Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

On 2023.06.26, the SlowMist security team received the Acala team's security audit application for asset-router, 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
		Function Return Value Security Audit
		External Call Function Security Audit

Serial Number	Audit Class	Audit Subclass
7	Security Design 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

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Missing return value check	Others	Low	Fixed
N2	Suggestions for deprecating functions	Others	Suggestion	Acknowledged

NO	Title	Category	Level	Status
N3	Gas optimization	Gas Optimization Audit	Suggestion	Acknowledged
N4	Missing permission check	Authority Control Vulnerability Audit	Suggestion	Acknowledged

4 Code Overview

4.1 Contracts Description

https://github.com/AcalaNetwork/asset-router

Commit: cca9b733112f9aa4e7af917f1a992162abd42a1e

Review Commit:91a3b5e331b863c06d2337edc80336c8987c21cf

The main network address of the contract is as follows:

Acala:https://blockscout.acala.network/address/0x4D72F3b3e6D2AFeD99923137a04b9Cd9Cd2F21C5/contracts#address-tabs

Karura:https://blockscout.karura.network/address/0x38bCDdc086eC0f68390d7c6452361036469D1203/contracts#address-tabs

4.2 Visibility Description

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

BaseRouter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
routeImpl	Internal	Can Modify State	-
routeNoFee	Public	Can Modify State	-
route	Public	Can Modify State	-

BaseRouter			

Factory			
Function Name	Visibility	Mutability	Modifiers
deployXcmRouter	Public	Can Modify State	-
deployXcmRouterAndRoute	Public	Can Modify State	-
deployXcmRouterAndRouteNoFee	Public	Can Modify State	-
deployWormholeRouter	Public	Can Modify State	-
deployWormholeRouterAndRoute	Public	Can Modify State	-
deployWormholeRouterAndRouteNoFee	Public	Can Modify State	-

FeeRegistry			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getFee	Public	-	-

WormholeRouter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	BaseRouter
routeImpl	Internal	Can Modify State	-

XcmRouter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	BaseRouter
routeImpl	Internal	Can Modify State	-

4.3 Vulnerability Summary

[N1] [Low] Missing return value check

Category: Others

Content

- src/XcmRouter.sol

The return value of `transfer` was not checked.

```
function routeImpl(ERC20 token) internal override {
    IXtokens(XTOKENS).transfer(
        address(token), token.balanceOf(address(this)), _instructions.dest,
_instructions.weight
    );
}
```

- src/WormholeRouter.sol

The return values of `approve` and `transferTokens` are not checked.

```
function routeImpl(ERC20 token) internal override {
    token.approve(_tokenBridgeAddress, token.balanceOf(address(this)));

    ITokenBridge(_tokenBridgeAddress).transferTokens(
        address(token),
        token.balanceOf(address(this)),
        _instructions.recipientChain,
        _instructions.recipient,
        _instructions.arbiterFee,
        _instructions.nonce
    );
}
```

Solution

It is recommended to check the return value.

Status

Fixed

[N2] [Suggestion] Suggestions for deprecating functions

Category: Others

Content

- src/BaseRouter.sol

`selfdestruct` has been deprecated. (https://eips.ethereum.org/EIPS/eip-6049, https://eips.ethereum.org/EIPS/eip-4758)

```
  function routeNoFee(ERC20 token) public {
        routeImpl(token);

        // selfdestruct only if balance is zero to make sure this cannot be used to steal native tokens
        if (address(this).balance == 0) {
            emit RouterDestroyed(address(this));
            selfdestruct(payable(msg.sender));
        }
    }
```

Solution

`selfdestruct` is not recommended.

Status

Acknowledged; acala evm still supports the function of selfdestruct.

[N3] [Suggestion] Gas optimization

Category: Gas Optimization Audit

Content

- src/Factory.sol

The `deployXcmRouter` and `deployWormholeRouter` function uses try,catch to find out if a contract has been created, so that each call consumes more gas if the contract has been created than if it has not been created.

Solution

Using selfdestruct does not return gas, so you can just use a map to record whether the contract corresponding to the fee has been created, which saves gas.

Status

Acknowledged; Conforms to design expectations.

[N4] [Suggestion] Missing permission check

Category: Authority Control Vulnerability Audit

Content

- src/BaseRouter.sol

If the `token` has a fee, then anyone can transfer the token from the contract.

```
    function route(ERC20 token, address relayer) public {
        uint256 fee = fees.getFee(address(token));

        // should use routeNoFee if relayer is not expecting a fee
        require(fee > 0, "zero fee");

        token.safeTransfer(relayer, fee);
        routeNoFee(token);
    }
```

Solution

Can add verification to the caller, or add a whitelist to the receiver.

Status

Acknowledged; Communicate with the project that this is in line with design expectations. Anyone can trigger a route to earn a fee.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002306270001	SlowMist Security Team	2023.06.26 - 2023.06.28	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.

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