AlphaFi Smart Contract

Audit Report

contact@bitslab.xyz

https://twitter.com/movebit_

Thu Jun 27 2024

AlphaFi Smart Contract Audit Report

1 Executive Summary

1.1 Project Information

Description	A Yield Aggregator on SUI.
Type	DeFi
Auditors	MoveBit
Timeline	Mon Jun 17 2024 - Thu Jun 27 2024
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/AlphaFiTech/contracts
Commits	e4cca71655209e99c0d484e0ed60a92a4f7d42de bbe315c4dcac28c6840612a21f5d083311be67db

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MOV	alphafi/Move.toml	877640e7af82d058351ddecfe47cebf5cfd802c5
ASP	alphafi/sources/cetus_pool/alpha_sui_pool.move	7c21aa32a9492c94e281d7d38b94ce97fb84897b
CPO	alphafi/sources/cetus_pool/cetus_pool.move	9aaf6fe42367d5c5af4d81a0f372d7bdc4706f03
ASI	alphafi/sources/cetus_pool/alpha_sui_investor.move	01d70ceca56bd86d4e61b4c22a1d18fb3ce625d7
CIN	alphafi/sources/cetus_pool/cetus_investor.move	eb003b95afd0dc97cfc83612163c940779c9d402
DIS	alphafi/sources/distributor.move	c66d85517f9f57210dcb0b97c9a9ba64dc19c2b7
ALP	alphafi/sources/alphapool.move	a547f7293d2756675254387684396c329a7cd661
CVE	alphafi/sources/current_version.move	7a6c8b4d779447e471d58ddd8333ad4c4dae5cd7
ERR	alphafi/sources/error.move	cc746d71b714508402bd52b1cfbad045d152e678
VER	alphafi/sources/version.move	c5433cec82d4b961b2e32fdb596c326a60ef2128
ALL	alphafi/sources/allocator.move	0bf198f207ad427831c99781cc0189187d0a9066

CON	alphafi/sources/converter.move	3938c56f8d81a2a0b8aee72de28a4bc092b818b9
MOV6	alpha/Move.toml	4bcac731580c6ff779ded3801ad067f6a27392e1
ALP1	alpha/sources/alpha.move	eff0fd09c945d949b1bd943d358359351f535c3d

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	2	1	1
Informational	0	0	0
Minor	1	0	1
Medium	1	1	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

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 2 issues of varying severity, listed below.

ID	Title	Severity	Status
ASI-1	Self Dos in Rebalance	Medium	Fixed
DIS-1	Lack of Events Emit	Minor	Acknowledged

3 Participant Process

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

- The `Admin` can create an investor through `create_investor()` .
- The `Admin` can add reward eligible field throguh `add_rewards_eligible_field()` .
- The `Admin` can set reward eligible field through `set_rewards_eligible_field()` .
- The `Admin` can set the performance fee of an investor through `set_performance_fee()` .
- The `Admin` can take liquidity from the old position in chunks and supply to a new range through `rebalance()` .
- The `Admin` can open a new position with a wider range and shift all assets from the previous position to the new position through `reposition()` .
- The `Admin` can create a pool or a distributor through `create()` .
- The `Admin` can set deposit fee or withdraw fee through `set_deposit_fee()` or `set_withdrawal_fee()` .
- The `Admin` can set lock period through `set_lock_period()` .
- The `Admin` can add more admins through `add_admin()` .
- The `Admin` can start the alpha minting through `set_start_timestamp_and_destroy_cap()` .
- The `Admin` can change fee receiving, airport,team,dust and onhold receipts addresses through `change_fee_wallet_address()` , `change_airdrop_wallet_address()` , `change_team_wallet_address` `change_dust_wallet_address()` , `change_onhold_receipts_wallet_address()` .
- The `Admin` can add a new pool to allocator through `add_new_pool()` .
- The `Admin` can remove a reward type through `remove_unlock_per_second()` .
- The `Admin` can set weights of multiple pools for a particular reward type through `set_weights()` .
- The `Admin` can add a new reward to be distributed to pools through `add_reward()` .

User

- The `User` can collects all trade fee rewards from the trades happening in the range and reinvest them as liquidity in the position to autocompound user's supply through `autocompound()` .
- The `User` can receive all rewards for the current pool through `get_user_rewards_all()` .
- The `User` can supply tokens to a particular pool through `user_deposit()` or `deposit()` .
- The `User` can withdraw tokens from a particular pool through `user_withdraw()` or `withdraw()` .
- The `User` can updates the pool with all rewards through `get_pool_rewards_all()` .
- The `User` can transfer tokens from `team_wallet_balance` to `team_wallet_address` after 300 days through `withdraw_team_balance()` .
- The `User` can transfer tokens from `airdrop_wallet_balance` to `airdrop_wallet` through `withdraw_airdrop_balance()` .

4 Findings

ASI-1 Self Dos in Rebalance

Severity: Medium

Status: Fixed

Code Location:

alphafi/sources/cetus_pool/alpha_sui_investor.move#177;

alphafi/sources/cetus_pool/cetus_investor.move#178

Descriptions:

In the `rebalance` function, the liquidity per loop and the remaining liquidity (`leftover`) are calculated first, and then the corresponding amount of liquidity is removed from the `cetus pool` in each loop. However, when the allocated liquidity is exactly divisible by the number of loops, the `leftover` will be 0. In the final iteration, this results in removing 0 liquidity from the `cetus pool` , causing the program to terminate because the `cetus pool` requires the removed liquidity amount to be greater than 0, otherwise it will abort.

```
let mut liquidity_per_loop = liquidity/(loops as u128);
        let leftover = liquidity % (loops as u128);
        let mut i = 0;
        while(i < (loops+1)){
            if( i == loops){
                liquidity_per_loop = leftover;
            };

            let (bal_a, bal_b) = pool::remove_liquidity<T,S>(config, pool, &mut position,
liquidity_per_loop, clock);
//...
}
```

The function `remove_liquidity()` of cetus is called as follows.

```
public fun remove_liquidity<CoinTypeA, CoinTypeB>(
        config: &GlobalConfig,
```

```
        pool: &mut Pool<CoinTypeA, CoinTypeB>,
        position_nft: &mut Position,
        delta_liquidity: u128,
        clock: &Clock,
    ): (Balance<CoinTypeA>, Balance<CoinTypeB>) {
        checked_package_version(config);
        assert!(!pool.is_pause, EPoolIsPaused);
        assert!(delta_liquidity > 0, ELiquidityIsZero);
//...
}
```

Suggestion:

It is recommended to end the loop when the `leftover` is 0.

Resolution:

The client modified the code and fixed this issue.

DIS-1 Lack of Events Emit

Severity: Minor

Status: Acknowledged

Code Location:

alphafi/sources/distributor.move#169-197

Descriptions:

The contract lacks appropriate events for monitoring sensitive operations, which could make it difficult to track sensitive actions or detect potential issues.

Suggestion:

It is recommended to emit events for those important functions.

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.