

Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

2 Audit Methodology

3 Project Overview

3.1 Project Introduction

3.2 Vulnerability Information

4 Code Overview

4.1 Contracts Description

4.2 Visibility Description

4.3 Vulnerability Summary

5 Audit Result

6 Statement

1 Executive Summary

On 2022.02.15, the SlowMist security team received the team's security audit application for ccTokens, 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

1007db97c1c3df65d5d3edc5340ea3878b5456ba

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Missing address check	Others	Suggestion	Ignored
N2	Missing address check	Others	Suggestion	Ignored
N3	Missing event log	Others	Suggestion	Ignored

NO	Title	Category	Level	Status
N4	Authority control issues	Authority Control Vulnerability	Low	Ignored
N5	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:

Bridge

ETH: 0xCFc4db8515807262bdA0EFC928B28CfD76605EeC

smartBCH: 0x400D6d489adaC736e4AcBD60D76b20564418fD17

4.2 Visibility Description

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

Bridge			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	-
setPair	External	Can Modify State	onlyConfigurationController
setCcToken	External	Can Modify State	onlyConfigurationController
setControllerAddr	External	Can Modify State	onlyConfigurationController
setCctokenConfig	External	Can Modify State	onlyConfigurationController

Bridge			
setRelayer	External	Can Modify State	onlyConfigurationController
setRepository	External	Can Modify State	-
setFeeTo	External	Can Modify State	onlyConfigurationController
setMemberMgr	External	Can Modify State	onlyOwner
setConfigurationController	External	Can Modify State	onlyOwner
getChainId	Internal	-	-
checkFee	Internal	-	-
exchange	Public	Payable	-
confirm	External	Can Modify State	-
withdrawToken	External	Can Modify State	onlyOwner
withdrawETH	External	Payable	onlyOwner
<Receive Ether>	External	Payable	-

Ownable			
Function Name	Visibility	Mutability	Modifiers
ownerInit	Internal	Can Modify State	-
owner	Public	-	-
transferOwnership	Public	Can Modify State	onlyOwner
acceptOwnership	Public	Can Modify State	onlyPendingOwner

Context

Context			
Function Name	Visibility	Mutability	Modifiers
_msgSender	Internal	-	-
_msgData	Internal	-	-

MemberMgr			
Function Name	Visibility	Mutability	Modifiers
getCustodian	External	-	-
getStatusString	Internal	-	-
getMerchantNumber	Public	-	-
getMerchantList	Public	-	-
getMerchantState	Public	-	-
setCustodian	External	Can Modify State	onlyOwner
addMerchant	External	Can Modify State	onlyOwner
stopMerchant	External	Can Modify State	onlyOwner
resumeMerchant	External	Can Modify State	onlyOwner
isMerchant	External	-	-

Ownable			
Function Name	Visibility	Mutability	Modifiers
_owner	Public	-	-
<Constructor>	Public	Can Modify State	-

Ownable			
transferOwnership	Public	Can Modify State	onlyOwner
_transferOwnership	Internal	Can Modify State	-

OwnableIf			
Function Name	Visibility	Mutability	Modifiers
_owner	Public	-	-

4.3 Vulnerability Summary

[N1] [Suggestion] Missing address check

Category: Others

Content

In the `initialize` function, there is a lack of judgment on whether to set the `_owner` parameter as a black hole address. If it is set to `address(0)` when initializing the operation, it will lead to the loss of the owner's permission.

Code location: Bridge-V1.0/Bridge.sol#L131-144

```
    function initialize(address _owner, address _memberMgr, address _relayer, address
_configurationController, address _feeTo, address _WETH) external {
        require(!isInitialized, "Initialized");
        require(_memberMgr != address(0), "memberMgr: address 0");
        require(_relayer != address(0), "relayer: address 0");
        require(_WETH != address(0), "WETH: address 0");
        ownerInit(_owner);
        memberMgr = _memberMgr;
        relayer = _relayer;
        configurationController = _configurationController;
        feeTo = _feeTo;
        WETH = _WETH;
        isInitialized = true;
        emit Initialize(_owner, _memberMgr, _relayer, _configurationController,
```

```
_feeTo, _WETH);
    }
```

Solution

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

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

Status

Ignored; The project team response: Careful inspection will be carried out before and after configuration.

[N2] [Suggestion] Missing address check

Category: Others

Content

When setting the `ccTokenInfo[_cctoken].isCcToken` and `ccTokenInfo[_cctoken].controllerAddr`, there is a lack of judgment on whether to set the `_cctoken` and the `controllerAddr` as a black hole address. If the ConfigurationController role makes a mistake to set the `_cctoken` as `address(0)` and the `_status` as `true` or set the `controllerAddr` as `address(0)`, it will cause the `exchange` function or the `confirm` funtion error.

Code location: Bridge-V1.0/Bridge.sol#L156-159

```
    function setCcToken(address _cctoken, bool _status) external
onlyConfigurationController{
        ccTokenInfo[_cctoken].isCcToken = _status;
        emit SetCcToken(_cctoken, _status);
    }
```

Code location: Bridge-V1.0/Bridge.sol#L161-164

```
    function setControllerAddr(address _cctoken, address _controller) external
onlyConfigurationController{
        ccTokenInfo[_cctoken].controllerAddr = _controller;
        emit SetControllerAddr(_cctoken, _controller);
    }
```

Code location: Bridge-V1.0/Bridge.sol#L166-171

```
    function setCctokenConfig(address _cctoken, bool _status, address _controller)
external onlyConfigurationController {
        ccTokenInfo[_cctoken].isCcToken = _status;
        ccTokenInfo[_cctoken].controllerAddr = _controller;
        emit SetCcToken(_cctoken, _status);
        emit SetControllerAddr(_cctoken, _controller);
    }
```

Solution

When setting the `_cctoken` and the `controllerAddr`, increase the judgment of the black hole address.

