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 2025.03.13, the SlowMist security team received the team's security audit application for MindNetwork FHE Token (FHE), 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

This project implements a multi-chain ERC20 token system with cross-chain interoperability. It features asset bridging via the CCIP, a fixed 1 billion token supply managed through role-based minting, and a secure Merkle Proof-based airdrop mechanism.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Risks of excessive privilege	Authority Control Vulnerability Audit	Medium	Fixed
N2	External call reminder	Design Logic Audit	Information	Fixed
N3	Array Length Mismatch	Design Logic Audit	Suggestion	Fixed

4 Code Overview

4.1 Contracts Description

Audit Version:

https://github.com/mind-network/mind-token-contracts

commit: 408b3342247329c60a44b8511815f80dc2901983

Fixed Version:

https://github.com/mind-network/mind-token-contracts

commit: b936cd5e5e1d94691ba9ad53e6b014b4c0d5bd10

The main network address of the contract is as follows:

https://etherscan.io/address/0xd55C9fB62E176a8Eb6968f32958FeFDD0962727E

https://bscscan.com/address/0xd55C9fB62E176a8Eb6968f32958FeFDD0962727E

https://explorer.mindnetwork.xyz/address/0xd55C9fB62E176a8Eb6968f32958FeFDD0962727E

4.2 Visibility Description

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

arbitrum/FHE			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20 ERC20Permit

arbitrum/FHE

setCCIPAdmin	External	Can Modify State	onlyRole
getCCIPAdmin	External	-	-
mint	Public	Can Modify State	onlyRole
isArbitrumEnabled	External	-	-
registerTokenOnL2	Public	Payable	onlyRole

Airdrop

Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
batchClaim	External	Can Modify State	onlyRole
claim	External	Can Modify State	-
_claim	Private	Can Modify State	-
withdrawERC20	External	Can Modify State	onlyRole

ethereum/FHE

Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20 ERC20Permit
transferCCIPAdmin	External	Can Modify State	-
acceptCCIPAdmin	External	Can Modify State	-
getCCIPAdmin	External	-	-

mindchain/FHE

Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20 ERC20Permit

mindchain/FHE			
setCCIPAdmin	External	Can Modify State	onlyRole
getCCIPAdmin	External	-	-
mint	Public	Can Modify State	onlyRole
bridgeMint	External	Can Modify State	onlyL2Gateway
bridgeBurn	External	Can Modify State	onlyL2Gateway

other-evm/FHE			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20 ERC20Permit
setCCIPAdmin	External	Can Modify State	onlyRole
getCCIPAdmin	External	-	-
mint	Public	Can Modify State	onlyRole

4.3 Vulnerability Summary

[N1] [Medium] Risks of excessive privilege

Category: Authority Control Vulnerability Audit

Content

In the mindchain/FHE contract, the l2Gateway can mint and burn tokens for specific users, and the l2Gateway contract code is not within the audit scope.

Code location:

contracts/mindchain/FHE.sol#L47-60

```
    /**
     * @notice should increase token supply by amount, and should only be callable by
the L2Gateway.
     */
    function bridgeMint(address account, uint256 amount) external override
```

```
onlyL2Gateway {
        require(totalSupply() + amount <= maxSupply, "EXCEEDED_MAX_SUPPLY");
        _mint(account, amount);
    }

    /**
     * @notice should decrease token supply by amount, and should only be callable by
the L2Gateway.
     */
    function bridgeBurn(address account, uint256 amount) external override
onlyL2Gateway {
        _burn(account, amount);
    }
```

Solution

It is recommended that in the early stages of the project, the core role like the owner should use multi-signatures and the time-lock contract to avoid single-point risks. After the project is running stably, the authority of the core role should be handed over to community governance for management.

Status

Fixed; Fully adopted the CCIP solution and removed the mindchain/FHE.sol contract.

[N2] [Information] External call reminder

Category: Design Logic Audit

Content

In the arbitrum/FHE contract, the DEFAULT_ADMIN_ROLE can call the registerTokenOnL2 function to configure the Gateway. However, registerTokenToL2 and setGateway involve external calls, and their code is not within the audit scope.

