Kai Leverage

Audit Report

contact@bitslab.xyz

https://twitter.com/movebit_

Tue Aug 06 2024

Kai Leverage Audit Report

1 Executive Summary

1.1 Project Information

Description	A lending liquidity aggregation protocol
Type	DeFi
Auditors	MoveBit
Timeline	Tue Jun 25 2024 - Tue Aug 06 2024
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/kklas/kai-leverage-movebit
Commits	cb6472de651a3a70c17f2a175ffc4d55d41586c9 8e5d2dd6a382d325f24a3c428efb393cc52423a1 60fde849b0ae0428470fd97bdcd383881aa2f76a

1.2 Files in Scope

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

ID	File	SHA-1 Hash
AIN	kai-leverage/sources/access_init.move	4cbc6488eae731889593520adde6cf7084309d66
UTI1	kai-leverage/sources/util.move	d97b9e357c2b8b9d05695a16f5413649ef313d94
PIE	kai-leverage/sources/primitives/piecewise.move	7b615f04eb976475897c1ac5490143afa8fe32e2
FLO	kai-leverage/sources/clmm/flowx.move	1908d4772f0db6190d4610c0555a398b19edf545
PMO	kai-leverage/sources/clmm/position_model.move	2e0f6e57b6ec180775612562bb6a0dfbd599b51e
PCO	kai-leverage/sources/clmm/position_core.move	696500a61e35444e436613edabd3201d61483b14
TUR	kai-leverage/sources/clmm/turbos.move	36c0131381e951863e5ee7aa32a0427db67f003b
CET	kai-leverage/sources/clmm/cetus.move	70b91d4cec5edd34dede7a050d2b9081fdb6e39f
SPO	kai-leverage/sources/supply_pool.move	20ab7a23a00eb95b094f4b68cef566518b32a5f5
BBA	kai-leverage/sources/primitives/balance_bag.move	04d22050bbd79420c14c5b0c86172747628a256c
EQU	kai-leverage/sources/primitives/equity.move	f4f98213027bb7383abe26a732f49abc620dd24f

DEB	kai-leverage/sources/primitives/debt.move	504b164b93b2097f6aa642cf5a73b0133e7b2ed2
DBA	kai-leverage/sources/primitives/debt_bag.move	0cd2899b773120ed3e8f1c3494b41d08fad54a82
DIN	kai-leverage/sources/debt_info.move	0b7171f4e5c23b010df48cae8b126f2045a35343
PYT1	kai-leverage/sources/pyth.move	98811b42d10e555feacaafc69f61d73850affc09
ACC	access-management/sources/access.move	f431d0f8101928d4696249b1bb601dd6a6377f69
DMA	access-management/sources/dynamic_map.move	1b3c2184521260ad0996e4d8e4a5a4542518c212

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	4	4	0
Informational	0	0	0
Minor	0	0	0
Medium	1	1	0
Major	3	3	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 Kuna Labs to identify any potential issues and vulnerabilities in the source code of the Kai Leverage 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 4 issues of varying severity, listed below.

ID	Title	Severity	Status
POS-1	The Current Price Range Check is Incorrect	Major	Fixed
PYT-1	Getting the `expo` from `pyth` is Incorrect	Major	Fixed
SPO-1	The Administrator will Receive fewer Shares than Expected	Medium	Fixed
POS1-1	Create Position Fails	Major	Fixed

3 Participant Process

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

Admin

- The admin can set whether new positions are allowed through the `set_allow_new_positions` function.
- The admin can set the minimum liquidity start price delta in basis points through the `set_min_liq_start_price_delta_bps` function.
- The admin can set the minimum initial margin in basis points through the `set_min_init_margin_bps` function.
- The admin can add an empty Pyth configuration through the `config_add_empty_pyth_config` function.
- The admin can set the maximum age of the Pyth configuration in seconds through the `set_pyth_config_max_age_secs` function.
- The admin can allow a Pyth price information oracle (PIO) through the `pyth_config_allow_pio` function.
- The admin can disallow a Pyth price information oracle (PIO) through the `pyth_config_disallow_pio` function.
- The admin can set the deleverage margin in basis points through the `set_deleverage_margin_bps` function.
- The admin can set the base deleverage factor in basis points through the `set_base_deleverage_factor_bps` function.
- The admin can set the liquidation margin in basis points through the `set_liq_margin_bps` function.
- The admin can set the base liquidation factor in basis points through the `set_base_liq_factor_bps` function.
- The admin can set the liquidation bonus in basis points through the `set_liq_bonus_bps` function.
- The admin can set the maximum position liquidity through the `set_max_position_l` function.