Like this: `require(_cctoken != address(0));`

Status

Ignored; The project team response: Careful inspection will be carried out before and after configuration, and if misoperation, the configuration can be modified again to cover the error.

[N3] [Suggestion] Missing event log

Category: Others

Content

The owner role can call the `withdrawToken` and `withdrawETH` functions to withdraw the tokens in the contract to the himself without restriction. If there is no event record, it is not conducive to the review of community users.

Code location: Bridge-V1.0/Bridge.sol#L331-333

```
    function withdrawToken(address _token) external onlyOwner{
        IERC20(_token).safeTransfer(msg.sender,
IERC20(_token).balanceOf(address(this)));
    }
```

Code location: Bridge-V1.0/Bridge.sol#L339-342

```
    function withdrawETH() external payable onlyOwner{
        uint256 value = address(this).balance;
        payable(owner()).transfer(value);
    }
```

Solution

It is recommended to add related event log.

Status

Ignored; The project team response: This issue does not affect the use of the contract and its functions.

[N4] [Low] Authority control issues

Category: Authority Control Vulnerability

Content

The custodian role can call the `confirm` function to pass in any `info` parameter of `ConfirmInfo` type, and transfer a specified amount of tokens from the repository to the address specified in `info` (provided that the address is the merchant on the `_chainIDA` chain). If the amount of the constructed `info` parameter is too large, the tokens in the repository will be used up.

Code location: Bridge-V1.0/Bridge.sol#L289-324

```
function confirm(ConfirmInfo memory info) external returns(bool){
        // Check the information of exchange, including transaction fee, amount,
pairinfo etc.
        require(MemberMgr(memberMgr).getCustodian() == msg.sender, "Only custodian");
        PairInfo memory pair = pairs[info._tokenB][info._chainIDA][info._tokenA];
        require(MemberMgr(memberMgr).isMerchant(info._chainIDA, info._to), "invalid
merchant");
        require(!orderIDStatus[info._orderID], "the orderID already finished");
        require(info._amount >= info._fee, "the amount less than transaction fee");
        uint256 amount0 =  info._amount - info._fee;
        require(!pair.pauseStatus, "the pair is in pause");
        require(pair.bindingStatus, "invalid pair");
        require(info._amount >= pair.minAmount, "Less than the minimum amount");
        orderIDStatus[info._orderID] = true;
```

```
        /**
         * Different processing methods according to the three types of `_tokenB`
(target token).
         * 1. ccToken
         * 2. eth
         * 3. standard tokens issued by third parties
         */
        if (ccTokenInfo[info._tokenB].isCcToken) {
            address controller = ccTokenInfo[info._tokenB].controllerAddr;
            require(Controller(controller).bridgeMint(info._to, amount0), "mint
failed");
            emit Confirm(info._orderID, msg.sender, info._tokenA, info._tokenB,
info._chainIDA, info._amount, info._to, info._fee);
            return true;
        }

        if(info._tokenB == address(0)){
            IERC20(WETH).safeTransferFrom(repository, address(this), amount0);
            IWETH(WETH).withdraw(amount0);
            payable(info._to).transfer(amount0);
            emit Confirm(info._orderID, msg.sender, info._tokenA, info._tokenB,
info._chainIDA, info._amount, info._to, info._fee);
            return true;
        }
        IERC20(info._tokenB).safeTransferFrom(repository, info._to, amount0);
        emit Confirm(info._orderID, msg.sender, info._tokenA, info._tokenB,
info._chainIDA, info._amount, info._to, info._fee);
        return true;
    }
```

Solution

It is recommended to transfer the authority of the custodian role to the governance contract or timelock contract, at least multi-sign should be used.

Status

Ignored; The project team response: It is a business requirement that the custodian has the authority to control the destruction and coinage. In addition, the custodian is the subject of public disclosure, has the responsibility to ensure the safety of funds, and is responsible for the loss of reserves caused by any subjective evil or hacker attack.

At present, the project accepts multiple qualified custodian institutions to jointly manage the reserve, and we are also recruiting partners. After compliance custodian institutions join, it will change the custody address to the multi signature address.

[N5] [Low] Authority control issues

Category: Authority Control Vulnerability

Content

The owner role can call the `withdrawToken` and `withdrawETH` functions to withdraw the tokens in the contract to the himself without restriction.

Code location: Bridge-V1.0/Bridge.sol#L331-333

```
    function withdrawToken(address _token) external onlyOwner{
        IERC20(_token).safeTransfer(msg.sender,
IERC20(_token).balanceOf(address(this)));
    }
```

Code location: Bridge-V1.0/Bridge.sol#L339-342

```
    function withdrawETH() external payable onlyOwner{
        uint256 value = address(this).balance;
        payable(owner()).transfer(value);
    }
```

Solution

It is recommended to transfer the authority of the Owner role to the governance contract or timelock contract, at least multi-sign should be used.

Status

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

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002202180003	SlowMist Security Team	2022.02.15 - 2022.02.18	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 2 low risk, 3 suggestion vulnerabilities. And 2 low risk, 3 suggestion vulnerabilities were ignored. 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