SLOWMIST

Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2024.10.08, the SlowMist security team received the Chainbase Network team's security audit application for Chainbase AVS, 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 audit primarily focuses on the ChainbaseServiceManager contract, which inherits from BLSSignatureChecker, ServiceManagerBase, and ChainbaseServiceManagerStorage, and implements functionalities related to chain service management, particularly concerning the creation, response, and verification processes for tasks. The ChainbaseServiceManagerStorage contract mainly defines the variables required for the business logic of the contract.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Missing the event records	Others	Suggestion	Acknowledged
N2	Overflow vulnerability caused by type conversion	Integer Overflow and Underflow Vulnerability	High	Fixed

4 Code Overview

4.1 Contracts Description

Audit Version:

https://github.com/chainbase-labs/chainbase-avs/tree/feat-node/contracts

commit: 525894f26f6bc12ca95d876e19da4bef4ed70eed

Fixed Version:

https://github.com/chainbase-labs/chainbase-avs/

commit: 940ae00ac89a892621cc1eb3a2782abceeb679fa

Audit scope:

- src/ChainbaseServiceManager.sol
- src/ChainbaseServiceManagerStorage.sol
- src/IChainbaseServiceManager.sol

The main network address of the contract is as follows:

https://etherscan.io/address/0xb73a87E8F7f9129816d40940Ca19DFa396944C71

4.2 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

ChainbaseServiceManager			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	BLSSignatureChecker ServiceManagerBase
initialize	Public	Can Modify State	initializer
setAggregator	External	Can Modify State	onlyOwner
setGenerator	External	Can Modify State	onlyOwner
createNewTask	External	Can Modify State	onlyGenerator
respondToTask	External	Can Modify State	onlyAggregator
taskNumber	External	-	-

AVS			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
operators	External	-	-
strategyParams	External	-	-
isInAllowlist	External	-	-
canRegister	External	-	-
registerOperator	External	Can Modify State	-
UpdateMetadataURI	External	Can Modify State	onlyOwner
getRestakeableStrategies	External	-	-
getOperatorRestakedStrategies	External	-	-
_getRestakeableStrategies	Internal	-	-
_setStrategyParams	Private	Can Modify State	-

4.3 Vulnerability Summary

[N1] [Suggestion] Missing the event records

Category: Others

Content

In the following contracts, the owner role can modify some sensitive parameters, but there are no event logs in these functions.

Code location:

contracts/src/ChainbaseServiceManager.sol#L69-79

```
    function setAggregator(address _aggregator) external onlyOwner {
        aggregator = _aggregator;
    }

    function setGenerator(address _generator) external onlyOwner {
        generator = _generator;
    }
```

Solution

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

Status

Acknowledged

[N2] [High] Overflow vulnerability caused by type conversion

Category: Integer Overflow and Underflow Vulnerability

Content

In the respondToTask function of the ChainbaseServiceManager contract, a for loop is used to check if the value of each returned signedStakeForQuorum multiplied by _THRESHOLD_DENOMINATOR (100) is greater than the value of each returned totalStakeForQuorum multiplied by the average threshold. This is done to verify whether the signer has at least the threshold percentage of stake for each quorum.

However, when quorumThresholdPercentage is set in the createNewTask function, it is a value of type uint32. During the check, this value is directly cast to type uint8. If the value of quorumThresholdPercentage exceeds the maximum value of type uint8 (255), an underflow issue will occur. This could ultimately lead to quorumStakeTotals.totalStakeForQuorum[i] * uint8(quorumThresholdPercentage) being less than the expected value, causing the check to always pass.

Code location:

contracts/src/ChainbaseServiceManager.sol#L150

```
    function respondToTask(
        Task calldata task,
        TaskResponse calldata taskResponse,
        NonSignerStakesAndSignature memory nonSignerStakesAndSignature
    ) external override onlyAggregator {
        ...

        uint32 quorumThresholdPercentage = task.quorumThresholdPercentage;

        ...

        // check that signatories own at least a threshold percentage of each quorum
        for (uint256 i = 0; i < quorumNumbers.length; i++) {
            // we don't check that the quorumThresholdPercentages are not >100 because
a greater value would trivially fail the check, implying
            // signed stake > total stake
            require(
                quorumStakeTotals.signedStakeForQuorum[i] * _THRESHOLD_DENOMINATOR
                    >= quorumStakeTotals.totalStakeForQuorum[i] *
uint8(quorumThresholdPercentage),
                "ChainbaseServiceManager: signatories do not own at least threshold
percentage of a quorum"
            );
        }

        ...
    }
```

Solution

It is recommended to add a check in the createNewTask function to ensure that the value of quorumThresholdPercentage is less than the maximum value of type uint8 (255).

Status

Fixed; Fixed in commit 11e1f65e0632c7c00793e6005dd8803b414fa784.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002410100004	SlowMist Security Team	2024.10.08 - 2024.10.10	Passed

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 and 1 suggestion. All findings were fixed or acknowledged. The code has been deployed to the mainnet.

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