Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2024.12.23, the SlowMist security team received the StakeStone team's security audit application for SBTC Bera Vault, 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 audit of the SBTC Bera Vault contract for the StakeStone protocol.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Asset removal triggers excess withdrawals	Design Logic Audit	Critical	Fixed

NO	Title	Category	Level	Status
N2	Missing event records	Others	Suggestion	Fixed
N3	Missing zero address check	Others	Suggestion	Fixed
N4	Redundant code	Others	Suggestion	Fixed
N5	Risk of excessive authority	Authority Control Vulnerability Audit	Medium	Acknowledged

4 Code Overview

4.1 Contracts Description

https://github.com/stakestone/stone_bera_vault/blob/master/src/SBTCBeraVault.sol

Initial audit version: 6b1a8f6f6f3024adaebb4957884068a1d4bb3e1f

Final audit version: 0d94c3acd951c231c44a170d4745048e9102bc1c

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:

SBTCBeraVault			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
deposit	Public	Can Modify State	-
mint	External	Can Modify State	-
requestRedeem	External	Can Modify State	-

SBTCBeraVault			
cancelRequest	External	Can Modify State	-
claimRedeemRequest	Public	Can Modify State	-
pendingRedeemRequest	Public	-	-
claimableRedeemRequest	External	-	-
previewDeposit	Public	-	-
previewMint	Public	-	-
getRate	Public	-	-
getUnderlyings	External	-	-
getWithdrawTokens	External	-	-
rollToNextRound	External	Can Modify State	onlyRole
withdrawAssets	External	Can Modify State	onlyRole
repayAssets	External	Can Modify State	onlyRole
setCap	External	Can Modify State	onlyRole
addUnderlyingAsset	External	Can Modify State	onlyRole
removeUnderlyingAsset	External	Can Modify State	onlyRole
addWithdrawToken	External	Can Modify State	onlyRole
removeWithdrawToken	External	Can Modify State	onlyRole
setDepositPause	External	Can Modify State	onlyRole
setFeeRate	External	Can Modify State	onlyRole
setFeeRecipient	External	Can Modify State	onlyRole

4.3 Vulnerability Summary

[N1] [Critical] Asset removal triggers excess withdrawals

Category: Design Logic Audit

Content

In the SBTCBeraVault contract, when the `removeUnderlyingAsset` function deletes the `tokenDecimals` mapping of the `_asset` token, if the token is also `withdrawTokens`, the default value of 0 will be used as `decimals` for subsequent calculations, making the redemption amount calculation results in the `claimRedeemRequest`, `claimableRedeemRequest`, and `rollToNextRound` functions significantly greater than the actual amount due, causing the user to obtain excess tokens.

- src/SBTCBeraVault.sol#L420-L440

```
    function removeUnderlyingAsset(
        address _asset
    ) external onlyRole(VAULT_OPERATOR_ROLE) {
        if (!isUnderlyingAsset[_asset]) revert InvalidAsset();

        address[] memory assets = underlyingAssets;

        uint256 length = assets.length;
        uint256 i;
        for (i; i < length; i++) {
            if (assets[i] == _asset) {
                underlyingAssets[i] = underlyingAssets[length - 1];
                underlyingAssets.pop();
                break;
            }
        }
        isUnderlyingAsset[_asset] = false;
        delete tokenDecimals[_asset];

        emit RemoveUnderlyingAsset(_asset);
    }
```

Solution

It is recommended not to delete the value of tokenDecimals[_asset] to facilitate the calculation of the redemption amount.

Status

Fixed

[N2] [Suggestion] Missing event records

Category: Others

Content

In the SBTCBeraVault contract, the `SetCap` event is not triggered after the `constructor` function sets the `cap` parameter.

- src/SBTCBeraVault.sol#L85-L90

```
    constructor(address _lpToken, uint256 _cap) {
        //...
        cap = _cap;
    }
```

Solution

It is recommended to add event logging.

Status

Fixed

[N3] [Suggestion] Missing zero address check

Category: Others

Content

In the SBTCBeraVault contract, the `constructor` function lacks a zero address check for the `_lpToken` address.

- src/SBTCBeraVault.sol#L85-L90

```
    constructor(address _lpToken, uint256 _cap) {
        //...
        lpToken = Token(_lpToken);
        //...
    }
```

Solution

It is recommended to add a zero address check.

Status

Fixed

[N4] [Suggestion] Redundant code

Category: Others

Content

1.In the SBTCBeraVault contract, the `deposit` function and the `mint` function check whether `lpToken.totalSupply() + shares` is greater than `cap`, which has already been checked once in the `previewDeposit` function and the previewMint function.

- src/SBTCBeraVault.sol#L92-L124, L126-L158

```
    function deposit(
        address _asset,
        uint256 _amount,
        address _receiver
    ) public returns (uint256 shares) {
        if ((shares = previewDeposit(_asset, _amount)) == 0)
            revert ZeroShares();

        if (lpToken.totalSupply() + shares > cap) revert DepositCapped();
        //...
    }

    function mint(
        address _asset,
        uint256 _shares,
        address _receiver
    ) external returns (uint256 assets) {
        //...
        if (lpToken.totalSupply() + _shares > cap) revert DepositCapped();

        assets = previewMint(_asset, _shares);
        //...
    }
```

2.In the SBTCBeraVault contract, the `requestShares` and `shares` parameters of the `claimRedeemRequest` function are repeated, and their values are both `redeemRequest.requestShares`.

- src/SBTCBeraVault.sol#L227-L255

```
    function claimRedeemRequest() public {
        //...
        uint256 requestShares = redeemRequest.requestShares;
```

```
        uint256 round = redeemRequest.requestRound;
        uint256 shares = redeemRequest.requestShares;
        //...
    }
```

Solution

It is recommended to delete the redundant code.

Status

Fixed

[N5] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

1.In the SBTCBeraVault contract, the `VAULT_OPERATOR_ROLE` role can modify important parameters in the contract and roll to a new round through the following functions.

- src/SBTCBeraVault.sol#L333-L362, L399-L402, L404-L418, L420-L440, L442-L452, L452-L474, L476-L482, L484-L493, L494-L500

```
    function rollToNextRound
    function setCap
    function addUnderlyingAsset
    function removeUnderlyingAsset
    function addWithdrawToken
    function removeWithdrawToken
    function setDepositPause
    function setFeeRate
    function setFeeRecipient
```

2.In the SBTCBeraVault contract, the `ASSETS_MANAGEMENT_ROLE` role can withdraw or repay the `underlyingAssets` tokens in the contract through the `withdrawAssets` function and the `repayAssets` function.

- src/SBTCBeraVault.sol#L364-L381, L383-L397

```
    function withdrawAssets
    function repayAssets
```

Solution

In the short term, during the early stages of the project, the protocol may need to frequently set various parameters to ensure the stable operation of the protocol. Therefore, transferring the ownership of core roles to a multisig management can effectively solve the single-point risk, but it cannot mitigate the excessive privilege risk. In the long run, after the protocol stabilizes, transferring the owner ownership to community governance and executing through a timelock can effectively mitigate the excessive privilege risk and increase the community users' trust in the protocol.

Status

Acknowledged; According to the project team, privileged roles in the contract will be managed through a multi-signature wallet system to enhance security governance.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002412240002	SlowMist Security Team	2024.12.23 - 2024.12.24	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 critical risk, 1 medium risk, 3 suggestion.

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