SLOWMIST

Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2024.03.07, the SlowMist security team received the team's security audit application for TaprootChain - BTCLayer2Bridge, 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 is an Ethereum-based smart contract named BTCLayer2Bridge. The contract provides a bridging service for transferring assets between Bitcoin and Ethereum blockchains. This audit only covers the decentralized aspects of the BTCLayer2Bridge contract and excludes a risk assessment of the centralized components and external dependencies. Users can use this contract to wrap, mint, burn, and transfer ERC20 and ERC721 tokens across chains. The contract includes management addresses, fee settings, and a list of unlock token admin addresses, allowing for various types of transactions and operations. It also covers handling of fees and permission management.

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	High	Fixed
N2	Receive can lock users' native tokens	Others	Low	Acknowledged
N3	Parameter _symbol is not case checked	Design Logic Audit	Low	Acknowledged
N4	Delete the address without popping up the list	Design Logic Audit	Suggestion	Acknowledged
N5	SuperAdmin Transfer Recommendations	Others	Suggestion	Acknowledged
N6	Low-level call reminder	Reentrancy Vulnerability	Suggestion	Acknowledged
N7	Preemptive Initialization	Race Conditions Vulnerability	Suggestion	Acknowledged
N8	Dev address setting enhancement suggestions	Others	Suggestion	Acknowledged
N9	External call reminder	Unsafe External Call Audit	Suggestion	Acknowledged

4 Code Overview

4.1 Contracts Description

Audit Version:

https://github.com/TaprootChain/contract

commit: d18367af4b0aff67968393e7d5db53698afee116

- contracts/*.sol

File name and Hash:

IBTCLayer2BridgeERC20.sol

Hash(SHA256): 5cd3c9298df6956063a7d842b21ced9d55e258d3728d48ad9070cf13391bf14c

IBTCLayer2BridgeERC721.sol

Hash(SHA256): f1b6c7cbd91723e515cad6f4f316c0f0960b5cc0bf82131084218d256ed1e819

The main network address of the contract is as follows:

Proxy: https://scan.taprootchain.io/address/0xb57F659958BEa213b85d6Bdf848F61D5C1964D1c

Implementation:

https://scan.taprootchain.io/address/0x674a75C1EcfA9607Fb1A2B9AfDE86A8BAb97D18C

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
addUnlockTokenAdminAddress	Public	Can Modify State	onlyValidAddress
delUnlockTokenAdminAddress	Public	Can Modify State	onlyValidAddress
addERC20TokenWrapped	Public	Can Modify State	-
mintERC20Token	Public	Can Modify State	-

BTCLayer2Bridge			
burnERC20Token	Public	Payable	-
addERC721TokenWrapped	Public	Can Modify State	-
setBaseURI	Public	Can Modify State	-
tokenURI	Public	Can Modify State	-
batchMintERC721Token	Public	Can Modify State	-
batchBurnERC721Token	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	-

4.3 Vulnerability Summary

[N1] [High] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

1.In the BTCLayer2Bridge contract, the superAdmin role is initialized in the initialize function and can be modified in the setSuperAdminAddress function. The superAdmin can also set the normalAdmin role through the setNormalAdminAddress function.

Code location:

https://github.com/TaprootChain/contract/blob/d18367af4b0aff67968393e7d5db53698afee116/BTCLayer2Bridge.sol#L140C1-L151C6

```
    function setSuperAdminAddress(address _account) public onlyValidAddress(_account)
{
        require(msg.sender == superAdminAddress, "Illegal permissions");
        address oldSuperAdminAddress = superAdminAddress;
        superAdminAddress = _account;
        emit SuperAdminAddressChanged(oldSuperAdminAddress, _account);
    }

    function setNormalAdminAddress(address _account) public
onlyValidAddress(_account) {
        require(msg.sender == superAdminAddress, "Illegal permissions");
        normalAdminAddress = _account;
        emit SetNormalAdminAddress(_account);
    }
```