Code location:

contracts/arbitrum/FHE.sol#L76-108

```
    function registerTokenOnL2(
        address l2CustomTokenAddress,
        uint256 maxSubmissionCostForCustomGateway,
        uint256 maxSubmissionCostForRouter,
        uint256 maxGasForCustomGateway,
        uint256 maxGasForRouter,
        uint256 gasPriceBid,
        uint256 valueForGateway,
        uint256 valueForRouter,
        address creditBackAddress
    ) public payable override onlyRole(DEFAULT_ADMIN_ROLE) {
        // we temporarily set `shouldRegisterGateway` to true for the callback in
registerTokenToL2 to succeed
        bool prev = shouldRegisterGateway;
        shouldRegisterGateway = true;

        IL1CustomGateway(customGatewayAddress).registerTokenToL2{value:
valueForGateway}(  //@audit
            l2CustomTokenAddress,
            maxGasForCustomGateway,
            gasPriceBid,
            maxSubmissionCostForCustomGateway,
            creditBackAddress
        );

        IL2GatewayRouter(routerAddress).setGateway{value: valueForRouter}(
            customGatewayAddress,
            maxGasForRouter,
            gasPriceBid,
            maxSubmissionCostForRouter,
            creditBackAddress
        );

        shouldRegisterGateway = prev;
    }
```

Solution

It is recommended to clarify if external call contracts are credible and check the validity of the incoming resolver address, data, and business logic.

Status

Fixed; Fully adopted the CCIP solution and removed the arbitrum/FHE.sol contract.

[N3] [Suggestion] Array Length Mismatch

Category: Design Logic Audit

Content

In the Airdrop.sol contract, the batchClaim function is used for batch claiming airdrops. However, the function lacks a check to ensure that the users, amounts, and proofs arrays have the same length, which could lead to out-of-bounds errors or undefined behavior.

Code location:

contracts/utils/Airdrop.sol#L33-41

```
function batchClaim(
    address[] calldata users,
    uint256[] calldata amounts,
    bytes32[][] calldata proofs
) external onlyRole(BATCH_ROLE) {
    for (uint256 i; i < users.length; i++) {
        _claim(users[i], amounts[i], proofs[i]);
    }
}
```

Solution

It is recommended to add a length check at the beginning of the function to ensure both arrays are of equal size.

Status

Fixed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002503170001	SlowMist Security Team	2025.03.13 - 2025.03.17	Passed

Summary conclusion: The SlowMist security team uses a manual and the SlowMist team's analysis tool to audit the project. During the audit work, we found 1 medium risk, 1 suggestion, and 1 information. All the findings were fixed. The project team has revoked the EOA addresses DEFAULT_ADMIN_ROLE role and granted the DEFAULT_ADMIN_ROLE role as the mulitisig wallet on Mind and BSC.

Mind:

Revoke tx:

https://explorer.mindnetwork.xyz/tx/0x9db44de7d2daaa0a66db0d9e1c39236a23c41a371d227b4467f3937565f41e5f.

Grant tx:

https://explorer.mindnetwork.xyz/tx/0xac460d4e5d71a3ed8168c49656f00e4244f04597cce673b2fa9be80c910db79d

BSC:

Revoke tx:

https://bscscan.com/tx/0xa514a41650984bc7ccb11a096dbf83929b19fa259e96c2adddf761193e5d7708

Grant tx:

https://bscscan.com/tx/0xe759d69dae0bb4e980cba3a6070fe11e231365d245817534e256509446c99ab8

The mulitisig contract (0x468fF40d484df5B3a0567FC8556F284E7E3Ce6Cd) in the BSC is controlled by 3 EOA addresses. And the mulitisig contract in the Mind is 0x5d2BCA21fF4Ca6e65682B039B9687383dd96a967, which is a unopen-sroced contract.

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.

SLOWMIST

Official Website

www.slowmist.com

✉

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist