Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.02.07, the SlowMist security team received the team's security audit application for Celer Network SGN-V2, 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:

- Reentrancy Vulnerability
- Replay Vulnerability
- Reordering Vulnerability
- Short Address Vulnerability
- Denial of Service Vulnerability
- Transaction Ordering Dependence Vulnerability
- Race Conditions Vulnerability
- Authority Control Vulnerability
- Integer Overflow and Underflow Vulnerability
- TimeStamp Dependence Vulnerability
- Uninitialized Storage Pointers Vulnerability
- Arithmetic Accuracy Deviation Vulnerability
- tx.origin Authentication Vulnerability

- "False top-up" Vulnerability
- Variable Coverage Vulnerability
- Gas Optimization Audit
- Malicious Event Log Audit
- Redundant Fallback Function Audit
- Unsafe External Call Audit
- Explicit Visibility of Functions State Variables Audit
- Design Logic Audit
- Scoping and Declarations Audit

3 Project Overview

3.1 Project Introduction

Audit Version:

fcc40a30579c030c2a458c7e3b23cbc42295eedb

Fixed Version:

ce081e85389ee1d989a3a2340cb0ec0bf9ee19e0

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Event log missing	Others	Suggestion	Fixed
N2	Lack of black hole address judgment	Others	Suggestion	Fixed

NO	Title	Category	Level	Status
N3	Event log missing	Others	Suggestion	Fixed
N4	Authority control issues	Authority Control Vulnerability	Low	Ignored

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:

MessageBusAddress			
Function Name	Visibility	Mutability	Modifiers
setMessageBus	Public	Can Modify State	onlyOwner

MessageReceiverApp			
Function Name	Visibility	Mutability	Modifiers
executeMessageWithTransfer	External	Payable	onlyMessageBus
executeMessageWithTransferFallback	External	Payable	onlyMessageBus
executeMessageWithTransferRefund	External	Payable	onlyMessageBus

MessageReceiverApp			
executeMessage	External	Payable	onlyMessageBus

MessageSenderApp			
Function Name	Visibility	Mutability	Modifiers
sendMessage	Internal	Can Modify State	-
sendMessageWithTransfer	Internal	Can Modify State	-
sendTokenTransfer	Internal	Can Modify State	-

MessageBus			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	MessageBusSender MessageBusReceiver
init	External	Can Modify State	-

MessageBusSender			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
sendMessage	External	Payable	-
sendMessageWithTransfer	External	Payable	-
withdrawFee	External	Can Modify State	-
calcFee	Public	-	-
setFeePerByte	External	Can Modify State	onlyOwner

MessageBusSender			
setFeeBase	External	Can Modify State	onlyOwner

MessageBusReceiver			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initReceiver	Internal	Can Modify State	-
executeMessageWithTransfer	External	Payable	-
executeMessageWithTransferRefund	External	Payable	-
executeMessage	External	Payable	-
executeMessageWithTransfer	Private	Can Modify State	-
executeMessageWithTransferFallback	Private	Can Modify State	-
executeMessageWithTransferRefund	Private	Can Modify State	-
verifyTransfer	Private	-	-
computeMessageOnlyId	Private	-	-
executeMessage	Private	Can Modify State	-
setLiquidityBridge	Public	Can Modify State	onlyOwner
setPegBridge	Public	Can Modify State	onlyOwner
setPegVault	Public	Can Modify State	onlyOwner
setPegBridgeV2	Public	Can Modify State	onlyOwner
setPegVaultV2	Public	Can Modify State	onlyOwner

4.3 Vulnerability Summary

[N1] [Suggestion] Event log missing

Category: Others

Content

The owner role can call these function: `setLiquidityBridge` , `setPegBridge`, `setPegVault`, `setPegBridgeV2`, `setPegVaultV2` to set the liquidityBridge, pegBridge, pegVault, pegBridgeV2 and pegVaultV2 address. If there is no event record, it is not conducive to the review of community users.