2.The superAdmin and normalAdmin roles can add or remove addresses from the `unlockTokenAdminAddressList` through the addUnlockTokenAdminAddress and delUnlockTokenAdminAddress functions. And only users whose boolean value of `unlockTokenAdminAddressSupported` in `unlockTokenAdminAddressList` is true can call the mintERC20Token, batchMintERC721Token, and unlockNativeToken functions

Code location:

https://github.com/TaprootChain/contract/blob/d18367af4b0aff67968393e7d5db53698afee116/BTCLayer2Bridge.sol#L153C1-L166C6

```
    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;
        emit AddUnlockTokenAdminAddress(_account);
    }

    function delUnlockTokenAdminAddress(address _account) public onlyValidAddress(_account) {
        require(msg.sender == superAdminAddress || msg.sender == normalAdminAddress, "Illegal permissions");
        require(unlockTokenAdminAddressSupported[_account] == true, "Current address is not exist");
        unlockTokenAdminAddressSupported[_account] = false;
        emit DelUnlockTokenAdminAddress(_account);
    }
```

3.The superAdmin and normalAdmin roles can add `tokenWrappedAddress` and set the ERC721 token BaseURI through the addERC20TokenWrapped, addERC721TokenWrapped, and setBaseURI function.

Code location:

https://github.com/TaprootChain/contract/blob/d18367af4b0aff67968393e7d5db53698afee116/BTCLayer2Bridge.sol#L168C1-L173C6

https://github.com/TaprootChain/contract/blob/d18367af4b0aff67968393e7d5db53698afee116/BTCLayer2Bridge.sol#L191C1-L202C6

```
    function addERC20TokenWrapped(string memory _name, string memory _symbol, uint8 _decimals, uint256 _cap) public returns(address) {
        require(msg.sender == superAdminAddress || msg.sender == normalAdminAddress, "Illegal permissions");
        address tokenWrappedAddress = IBTCLayer2BridgeERC20(bridgeERC20Address).addERC20TokenWrapped(_name, _symbol, _decimals, _cap);
        emit AddERC20TokenWrapped(tokenWrappedAddress, _name, _symbol, _decimals, _cap);
        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;
    }

    function setBaseURI(address token, string calldata newBaseTokenURI) public {
        require(msg.sender == superAdminAddress || msg.sender == normalAdminAddress,
"Illegal permissions");
        IBTCLayer2BridgeERC721(bridgeERC721Address).setBaseURI(token,
newBaseTokenURI);
        emit SetBaseURI(token, newBaseTokenURI);
    }
```

4.The superAdmin role can modify the `feeAddress` and `bridgeFee` parameters and the `bridgeFee` has no upper limit.

Code location:

https://github.com/TaprootChain/contract/blob/d18367af4b0aff67968393e7d5db53698afee116/BTCLayer2Bridge.sol#L287C1-L298C6

```
    function setBridgeSettingsFee(address _feeAddress, uint256 _bridgeFee) external {
        require(msg.sender == superAdminAddress, "Illegal  permissions");

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

        emit SetBridgeSettingsFee(_feeAddress, _bridgeFee);

}
```

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

Fixed; After communicating with the project team, they transferred the ownership and superAdmin role to an EOA address(0x0e5af757d64c0f2bcf392ca3f00313680284a0e7) in the following transactions:

https://scan.taprootchain.io/tx/0x8bff150243bc7165e1b27c361e9f7feb37c58023aa9d2a6d3745017c6f3e5539

https://scan.taprootchain.io/tx/0xf5fdd4670ca1df92e5755b7dd3386f50a318b48d29318d26b9e5f8602ac5c72f

And the EOA address is used the **Safeheron MPC Wallet** to control.

[N2] [Low] Receive can lock users’ native tokens

Category: Others

Content

There is a receive function in the BTCLayer2Bridge contract so that the contracts can receive native tokens. However, the receive function can lock users’ native tokens when users transfer the native token in these contracts by mistake. And the payable modifier can help these functions which need to call with the native tokens.

