Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

On 2023.02.28, the SlowMist security team received the hashkey team's security audit application for cross-chain, 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

Serial Number	Audit Class	Audit Subclass
		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

This is the cross-chain of hashkey.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Risk of excessive authority	Authority Control Vulnerability Audit	Low	Acknowledged

4 Code Overview

4.1 Contracts Description

Codebase:

Audit Version

File Name: sync.zip

Hash(sha256): deed2287387abb6817b2acc002bc9bc5f5118b7c9721c2481e636531c95ef91d

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:

LzAppUpgradeable			
Function Name	Visibility	Mutability	Modifiers
__LzAppUpgradeable_init	Internal	Can Modify State	onlyInitializing
__LzAppUpgradeable_init_unchained	Internal	Can Modify State	onlyInitializing
lzReceive	Public	Can Modify State	-
_blockingLzReceive	Internal	Can Modify State	-
_lzSend	Internal	Can Modify State	-
_checkGasLimit	Internal	-	-
getGasLimit	Public	-	-

LzAppUpgradeable			
getConfig	External	-	-
setConfig	External	Can Modify State	onlyOwner
setSendVersion	External	Can Modify State	onlyOwner
setReceiveVersion	External	Can Modify State	onlyOwner
forceResumeReceive	External	Can Modify State	onlyOwner
setTrustedRemote	External	Can Modify State	onlyOwner
setMinDstGasLookup	External	Can Modify State	onlyOwner
isTrustedRemote	External	-	-

NonblockingLzAppUpgradeable			
Function Name	Visibility	Mutability	Modifiers
__NonblockingLzAppUpgradeable_init	Internal	Can Modify State	onlyInitializing
__NonblockingLzAppUpgradeable_init_unchained	Internal	Can Modify State	onlyInitializing
_blockingLzReceive	Internal	Can Modify State	-
nonblockingLzReceive	Public	Can Modify State	-
_nonblockingLzReceive	Internal	Can Modify State	-
retryMessage	Public	Payable	-

DidSync			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
setAdapterParams	Public	Can Modify State	onlyOwner

DidSync			
sync	Public	Payable	-
_nonblockingLzReceive	Internal	Can Modify State	-
estimateSendFee	Public	-	-
_validate	Internal	-	-

SyncStorage			
Function Name	Visibility	Mutability	Modifiers

EternalStorageProxy			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Payable	TransparentUpgradeableProxy

4.3 Vulnerability Summary

[N1] [Low] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

The authority management in the contract is too centralized, and the Owner role has the right to modify the configuration parameters in the contract.

Code location: contracts/lzApp/LzAppUpgradeable.sol #L68-94

```
    function setConfig(uint16 _version, uint16 _chainId, uint _configType, bytes
calldata _config) external override onlyOwner {
        lzEndpoint.setConfig(_version, _chainId, _configType, _config);
    }

    function setSendVersion(uint16 _version) external override onlyOwner {
        lzEndpoint.setSendVersion(_version);
    }

    function setReceiveVersion(uint16 _version) external override onlyOwner {
```

```
        lzEndpoint.setReceiveVersion(_version);
    }

    function forceResumeReceive(uint16 _srcChainId, bytes calldata _srcAddress)
external override onlyOwner {
        lzEndpoint.forceResumeReceive(_srcChainId, _srcAddress);
    }

    // allow owner to set it multiple times.
    function setTrustedRemote(uint16 _srcChainId, bytes calldata _srcAddress)
external onlyOwner {
        trustedRemoteLookup[_srcChainId] = _srcAddress;
        emit SetTrustedRemote(_srcChainId, _srcAddress);
    }

    function setMinDstGasLookup(uint16 _dstChainId, uint _type, uint _dstGasAmount)
external onlyOwner {
        require(_dstGasAmount > 0, "LzApp: invalid _dstGasAmount");
        minDstGasLookup[_dstChainId][_type] = _dstGasAmount;
        emit SetMinDstGasLookup(_dstChainId, _type, _dstGasAmount);
    }
```

Code location: contracts/DidSync.sol #L35-37

```
    function setAdapterParams(uint16 version, uint gasForDestinationLzReceive) public
onlyOwner {
        adapterParams = abi.encodePacked(version, gasForDestinationLzReceive);
    }
```

Code location: contracts/DidSync.sol #40-43

```
    function setMaxKYCNumberWithGas(uint256 _maxKYCNumber, uint16 version, uint
gasForDestinationLzReceive) public onlyOwner {
        maxKYCNumber = _maxKYCNumber;
        adapterParams = abi.encodePacked(version, gasForDestinationLzReceive);
    }
```

Solution

It is recommended to use multi-signature to manage Owner permissions to prevent excessive concentration of permissions.

Status

Acknowledged

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002303020001	SlowMist Security Team	2023.02.28 - 2023.03.02	Low Risk

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 1 low risk vulnerability has been confirmed. 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