SLOWMIST

Smart Contract

Security Audit Report

Table Of Contents

1 Executive Summary

On 2024.01.24, the SlowMist security team received the Bitmap team's security audit application for BTCLayer2BridgeContract, 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 includes ERC20/ERC721 token contract and BTC Layer2 Bridge contract.

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	Confirming
N2	Missing the event records	Others	Suggestion	Confirming

NO	Title	Category	Level	Status
N3	Missing zero address check	Others	Suggestion	Confirming
N4	BTC address not verified	Others	Suggestion	Confirming
N5	Parameter _symbol is not case checked	Design Logic Audit	Low	Confirming

4 Code Overview

4.1 Contracts Description

https://github.com/MerlinLayer2/BTCLayer2BridgeContract

commit: 045ec451ffb316d5504c0c6fabdb9c23373431ab

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:

BTCLayer2Bridge			
Function Name	Visibility	Mutability	Modifiers
<Receive Ether>	External	Payable	-
initialize	External	Can Modify State	onlyValidAddress onlyValidAddress onlyValidAddress onlyValidAddress onlyValidAddress initializer
setSuperAdminAddress	Public	Can Modify State	onlyValidAddress
setNormalAdminAddress	Public	Can Modify State	onlyValidAddress
addUnlockTokenAdmi	Public	Can Modify	onlyValidAddress

BTCLayer2Bridge			
nAddress		State	
addERC20TokenWrapped	Public	Can Modify State	-
mintERC20Token	Public	Can Modify State	-
burnERC20Token	Public	Payable	-
addERC721TokenWrapped	Public	Can Modify State	-
setBaseURI	Public	Can Modify State	-
mintERC721Token	Public	Can Modify State	-
burnERC721Token	Public	Payable	-
unlockNativeToken	Public	Can Modify State	-
lockNativeToken	Public	Payable	-
allERC20TokenAddressLength	Public	-	-
allERC20TxHashLength	Public	-	-
allERC721TokenAddressLength	Public	-	-
allERC721TxHashLength	Public	-	-
allNativeTokenTxHashLength	Public	-	-
userERC20MintTxHashLength	Public	-	-
userERC721MintTxHashLength	Public	-	-
userNativeTokenMintTxHashLength	Public	-	-
setBridgeSettingsFee	External	Can Modify State	-

BTCLayer2BridgeERC721			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	onlyValidAddress onlyValidAddress initializer
addERC721TokenWrapped	External	Can Modify State	onlyBridge
setBaseURI	External	Can Modify State	onlyBridge
mintERC721Token	External	Can Modify State	onlyBridge
burnERC721Token	External	Can Modify State	onlyBridge
allERC721TokenAddressLength	Public	-	-
allERC721TxHashLength	Public	-	-
userERC721MintTxHashLength	Public	-	-

ERC721TokenWrapped			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC721
mint	External	Can Modify State	onlyBridge
burn	External	Can Modify State	onlyBridge
_baseURI	Internal	-	-
setBaseURI	External	Can Modify State	onlyBridge

BTCLayer2BridgeERC20			
Function Name	Visibility	Mutability	Modifiers
initialize	External	Can Modify State	onlyValidAddress onlyValidAddress initializer

BTCLayer2BridgeERC20			
addERC20TokenWrapped	External	Can Modify State	onlyBridge
mintERC20Token	External	Can Modify State	onlyBridge
burnERC20Token	External	Can Modify State	onlyBridge
allERC20TokenAddressLength	Public	-	-
allERC20TxHashLength	Public	-	-
userERC20MintTxHashLength	Public	-	-

ERC20TokenWrapped			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20 ERC20Permit
mint	External	Can Modify State	onlyBridge
burn	External	Can Modify State	onlyBridge
decimals	Public	-	-

4.3 Vulnerability Summary

[N1] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

In the BTCLayer2Bridge contract, SuperAdmin role can transfer SuperAdmin role permissions and set NormalAdmin role. The SuperAdmin and NormalAdmin roles can add the UnlockTokenAdmin role.

- BTCLayer2Bridge.sol#L113-L118,L120-L123,L125-L130

```
function setSuperAdminAddress
function setNormalAdminAddress
function addUnlockTokenAdminAddress
```

In the BTCLayer2Bridge contract, the superAdminAddress and normalAdminAddress roles can create ERC20 and ERC721 token Wrapped contracts, and modify the `baseURI` of the ERC721TokenWrapped contract.

- BTCLayer2Bridge.sol#L132-L137,L155-L160,L163-L165

```
function addERC20TokenWrapped
function addERC721TokenWrapped
function setBaseURI
```

In the BTCLayer2Bridge contract, the unlockTokenAdmin role can arbitrary mint ERC20 and ERC721 tokens, and unlock NativeToken to transfer Native Token to the specified address.

- BTCLayer2Bridge.sol#L139-L143

```
function mintERC20Token
function mintERC721Token
function unlockNativeToken
```

In the BTCLayer2Bridge contract, the superAdminAddress role can modify the `feeAddress` and `bridgeFee` parameters.

- BTCLayer2Bridge.sol#L239-L248

```
function setBridgeSettingsFee
```

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

Confirming

[N2] [Suggestion] Missing the event records

Category: Others

Content

In BTCLayer2Bridge, superAdmin and normalAdmin can modify sensitive parameters, but no events are recorded.