Code location: message/messagebus/MessageBusReceiver.sol#L364-366

```
    function setLiquidityBridge(address _addr) public onlyOwner {
        liquidityBridge = _addr;
    }
```

Code location: message/messagebus/MessageBusReceiver.sol#L368-370

```
    function setPegBridge(address _addr) public onlyOwner {
        pegBridge = _addr;
    }
```

Code location: message/messagebus/MessageBusReceiver.sol#L372-374

```
    function setPegVault(address _addr) public onlyOwner {
        pegVault = _addr;
    }
```

Code location: message/messagebus/MessageBusReceiver.sol#L376-378

```
    function setPegBridgeV2(address _addr) public onlyOwner {
        pegBridgeV2 = _addr;
    }
```

Code location: message/messagebus/MessageBusReceiver.sol#L380-382

```
    function setPegVaultV2(address _addr) public onlyOwner {
        pegVaultV2 = _addr;
    }
```

Solution

It is recommended to add related event log.

Status

Fixed

[N2] [Suggestion] Lack of black hole address judgment

Category: Others

Content

When setting the bridge address or vault address, there is a lack of judgment on whether to set it as a black hole address. If the owner makes a mistake, it will cause the contract to operate abnormally and the user's token will be lost.

Code location: message/messagebus/MessageBusReceiver.sol#L364-366

```
    function setLiquidityBridge(address _addr) public onlyOwner {
        liquidityBridge = _addr;
    }
```

Code location: message/messagebus/MessageBusReceiver.sol#L368-370

```
    function setPegBridge(address _addr) public onlyOwner {
        pegBridge = _addr;
    }
```

Code location: message/messagebus/MessageBusReceiver.sol#L372-374

```
    function setPegVault(address _addr) public onlyOwner {
        pegVault = _addr;
    }
```

Code location: message/messagebus/MessageBusReceiver.sol#L376-378

```
    function setPegBridgeV2(address _addr) public onlyOwner {
        pegBridgeV2 = _addr;
    }
```

Code location: message/messagebus/MessageBusReceiver.sol#L380-382

```
    function setPegVaultV2(address _addr) public onlyOwner {
        pegVaultV2 = _addr;
    }
```

Solution

When setting the address, increase the judgment of the black hole address.

`require(_addr != address(0));`

Status

Fixed

[N3] [Suggestion] Event log missing

Category: Others

Content

The owner role can call the `setFeePerByte` function and the `setFeeBase` function to set the feePerByte and the feeBase, which are used to calculate the fee of the message .If there is no event record, it is not conducive to the review of community users.

Code location: message/messagebus/MessageBusSender.sol#L109-116

```
    function setFeePerByte(uint256 _fee) external onlyOwner {
        feePerByte = _fee;
    }

    function setFeeBase(uint256 _fee) external onlyOwner {
        feeBase = _fee;
    }
```

Solution

It is recommended to add related event log.

Status

Fixed

[N4] [Low] Authority control issues

Category: Authority Control Vulnerability

Content

The Owner role can use the `setFeePerByte` function and the `setFeeBase` function to set the feePerByte and feeBase, which are used to calculate the fee of the message. But there is no size limit for the feePerByte and feeBase set here. If the setting is too large, users may need to pay a high fee to successfully transmit messages. (`fee = feeBase + _message.length * feePerByte`)

Code location: message/messagebus/MessageBusSender.sol#L109-116

```
    function setFeePerByte(uint256 _fee) external onlyOwner {
        feePerByte = _fee;
    }

    function setFeeBase(uint256 _fee) external onlyOwner {
        feeBase = _fee;
    }
```

Solution

It is recommended to transfer the authority of the Owner role to the governance contract, at least multi-sign should be used. Or limit the size of the feePerByte and feeBase settings.

Status

Ignored; The project team response: The authority of the owner role will be transferred to the governance contract or multi signature address after the system runs

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002202110004	SlowMist Security Team	2022.02.07 - 2022.02.11	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, 3 suggestion vulnerabilities. And 1 low risk vulnerability was ignored; All other 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