Blockchain Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.08.01, the SlowMist security team received the Bifrost team's security audit application for Bifrost Finance, 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" 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 party 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.

Level	Description
Suggestion	There are better practices for coding or architecture.

In black box testing and gray box testing, we use methods such as fuzz testing and script testing to test the robustness of the interface or the stability of the components by feeding random data or constructing data with a specific structure, and to mine some boundaries Abnormal performance of the system under conditions such as bugs or abnormal performance. In white box testing, we use methods such as code review, combined with the relevant experience accumulated by the security team on known blockchain security vulnerabilities, to analyze the object definition and logic implementation of the code to ensure that the code has the key components of the key logic. Realize no known vulnerabilities; at the same time, enter the vulnerability mining mode for new scenarios and new technologies, and find possible 0day errors.

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:

NO.	Audit Items	Result
1	Replay Vulnerability	Passed
2	Reordering Vulnerability	Passed

NO.	Audit Items	Result
3	Race Conditions Vulnerability	Passed
4	Authority Control Vulnerability	Passed
5	Block data Dependence Vulnerability	Passed
6	Explicit Visibility of Functions Audit	Passed
7	Arithmetic Accuracy Deviation Vulnerability	Passed
8	Malicious Event Log Audit	Passed
9	Others	Some Risks
10	SAST	Some Risks
11	State Consistency Audit	Passed
12	Failure Rollback Audit	Passed
13	Unit Test Audit	Some Risks
14	Integer Overflow Audit	Passed
15	Parameter Verification Audit	Passed
16	Error Unhandle Audit	Passed
17	Boundary Check Audit	Some Risks
18	Weights Audit	Passed
19	Macros Audit	Passed
20	Non-standard token security audit	Passed
21	Prevent misuse audit	Passed

3 Project Overview

3.1 Project Introduction

A parachain focused on building bridges of chains based on PoS consensus.

3.2 Coverage

Target Code and Revision:

https://github.com/bifrost-finance/bifrost/

branch: vglmr, commit: 5988d0babf9b743c95d56d215a7bd2c7df12e6ad

Review version:

https://github.com/bifrost-finance/bifrost/tree/fix-slowmist-vulnerabilities

Code scope：

pallets/slp/src/primitives/moonbeam_primitives.rs

pallets/slp/src/agents/moonbeam_agent/agent.rs

pallets/slp/src/agents/moonbeam_agent/mod.rs

pallets/slp/src/agents/moonbeam_agent/types.rs

3.3 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Lack of unit testing	Unit Test Audit	Low	Fixed
N2	Redundant codes	Others	Suggestion	Ignored
N3	Crate vulnerabilities	SAST	Suggestion	Fixed

NO	Title	Category	Level	Status
N4	No check list length	Boundary Check Audit	Medium	Fixed

4 Findings

4.1 Visibility Description

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

slp			
Function Name	Parameters verification coverage	XCM	unit test
initialize_delegator	1/1	0	1
bond	4/4	1	4
bond_extra	4/4	1	3
unbond	4/4	1	5
unbond_all	2/2	1	3
rebond	4/4	1	3
delegate	0/3	0	3
undelegate	3/3	1	3
redelegate	3/3	1	4
payout	0/4	0	1
liquidize	4/4	1	6

slp			
chill	2/2	0	1
transfer_back	4/4	1	2
transfer_to	4/4	0	2
tune_vtoken_exchange_rate	1/4	0	0
add_delegator	2/3	0	1
remove_delegator	2/2	0	1
add_validator	2/2	0	0
remove_validator	2/2	0	0
charge_hosting_fee	1/4	0	0
supplement_fee_reserve	0/4	0	9
check_delegator_ledger_query_response	3/4	0	0
check_validators_by_delegator_query_response	0/3	0	0
fail_delegator_ledger_query_response	0/1	0	3
fail_validators_by_delegator_query_response	0/1	0	1

4.2 Vulnerability Summary

[N1] [Low] Lack of unit testing

Category: Unit Test Audit

Content

pallets/slp/src/agents/moonbeam_agent/agent.rs

Some functions lack unit tests, see the table above for details.