- BTCLayer2Bridge.sol#L120-L123,L125-L130,L239-L248

```
    function setNormalAdminAddress(address _account) public
onlyValidAddress(_account) {
        require(msg.sender == superAdminAddress, "Illegal permissions");
        normalAdminAddress = _account;
    }
    function addUnlockTokenAdminAddress(address _account) public
onlyValidAddress(_account) {
        require(msg.sender == superAdminAddress || msg.sender == normalAdminAddress,
"Illegal permissions");
        require(unlockTokenAdminAddressSupported[_account] == false, "Current address
has been added");
        unlockTokenAdminAddressList.push(_account);
        unlockTokenAdminAddressSupported[_account] = true;
        }
    function setBridgeSettingsFee(address _feeAddress, uint256 _bridgeFee) external {
        require(msg.sender == superAdminAddress, "Illegal permissions");

        if (_feeAddress != address(0)) {
            feeAddress = _feeAddress;
        }
        if (_bridgeFee > 0) {
            bridgeFee = _bridgeFee;
        }
    }
```

Solution

It is recommended to record events when sensitive parameters are modified for self-inspection or community review.

Status

Confirming

[N3] [Suggestion] Missing zero address check

Category: Others

Content

In the BTCLayer2Bridge contract, the `unlockNativeToken` function does not perform a zero check on the `to` address.

- BTCLayer2Bridge.sol#L183-L194

```
    function unlockNativeToken(bytes32 txHash, address to, uint256 amount) public {
        require(unlockTokenAdminAddressSupported[msg.sender], "Illegal permissions");
        require(nativeTokenTxHashUnlocked[txHash] == false, "Transaction has been
executed");
        nativeTokenTxHashUnlocked[txHash] = true;
        allNativeTokenTxHash.push(txHash);
        userNativeTokenMintTxHash[to].push(txHash);
        (bool success, ) = to.call{value: amount}(new bytes(0));
        if (!success) {
            revert EtherTransferFailed();
        }
        emit UnlockNativeToken(txHash, to, amount);
    }
```

Solution

It is recommended to add the zero address check.

Status

Confirming

[N4] [Suggestion] BTC address not verified

Category: Others

Content

In the BTCLayer2Bridge contract, the `burnERC20Token` function, `burnERC721Token` function, and `lockNativeToken` function don't verify the `destBtcAddr` parameter.

- BTCLayer2Bridge.sol#L145-L153,L173-L181

```
    function burnERC20Token(address token, uint256 amount, string memory destBtcAddr)
public payable {
        require(msg.value == bridgeFee, "The bridgeFee is incorrect");
        IBTCLayer2BridgeERC20(bridgeERC20Address).burnERC20Token(msg.sender, token,
amount);
        (bool success, ) = feeAddress.call{value: bridgeFee}(new bytes(0));
        if (!success) {
            revert EtherTransferFailed();
        }
        emit BurnERC20Token(token, msg.sender, amount, destBtcAddr);
    }

    function burnERC721Token(address token, uint256 tokenId, string memory
destBtcAddr) public payable {
        require(msg.value == bridgeFee, "The bridgeFee is incorrect");
        IBTCLayer2BridgeERC721(bridgeERC721Address).burnERC721Token(msg.sender, token,
tokenId);
        (bool success, ) = feeAddress.call{value: bridgeFee}(new bytes(0));
        if (!success) {
            revert EtherTransferFailed();
        }
        emit BurnERC721Token(token, msg.sender, tokenId, destBtcAddr);
    }

     function lockNativeToken(string memory destBtcAddr) public payable {
        require(msg.value > bridgeFee, "Insufficient cross-chain assets");

        (bool success, ) = feeAddress.call{value: bridgeFee}(new bytes(0));
        if (!success) {
            revert EtherTransferFailed();
        }

        emit LockNativeToken(msg.sender, msg.value - bridgeFee, destBtcAddr);
    }
```

Solution

It is recommended to verify the destBtcAddr parameter to verify whether it is a valid BTC address.

Status

Confirming

[N5] [Low] Parameter _symbol is not case checked

Category: Design Logic Audit

Content

The `_symbol` field of ERC20 tokens and ERC721 tokens on the Ethereum chain is case-sensitive, but for BRC20 Tick is not case-sensitive. In the BTCLayer2Bridge contract, the `addERC20TokenWrapped` function and the `addERC721TokenWrapped` function do not standardize the case format of the `_symbol` parameter passed in.

- BTCLayer2Bridge.sol#L132-L137,L155-L160

```
    function addERC20TokenWrapped(string memory _name, string memory _symbol, uint8
_decimals) public returns(address) {
        require(msg.sender == superAdminAddress || msg.sender == normalAdminAddress,
"Illegal permissions");
        address tokenWrappedAddress =
IBTCLayer2BridgeERC20(bridgeERC20Address).addERC20TokenWrapped(_name, _symbol,
_decimals);
        emit AddERC20TokenWrapped(tokenWrappedAddress, _name, _symbol, _decimals);
        return tokenWrappedAddress;
    }
    function addERC721TokenWrapped(string memory _name, string memory _symbol, string
memory _baseURI) public returns(address) {
        require(msg.sender == superAdminAddress || msg.sender == normalAdminAddress,
"Illegal permissions");
        address tokenWrappedAddress =
IBTCLayer2BridgeERC721(bridgeERC721Address).addERC721TokenWrapped(_name, _symbol,
_baseURI);
        emit AddERC721TokenWrapped(tokenWrappedAddress, _name, _symbol, _baseURI);
        return tokenWrappedAddress;
    }
```

Solution

It is recommended to check the case format of the _symbol parameter and unify it into uppercase or lowercase format.

Status

Confirming

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002401260001	SlowMist Security Team	2024.01.24 - 2024.01.26	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 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.

SLOWMIST

Official Website
www.slowmist.com

✉

E-mail
team@slowmist.com

Twitter
@SlowMist_Team

Github
https://github.com/slowmist