# Smart Contract Security Audit Report

# Table Of Contents

# 1 Executive Summary

On 2024.12.11, the SlowMist security team received the StakeStone team's security audit application for Stone 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 the Bera Vault part of the StakeStone protocol, which mainly includes Vault and Oracle modules.

## 3.2 Vulnerability Information

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

| NO | Title | Category | Level | Status |
| --- | --- | --- | --- | --- |
| N1 | Cap can be set below total supply | Design Logic Audit | Low | Acknowledged |
| N2 | Asset removal lacks balance verification | Design Logic Audit | Medium | Acknowledged |

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

# 4 Code Overview

## 4.1 Contracts Description

https://github.com/stakestone/stone_bera_vault

Initial audit version: 1245b17b8136666e8f9f5a071cc5c219cf965cfd

Final audit version: ee6309b118c5c95690fc618f02ce9b77a3f26c94

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:

| StoneBeraVault | | | |
| --- | --- | --- | --- |
| 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 | - |
| cancelRequest | External | Can Modify State | - |

| StoneBeraVault | | | |
|---|---|---|---|
| claimRedeemRequest | Public | Can Modify State | - |
| pendingRedeemRequest | Public | - | - |
| claimableRedeemRequest | Public | - | - |
| totalAssets | Public | - | - |
| activeAssets | Public | - | - |
| activeShares | Public | - | - |
| convertToShares | Public | - | - |
| convertToAssets | Public | - | - |
| previewDeposit | Public | - | - |
| previewMint | Public | - | - |
| getRate | Public | - | - |
| getUnderlyings | 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 |
| setDepositPause | External | Can Modify State | onlyRole |

| Token | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | ERC20 |

| Token | | | |
|---|---|---|---|
| mint | External | Can Modify State | onlyRole |
| burn | External | Can Modify State | onlyRole |

| DepositWrapper | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| depositETH | External | Payable | - |
| <Receive Ether> | External | Payable | - |

| OracleConfigurator | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| updateOracle | External | Can Modify State | onlyRole |
| getPrice | External | - | - |

| Oracle | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | - |
| getPrice | External | - | - |

| StoneOracle | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | Oracle |
| getPrice | External | - | - |

| StoneOracle | | | |
|---|---|---|---|
| updatePrice | Public | Can Modify State | - |

| WETHOracle | | | |
|---|---|---|---|
| Function Name | Visibility | Mutability | Modifiers |
| <Constructor> | Public | Can Modify State | Oracle |
| getPrice | External | - | - |

## 4.3 Vulnerability Summary

**[N1] [Low] Cap can be set below total supply**

**Category: Design Logic Audit**

**Content**

In the StoneBeraVault contract, the `setCap` function did not check whether the `_cap` parameter was greater than or equal to the total supply of LP token, which could cause the `cap` to be less than the total supply of LP token.

- src/StoneBeraVault.sol#L426-L429

```
function setCap(uint256 _cap) external onlyRole(VAULT_OPERATOR_ROLE) {
    emit SetCap(cap, _cap);
    cap = _cap;
}
```

**Solution**

It is recommended to check whether the _cap parameter is greater than or equal to lpToken.totalSupply().

**Status**

Acknowledged; The project team indicates this aligns with the design intent: if the cap is set below total supply, deposits will be suspended until user withdrawals bring total supply below the cap level.

**[N2] [Medium] Asset removal lacks balance verification**

**Category: Design Logic Audit**

**Content**

In the `removeUnderlyingAsset` function of the StoneBeraVault contract, when the `VAULT_OPERATOR_ROLE` role removes the underlying asset, the function directly removes it from the `underlyingAssets` array without checking whether the contract still holds a balance of the asset. This may cause the actual value of the asset to be ignored when calculating `totalAssets`, thereby affecting the share calculation and user rights.

- src/StoneBeraVault.sol#L445-L464

```
    function removeUnderlyingAsset(
        address _asset
    ) external onlyRole(VAULT_OPERATOR_ROLE) {
        if (!isUnderlyingAssets[_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;
            }
        }
        isUnderlyingAssets[_asset] = false;

        emit RemoveUnderlyingAsset(_asset);
    }
```

**Solution**

It is recommended to add a balance check before removing the underlying asset to ensure that the asset has been fully liquidated before allowing the removal.

**Status**

Acknowledged; The project team states that removeUnderlyingAsset is not a routine operation - it will only be executed after all assets have been fully withdrawn, ensuring safe removal of the underlying asset.

### [N3] [Medium] Risk of excessive authority

**Category: Authority Control Vulnerability Audit**

**Content**

1.In the Token contract, the `MINTER_ROLE` and `BURNER_ROLE` roles can mint and burn tokens through the `mint` and `burn` functions. This is actually used in LP token.

- src/Token.sol#L18-L20, L22-L27

```
function mint
function burn
```

2.In the OracleConfigurator contract, the `ORACLE_MANAGER_ROLE` role can modify the oracle address corresponding to the token through the `updateOracle` function.

- src/oracle/OracleConfigurator.sol#L21-L31

```
function updateOracle
```

3.In the StoneBeraVault contract, the `VAULT_OPERATOR_ROLE` role can modify important parameters in the contract through the following functions.

- src/StoneBeraVault.sol#L344-L375, L426-L429, L431-L443, L445-L464, L466-L472

```
function rollToNextRound
function setCap
function addUnderlyingAsset
function removeUnderlyingAsset
function setDepositPause
```

4.In the StoneBeraVault contract, the `ASSETS_MANAGEMENT_ROLE` role can borrow and repay assets in the contract through the `withdrawAssets` function and the `repayAssets` function.

- src/StoneBeraVault.sol#L377-L399, L401-L424

```
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; The project team confirms that privileged addresses will be transferred to a GnosisSafe multisig wallet after deployment.

### [N4] [Suggestion] Missing event records

**Category: Others**

**Content**

In the StoneOracle contract, the `updatePrice` function can update the price of the token but lacks event logging.

- src/oracle/StoneOralce.sol#L37-L42

```
    function updatePrice() public {
        uint256 price = stoneVault.currentSharePrice();
        if (price == 0) revert InvalidPrice();

        prices.push(price);
    }
```

**Solution**

It is recommended to add event logging.

**Status**

Acknowledged

### [N5] [Suggestion] Missing zero address check

**Category: Others**

**Content**

1.In the StoneOracle contract, the `constructor` function lacks a zero address check for the `_stoneVault` parameter.

- src/oracle/StoneOralce.sol#L18-L31

```
    constructor(
        address _token,
        string memory _name,
        address _stoneVault
    ) Oracle(_token, _name) {
        //...
    }
```

2.In the DepositWrapper contract, the `constructor` function lacks zero address checks for the `_weth` and `_vault` parameters.

- src/wrapper/DepositWrapper.sol#L27-L31

```
    constructor(address _weth, address _vault) {
        //...
    }
```

3.In the StoneBeraVault contract, the `constructor` function lacks zero address checks for the `_lpToken`, `_withdrawToken`, and `_oracleConfigurator` parameters.

- src/StoneBeraVault.sol#L74-L90

```
    constructor(
        address _lpToken,
        address _withdrawToken,
        address _oracleConfigurator,
        uint256 _cap
    ) {
        //...
    }
```

**Solution**

It is recommended to add a zero address check.

**Status**

Fixed

# 5 Audit Result

| Audit Number | Audit Team | Audit Date | Audit Result |
| --- | --- | --- | --- |
| 0X002412120001 | SlowMist Security Team | 2024.12.11 - 2024.12.12 | 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 2 medium risk, 1 low risk, 2 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