AlphaFi Smart Contarct

Audit Report

contact@bitslab.xyz

https://twitter.com/movebit_

Mon Dec 09 2024

AlphaFi Smart Contarct Audit Report

1 Executive Summary

1.1 Project Information

Description	A liquid staking project on sui.
Type	DeFi
Auditors	MoveBit
Timeline	Thu Nov 28 2024 - Mon Dec 09 2024
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/AlphaFiTech/liquid-staking/
Commits	3682e5b4a0f83c1427076a6c9ac92ce4922419c6 1226b080b97cfe60b5c8f73b9b1f73716d27b80c

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
MOV	contracts/Move.toml	1aa9bbf8ace928bd4da60ce77148fa5c780c0917
LST1	contracts/sources/liquid_staking.move	3167837e455531e62cfc2bf5815490fbd5866d0f
EVE	contracts/sources/events.move	06a8d4f8db6422981a143450d868df5b301d9ae0
FEE	contracts/sources/fees.move	192a70ef00877f778b0ea5cb43b7d30287fb972f
STO	contracts/sources/storage.move	a451b9416f538e175bcd1f1ff817fe0f41417b8e
VER	contracts/sources/version.move	702c56f0b731264e944a3fc8dd2ebbc307c5f9e8
CEL	contracts/sources/cell.move	8d221cbcdfb36f28f87e4786db27b7acde86cd07

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	5	5	0
Informational	1	1	0
Minor	2	2	0
Medium	2	2	0
Major	0	0	0
Critical	0	0	0

1.4 MoveBit Audit Breakdown

MoveBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow by bit operations
- Number of rounding errors
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting
- Unchecked CALL Return Values
- The flow of capability
- Witness Type

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Formal Verification(Optional)

Perform formal verification for key functions with the Move Prover.

(4) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by AlphaFi to identify any potential issues and vulnerabilities in the source code of the AlphaFi smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 5 issues of varying severity, listed below.

ID	Title	Severity	Status
FEE-1	Unreasonable Fee Setting	Minor	Fixed
LST-1	In the `flash_stake_start()` Function, the `sui_mint_amount` should be Rounded Up	Medium	Fixed
LST-2	Lack of Slippage Protection in `Mint` Function	Medium	Fixed
LST-3	Lack of Events Emit	Minor	Fixed
STO-1	Unused `public(package)` Visibility Function	Informational	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the AlphaFi Smart Contract :

Admin

- Admin can set the validator addresses and weights of the staking pool through `set_validator_addresses_and_weights<P>` .
- Admin can update the fee rate through `update_fees<P>()` .
- Admin can generate a new collection cap through `generate_new_collection_cap<P>()` .
- Admin can pause and unpause the staking pool through `pause<P>()` and `un_pause<P>()` .
- Admin can collect fees of the staking pool through `collect_fees<P>()` .

User

- User can create a new staking pool through `create_lst<P: drop>()` .
- User can flash loan `LST` coins and repay `SUI` coins through `flash_stake_start<P: drop>()` and `flash_stake_conclude<P: drop>()` .
- User can stake `SUI` coins in the pool and get `LST` coins through `mint<P: drop>()` .
- User can burn `LST` coins and get `SUI` coins through `redeem<P: drop>()` .

4 Findings

FEE-1 Unreasonable Fee Setting

Severity: Minor

Status: Fixed

Code Location:

contracts/sources/fees.move#163

Descriptions:

In the `fees` module, the system's fee validation is insufficient, allowing most fees to be set as high as 100%. If an administrator maliciously sets exorbitant fees, it could lead to significant user asset losses or disrupt the normal operation of the protocol.

Suggestion:

It is recommended to restrict fees to a reasonable range, such as 0% to 10% or any other range suitable for the protocol's use case.

LST-1 In the `flash_stake_start()` Function, the `sui_mint_amount` should be Rounded Up

Severity: Medium

Status: Fixed

Code Location:

contracts/sources/liquid_staking.move#275

Descriptions:

In the `flash_stake_start()` function, the protocol first calculates the `sui_mint_amount` based on the `lst_amount`.

```
self.refresh_no_entry<P>(system_state, ctx);
    // deduct fees
    let sui_mint_amount = self.lst_amount_to_sui_amount(amount);
```

Then, in the `flash_stake_conclude()` function, it deposits the SUI.

```
assert!(sui.balance().value() >= (sui_amount + fee));
    let mut sui_balance = sui.into_balance();
    // deduct fees
    self.fees.join(sui_balance.split(fee));

    self.flash_stake_supply_reduce<P>(lst_amount);
    let stake_balance = sui_balance.split(sui_amount);

    let stake_balance_value = stake_balance.value();
```

This pattern has a standard equivalent in the EVM ecosystem, ERC4626.

https://eips.ethereum.org/EIPS/eip-4626 In ERC4626, there is a description like this:

> If (1) it's calculating the amount
> of shares a user has to supply to
> receive a given amount of the underlying
> tokens or (2) it's calculating the
> amount of underlying tokens a user

has to provide to receive a certain

amount of shares, it should round up.

In the protocol, `sui_mint_amount` is calculated as `sui_mint_amount = total_sui_supply * lst_amount / total_lst_supply`, which currently rounds down.

```
fun lst_amount_to_sui_amount<P>(
    self: &LiquidStakingInfo<P>,
    lst_amount: u64
): u64 {
    let total_sui_supply = self.total_sui_supply();
    let total_lst_supply = self.total_lst_supply();

    assert!(total_lst_supply > 0, EZeroLstSupply);

    let sui_amount = (total_sui_supply as u128)
        * (lst_amount as u128)
        / (total_lst_supply as u128);

    sui_amount as u64
}
```

The suggestion is to modify the calculation to round up when calculating the `sui_mint_amount`.

Suggestion:

It is recommended to modify the calculation to round up when calculating the `sui_mint_amount`

Resolution:

This issue has been fixed. The client has adopted our suggestions.

LST-2 Lack of Slippage Protection in `Mint` Function

Severity: Medium

Status: Fixed

Code Location:

contracts/sources/liquid_staking.move#292

Descriptions:

The `mint` function in the protocol lacks slippage protection. Since the mint fee rate can be set to 100%, an administrator could adjust the mint fee rate without user awareness, resulting in substantial fees and potential user asset losses.

Suggestion:

It is recommended to add slippage protection or change the mint fee rate limit.

LST-3 Lack of Events Emit

Severity: Minor

Status: Fixed

Code Location:

contracts/sources/liquid_staking.move#415 426

Descriptions:

The contract lacks appropriate events for some key functions. The lack of event records for these functions may cause inconvenience in the subsequent tracking and contract status changes.

Suggestion:

It is recommended to emit events for the functions.

STO-1 Unused `public(package)` Visibility Function

Severity: Informational

Status: Fixed

Code Location:

contracts/sources/storage.move#518

Descriptions:

The `join_fungible_stake` function, declared with `public(package)` visibility, can only be invoked within the module. However, this function has not been called anywhere throughout the entire project.

Suggestion:

It is recommended to remove the function or change its visibility.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.