SLOWMIST

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 2024.04.01, the SlowMist security team received the Bitlayer team's security audit application for getBTC, 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

The project includes a contract for users to exchange tokens for native tokens (gas tokens).

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	Medium	Acknowledged
N2	Missing zero address check	Others	Suggestion	Fixed

NO	Title	Category	Level	Status
N3	Variable names are the same	Others	Information	Fixed

4 Code Overview

4.1 Contracts Description

https://github.com/bitlayer-org/getBTC

Initial audit commit: 345036b3d5ce868347e5c46ab8a6fe2a071d78df

Final audit commit:8248293a6cfd3dc899d937cdbabec4c789026f2e

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:

TokenExchange			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	EIP712
permitAndSwap	External	Can Modify State	onlyOperator
setVaults	External	Can Modify State	onlyOwner
withdrawERC20	External	Can Modify State	onlyOwner
withdrawBTC	External	Can Modify State	onlyOwner
transferOwnership	External	Can Modify State	onlyOwner
setOperator	External	Can Modify State	onlyOwner
splitSignature	Internal	-	-

TokenExchange			
verifySignture	Internal	Can Modify State	-
nonces	Public	-	-
DOMAIN_SEPARATOR	External	-	-
name	Public	-	-
<Receive Ether>	External	Payable	-

MockERC20			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20 ERC20Permit

4.3 Vulnerability Summary

[N1] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

1.In the TokenExchange contract, the `owner` role can set `vaults` mapping through the setVaults function.

- TokenExchange.sol#L74-L77

```
function setVaults(address valut, bool status) external onlyOwner {
    vaults[valut] = status;
    emit SetVaults(valut, status);
}
```

2.In the TokenExchange contract, the `owner` role can set the `Operator` role address through the `setOperator` function; the owner's ownership can be transferred through the `transferOwnership` function.

- TokenExchange.sol#L90-L94,L95-L99

```
function transferOwnership(address newOwner) external onlyOwner {
    require(newOwner != address(0),"Owner_Should_Not_Zero_Address");
```

```
        owner = newOwner;
        emit TransferOwnership(newOwner);
    }
    function setOperator(address newOp) external onlyOwner {
        operator = newOp;
        emit SetOperator(newOp);
    }
```

3.In the TokenExchange contract, the `owner` role can withdraw the ERC20 token in the contract through the `withdrawERC20` function; the Native token in the contract can be withdrawn through the `withdrawBTC` function.

- TokenExchange.sol#L78-L82,L82-L88

```
    function withdrawERC20(address tokenAddress, address receiver, uint256 amount)
external onlyOwner {
        require(amount <=
IERC20(tokenAddress).balanceOf(address(this)),"Token_Not_Enough");
        SafeERC20.safeTransfer(IERC20(tokenAddress), receiver, amount);
        emit Withdrawn(tokenAddress, receiver, amount);
    }
    function withdrawBTC(address payable receiver, uint256 amount) external onlyOwner
{
        require(amount <= address(this).balance,"BTC_Not_Enough");
        (bool success, bytes memory returnData) = receiver.call{value: amount}("");
        require(success, string(returnData));
        emit Withdrawn(address(0), receiver, amount);
    }
```

Solution

In the short term, transferring owner ownership to multisig contracts is an effective solution to avoid single-point risk. But in the long run, it is a more reasonable solution to implement a privilege separation strategy and set up multiple privileged roles to manage each privileged function separately. The authority involving user funds should be managed by the community, and the authority involving emergency contract suspension can be managed by the EOA address. This ensures both a quick response to threats and the safety of user funds.

Status

Acknowledged

[N2] [Suggestion] Missing zero address check

Category: Others

Content

In the TokenExchange contract, the `withdrawBTC` function and `setOperator` function lack zero address check.

- TokenExchange.sol#L83-L88,L95-L98

```
    function withdrawBTC(address payable receiver, uint256 amount) external onlyOwner
{
        require(amount <= address(this).balance,"BTC_Not_Enough");
        (bool success, bytes memory returnData) = receiver.call{value: amount}("");
        require(success, string(returnData));
        emit Withdrawn(address(0), receiver, amount);
    }

    function setOperator(address newOp) external onlyOwner {
        operator = newOp;
        emit SetOperator(newOp);
    }
```

Solution

It is recommended to add zero address check.

Status

Fixed

[N3] [Information] Variable names are the same

Category: Others

Content

The private string variable `_name` defined in the TokenExchange contract has the same name as the private immutable string `_name` inherited from the EIP712 contract.

- TokenExchange.sol#L19

```
string private _name;
```

- EIP712.sol#L49

```
ShortString private immutable _name;
```

Solution

It is recommended to use a descriptive, unique name for each variable to improve code readability and maintainability.

Status

Fixed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002404030001	SlowMist Security Team	2024.04.01 - 2024.04.03	Medium 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 medium risk, 1 suggestion, 1 information.

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