOwners, uint96[]
memory values) public virtual payable {
        uint256 transferredAmount = msg.value;

        require(transferredAmount >= mintingFee, 'Insufficient paid amount');

        (bool success, ) = feeAddress.call{value: transferredAmount}(new bytes(0));
```

```
        require(success, 'Transfer failed');

        _tokenIds.increment();

        uint256 newTokenId = _tokenIds.current();

        pendingRequests.push(newTokenId);
        _pendingIds[newTokenId].owner = _msgSender();
        _pendingIds[newTokenId].tokenURI = _tokenURI;
        _userPendingIds[_msgSender()].push(newTokenId);

        IRoyaltiesProvider(royaltiesProviderAddress).setRoyalties(newTokenId,
royaltyOwners, values);
    }
```

Solution

It is recommended to check the return value of the setRoyalties function.

Status

Fixed

[N3] [Suggestion] Missing event record

Category: Others

Content

Modifying sensitive parameters in the contract does not log an event.

Laqira-Collectibles-and-NFT-Marketplace/contracts/LaqiraNFT.sol #L123-L133

```
    function setMintingFeeAmount(uint256 _amount) public virtual onlyOwner {
        mintingFee = _amount;
    }

    function setAsOperator(address _operator) public virtual onlyOwner {
        operators[_operator] = true;
    }

    function removeOperator(address _operator) public virtual onlyOwner {
        operators[_operator] = false;
    }
```

Code location:

Laqira-Collectibles-and-NFT-Marketplace/contracts/LaqiraNFT.sol #L145-L147

```
    function setFeeAddress(address _newAddress) public virtual onlyOwner {
        feeAddress = _newAddress;
    }
```

Code location:

Laqira-Collectibles-and-NFT-Marketplace/contracts/LaqiraNFT.sol #L154-L156

```
    function setRoyaltiesProviderAddress(address _royaltiesProviderAddress) public
virtual onlyOwner {
        royaltiesProviderAddress = _royaltiesProviderAddress;
    }
```

Code location:

Laqira-Collectibles-and-NFT-Marketplace/contracts/exchange/ExchangeNFTConfiguration.sol #L55-L57

```
    function setNftEnables(address _nftToken, bool _enable) public virtual override
onlyOwner {
        nftEnables[_nftToken] = _enable;
    }
```

Code location:

Laqira-Collectibles-and-NFT-Marketplace/contracts/exchange/ExchangeNFTs.sol #L74-L77

```
    function setConfig(address _config) public virtual onlyOwner {
        require(address(config) != _config, 'forbidden');
        config = IExchangeNFTConfiguration(_config);
    }
```

Code location:

Laqira-Collectibles-and-NFT-Marketplace/contracts/exchange/RoyaltiesProvider.sol #L40-L46

```
function setTotalRoyalties(uint96 _value) public virtual onlyOwner {
    totalRoyalties = _value;
}

function setAllowedNFT(address _NFTAddress) public virtual onlyOwner {
    allowedNFT = _NFTAddress;
}
```

Solution

It is recommended to record events when modifying sensitive parameters.

Status

Fixed

[N4] [Suggestion] Missing zero address validation

Category: Others

Content

Missing zero address validation when setting the address in the function.

Code location:

Laqira-Collectibles-and-NFT-Marketplace/contracts/LaqiraNFT.soln #L123-L125

```
function setMintingFeeAmount(uint256 _amount) public virtual onlyOwner {
    mintingFee = _amount;
}
```

Code location:

Laqira-Collectibles-and-NFT-Marketplace/contracts/LaqiraNFT.soln #L145-L147

```
function setFeeAddress(address _newAddress) public virtual onlyOwner {
    feeAddress = _newAddress;
}
```

Code location:

Laqira-Collectibles-and-NFT-Marketplace/contracts/LaqiraNFT.soln #L154-L156

```
    function setRoyaltiesProviderAddress(address _royaltiesProviderAddress) public
virtual onlyOwner {
        royaltiesProviderAddress = _royaltiesProviderAddress;
    }
```

Code location:

Laqira-Collectibles-and-NFT-Marketplace/contracts/LaqiraNFT.sol #L134-L147

```
    function setRoyaltiesProvider(
        address _nftToken,
        address _quoteToken,
        address _royaltiesProvider
    ) public virtual override onlyOwner {
        emit SetRoyaltiesProvider(
            _nftToken,
            _quoteToken,
            _msgSender(),
            royaltiesProviders[_nftToken][_quoteToken],
            _royaltiesProvider
        );
        royaltiesProviders[_nftToken][_quoteToken] = _royaltiesProvider;
    }
```

Code location:

Laqira-Collectibles-and-NFT-Marketplace/contracts/exchange/RoyaltiesProvider.sol #L44-L46

```
    function setAllowedNFT(address _NFTAddress) public virtual onlyOwner {
        allowedNFT = _NFTAddress;
    }
```

Solution

It is recommended to add zero address validation.

Status

Fixed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002204120004	SlowMist Security Team	2022.03.24 - 2022.04.12	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 4 suggestion vulnerabilities. And 1 suggestion vulnerabilities were ignored. All the other findings were been fixed. The code was 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