Code location:

https://github.com/TaprootChain/contract/blob/d18367af4b0aff67968393e7d5db53698afee116/BTCLayer2Bridge.sol#L119

```
    receive() external payable {}
```

Solution

It’s recommended to remove the receive() function if the contract only needs to receive ether from functions in the contract, and use the payable modifier in these functions instead. Or provide refund logic for users when need to use the receive() function.

Status

Acknowledged

[N3] [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.

Code location:

https://github.com/TaprootChain/contract/blob/d18367af4b0aff67968393e7d5db53698afee116/BTCLayer2Bridge.sol#L168C1-L173C6

https://github.com/TaprootChain/contract/blob/d18367af4b0aff67968393e7d5db53698afee116/BTCLayer2Bridge.sol#L191C1-L196C6

```
    function addERC20TokenWrapped(string memory _name, string memory _symbol, uint8
_decimals, uint256 _cap) public returns(address) {
        require(msg.sender == superAdminAddress || msg.sender == normalAdminAddress,
"Illegal permissions");
        address tokenWrappedAddress =
IBTCLayer2BridgeERC20(bridgeERC20Address).addERC20TokenWrapped(_name, _symbol,
_decimals, _cap);
        emit AddERC20TokenWrapped(tokenWrappedAddress, _name, _symbol, _decimals,
_cap);
        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 the uppercase or lowercase format.

Status

Acknowledged

[N4] [Suggestion] Delete the address without popping up the list

Category: Design Logic Audit

Content

In the addUnlockTokenAdminAddress function, the superAdmin and normalAdmin roles can add user address into the `unlockTokenAdminAddressList` and set the `unlockTokenAdminAddressSupported` to true. But in the delUnlockTokenAdminAddress function, the superAdmin and normalAdmin roles remove the unlockTokenAdminAddress just by setting the `unlockTokenAdminAddressSupported` to false without popping up from the `unlockTokenAdminAddressList`. Once called the delUnlockTokenAdminAddress function deletes the address, the superAdmin and normalAdmin roles can call the addUnlockTokenAdminAddress function to add the same address added before into the `unlockTokenAdminAddressList` and the length of the `unlockTokenAdminAddressList` will increase.

Code location:

https://github.com/TaprootChain/contract/blob/d18367af4b0aff67968393e7d5db53698afee116/BTCLayer2Bridge.sol#L153C1-L166C6

```
    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;
        emit AddUnlockTokenAdminAddress(_account);
    }

    function delUnlockTokenAdminAddress(address _account) public
onlyValidAddress(_account) {
        require(msg.sender == superAdminAddress || msg.sender == normalAdminAddress,
"Illegal permissions");
        require(unlockTokenAdminAddressSupported[_account] == true, "Current address
is not exist");
        unlockTokenAdminAddressSupported[_account] = false;
        emit DelUnlockTokenAdminAddress(_account);
    }
```

Solution

It’s recommended to pop up the address from the `unlockTokenAdminAddressList` when deleting unlockTokenAdminAddress.

Status

Acknowledged

[N5] [Suggestion] SuperAdmin Transfer Recommendations

Category: Others

Content

In the BTCLayer2Bridge contract, superAdmin directly overwrites the previous address with the new address during transfer. If superAdmin calls the setSuperAdminAddress function with the wrong address when the operation is wrong, this will result in the loss of the superAdmin role permissions.

Code location:

https://github.com/TaprootChain/contract/blob/d18367af4b0aff67968393e7d5db53698afee116/BTCLayer2Bridge.sol#L140C1-L145C6

```
    function setSuperAdminAddress(address _account) public onlyValidAddress(_account)
{
        require(msg.sender == superAdminAddress, "Illegal permissions");
        address oldSuperAdminAddress = superAdminAddress;
        superAdminAddress = _account;
        emit SuperAdminAddressChanged(oldSuperAdminAddress, _account);
    }
```

Solution

It is recommended to add an operation similar to pendingOwner for the second confirmation, so that even if the operation is wrong or the incoming address is wrong, the operation can be cancelled in the second confirmation operation.

Status

Acknowledged

[N6] [Suggestion] Low-level call reminder

Category: Reentrancy Vulnerability

Content

In the BTCLayer2Bridge contract, the burnERC20Token, batchBurnERC721Token, lockNativeToken, and unlockNativeToken use low-level calls to transfer native tokens to the feeAddress and to address from the unlockNativeToken function. But do not limit the amount of gas used to transfer native tokens to the user.

Code location:

https://github.com/TaprootChain/contract/blob/d18367af4b0aff67968393e7d5db53698afee116/BTCLayer2Bridge.sol#L184 ,L223 ,L237, L247