Solution

Add unit testing for all key functions

Status

Fixed

[N2] [Suggestion] Redundant codes

Category: Others

Content

pallets/slp/src/agents/moonbeam_agent/agent.rs

```
fn delegate( //SlowMist// Redundant
    &self,
    _who: &MultiLocation,
    _targets: &Vec<MultiLocation>,
    _currency_id: CurrencyId,
) -> Result<QueryId, Error<T>> {
    Err(Error::<T>::Unsupported)
}

fn redelegate(
    &self,
    who: &MultiLocation,
    _targets: &Option<Vec<MultiLocation>>, //SlowMist// Redundant
    currency_id: CurrencyId,
) -> Result<QueryId, Error<T>>

fn payout( //SlowMist// Redundant
    &self,
    _who: &MultiLocation,
    _validator: &MultiLocation,
    _when: &Option<TimeUnit>,
    _currency_id: CurrencyId,
) -> Result<(), Error<T>> {
```

```
    Err(Error::<T>::Unsupported)
}

fn liquidize(
    &self,
    who: &MultiLocation,
    _when: &Option<TimeUnit>, //SlowMist// Redundant
    validator: &Option<MultiLocation>,
    currency_id: CurrencyId,
) -> Result<QueryId, Error<T>>

fn tune_vtoken_exchange_rate(
    &self,
    _who: &Option<MultiLocation>, //SlowMist// Redundant
    token_amount: BalanceOf<T>,
    _vtoken_amount: BalanceOf<T>, //SlowMist// Redundant
    currency_id: CurrencyId,
) -> Result<(), Error<T>>

fn tune_vtoken_exchange_rate(
    &self,
    _who: &Option<MultiLocation>, //SlowMist// Redundant
    token_amount: BalanceOf<T>,
    _vtoken_amount: BalanceOf<T>, //SlowMist// Redundant
    currency_id: CurrencyId,
) -> Result<(), Error<T>>

fn charge_hosting_fee(
    &self,
    amount: BalanceOf<T>,
    _from: &MultiLocation, //SlowMist// Redundant
    to: &MultiLocation,
    currency_id: CurrencyId,
) -> DispatchResult

    fn charge_hosting_fee(
    &self,
    amount: BalanceOf<T>,
    _from: &MultiLocation, //SlowMist// Redundant
    to: &MultiLocation,
    currency_id: CurrencyId,
) -> DispatchResult

fn check_validators_by_delegator_query_response( //SlowMist// Redundant
```

```
    &self,
    _query_id: QueryId,
    _entry: ValidatorsByDelegatorUpdateEntry<MultiLocation, MultiLocation, Hash<T>>,
    _manual_mode: bool,
) -> Result<bool, Error<T>> {
    Err(Error::<T>::Unsupported)
}

fn fail_validators_by_delegator_query_response( //SlowMist// Redundant
    &self,
    _query_id: QueryId,
) -> Result<(), Error<T>> {
    Err(Error::<T>::Unsupported)
}
```

Solution

Remove the redundant functions or parameters

Status

Ignored; Actually it is an unified trait with functions designed to be compatible with multiple staking mode of different chains.

[N3] [Suggestion] Crate vulnerabilities

Category: SAST

Content

```
Crate:     lru
Version:   0.6.6
Title:     Use after free in lru crate
Date:      2021-12-21
ID:        RUSTSEC-2021-0130
URL:       https://rustsec.org/advisories/RUSTSEC-2021-0130
Solution:  Upgrade to >=0.7.1

Crate:     owning_ref
Version:   0.4.1
Title:     Multiple soundness issues in `owning_ref`
Date:      2022-01-26
```

```
ID:        RUSTSEC-2022-0040
URL:       https://rustsec.org/advisories/RUSTSEC-2022-0040
Solution:  No fixed upgrade is available!

Crate:     time
Version:   0.1.44
Title:     Potential segfault in the time crate
Date:      2020-11-18
ID:        RUSTSEC-2020-0071
URL:       https://rustsec.org/advisories/RUSTSEC-2020-0071
Solution:  Upgrade to >=0.2.23
```

Solution

Upgrade the crates.

Status

Fixed

[N4] [Medium] No check list length

Category: Boundary Check Audit

Content

pallets/slp/src/agents/moonbeam_agent/agent.rs

```
fn add_delegator(
    &self,
    index: u16,
    who: &MultiLocation,
    currency_id: CurrencyId,
) -> DispatchResult

fn add_validator(&self, who: &MultiLocation, currency_id: CurrencyId) ->
DispatchResult
```

The maximum length of `delegators/validators` is not checked when adding item.

Solution

Check the length of the `delegators/validators`

Status

Fixed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002208050004	SlowMist Security Team	2022.08.01 - 2022.08.05	Passed

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, 2 suggestion vulnerabilities. And 1 suggestion vulnerabilities were ignored; All other findings were fixed.

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