```
    (bool success, ) = feeAddress.call{value: bridgeFee}(new bytes(0));
    (bool success, ) = to.call{value: amount}(new bytes(0));
```

Solution

When using low-level calls, it is recommended to limit the amount of gas used.

Status

Acknowledged

[N7] [Suggestion] Preemptive Initialization

Category: Race Conditions Vulnerability

Content

By calling the initialize and deploy functions to initialize the contracts, there is a potential issue that malicious attackers preemptively call the initialize function to initialize.

Code location:

https://github.com/TaprootChain/contract/blob/d18367af4b0aff67968393e7d5db53698afee116/BTCLayer2Bridge.sol#L121C1-L138C6

```
    function initialize(
        address _initialOwner,
        address _superAdminAddress,
        address _bridgeERC20Address,
        address _bridgeERC721Address,
        address _feeAddress
    ) external onlyValidAddress(_initialOwner)
    onlyValidAddress(_superAdminAddress)
    onlyValidAddress(_bridgeERC20Address)
    onlyValidAddress(_bridgeERC721Address)
    onlyValidAddress(_feeAddress) virtual initializer {
        superAdminAddress = _superAdminAddress;
        bridgeERC20Address = _bridgeERC20Address;
        bridgeERC721Address = _bridgeERC721Address;
        feeAddress = _feeAddress;
        // Initialize OZ contracts
        __Ownable_init_unchained(_initialOwner);
    }
```

Solution

It is suggested that the initialization operation can be called in the same transaction immediately after the contract is created to avoid being maliciously called by the attacker.

Status

Acknowledged

[N8] [Suggestion] Dev address setting enhancement suggestions

Category: Others

Content

In the contract, the superAdmin role can set the feeAddress to receive the fee. If the addfeeAddress is an EOA address, in a scenario where the private keys are leaked, the team’s revenue will be stolen.

Code location:

https://github.com/TaprootChain/contract/blob/d18367af4b0aff67968393e7d5db53698afee116/BTCLayer2Bridge.sol#L287C1-L298C6

```
    function setBridgeSettingsFee(address _feeAddress, uint256 _bridgeFee) external {
        require(msg.sender == superAdminAddress, "Illegal  permissions");

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

Solution

It is recommended to set the insurance address as a multi-signature contract to avoid the leakage of private keys and the theft of team income.

Status

Acknowledged

[N9] [Suggestion] External call reminder

Category: Unsafe External Call Audit

Content

In the contract, the core functions `mintERC20Token`, `burnERC20Token`, `batchMintERC721Token`, `batchBurnERC721Token`, `unlockNativeToken`, and `lockNativeToken`, which are mainly used for fund interaction by `unlockTokenAdminAddressSupported` users, are all completed by external calls to bridgeERC20Address and bridgeERC721Address. The current contract also does complete verification of the incoming parameters `txHash`, `_symbol`, `_baseURI`, `destBtcAddr`, `inscriptionNumbers`, `inscriptionIds`, etc., and these verifications may be completed by a centralized system or these external call contracts. This audit does not include centralized systems or external call contracts. Users need to pay attention to these external risks when calling these functions.

Solution

It is recommended to clarify whether this external call contract is credible and check the validity of the incoming resolver address and data.

Status

Acknowledged

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002403110001	SlowMist Security Team	2024.03.07 - 2024.03.11	Low Risk

Summary conclusion: The SlowMist security team uses a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 high risk, 2 low risks, and 6 suggestions. After the over-privileged role was transferred to an EOA address that used the safeheron MPC wallet to control, the conclusion of this audit reduces to the low risk, the potential risks of centralized and external dependencies still need to stay vigilant.

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