- The admin can set the maximum global liquidity through the `set_max_global_l` function.
- The admin can set the rebalance fee in basis points through the `set_rebalance_fee_bps` function.
- The admin can set the liquidation fee in basis points through the `set_liq_fee_bps` function.
- The admin can set the position creation fee in SUI through the `set_position_creation_fee_sui` function.

User

- The user can create a position ticket through the `create_position_ticket` function.
- The user can borrow X for a position through the `borrow_for_position_x` function.
- The user can borrow Y for a position through the `borrow_for_position_y` function.
- The user can create a position through the `create_position` function.
- The user can create a position ticket through the `create_position_ticket` function.
- The user can create a deleverage ticket through the `create_deleverage_ticket` function.
- The user can deleverage through the `deleverage` function.
- The user can liquidate collateral X through the `liquidate_col_x` function.
- The user can liquidate collateral Y through the `liquidate_col_y` function.
- The user can reduce through the `reduce` function.
- The user can add liquidity through the `add_liquidity` function.
- The user can add liquidity with a fixed coin amount through the `add_liquidity_fix_coin` function.
- The user can repay debt X through the `repay_debt_x` function.
- The user can repay debt Y through the `repay_debt_y` function.
- The user can collect fees during rebalancing through the `rebalance_collect_fee` function.

- The user can collect rewards during rebalancing through the `rebalance_collect_reward` function.
- The user can add liquidity during rebalancing through the `rebalance_add_liquidity` function.
- The user can add liquidity with a fixed coin amount during rebalancing through the `rebalance_add_liquidity_by_fix_coin` function.

4 Findings

POS-1 The Current Price Range Check is Incorrect

Severity: Major

Status: Fixed

Code Location:

kai-leverage/vendor/flowx-clmm/sources/position.move#593

Descriptions:

In the `position.create_position_ticket()` function, the protocol performs a check to ensure the current price is within the range of the LP position.

```
// assert that the current price is within the range of the LP position
assert!(tick_a.gte(current_tick), EInvalidTickRange);
assert!(current_tick.lt(tick_b), EInvalidTickRange);
```

However, there is an issue with this check; the `current_tick` should be greater than or equal to `tick_a`.

Suggestion:

It is recommended to use `assert!(current_tick.gte(tick_a), EInvalidTickRange)`.

Resolution:

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

PYT-1 Getting the `expo` from `pyth` is Incorrect

Severity: Major

Status: Fixed

Code Location:

kai-leverage/vendor/pyth/sources/pyth.move#105

Descriptions:

In the `pyth.get_price_lo_hi_expo_dec()` function, the protocol calls `i64::get_magnitude_if_positive(&price.get_expo())` to get the expo.

```
fun get_price_lo_hi_expo_dec(
    price_info: &ValidatedPythPriceInfo, t: TypeName
): (u64, u64, u64, u64, u64) {
    let price = get_price(price_info, t);

    let conf = price.get_conf();
    let p = i64::get_magnitude_if_positive(&price.get_price());
    let expo = i64::get_magnitude_if_positive(&price.get_expo());
    let dec = decimals(t) as u64;

    (p, p - conf, p + conf, expo, dec)
}
```

However, in the Sui ecosystem, most tokens have a negative expo value. https://pyth.network/price-feeds Therefore, `get_magnitude_if_positive()` will throw an error, causing the program to fail.

Suggestion:

It is recommended to use to the following method to get the expo.

```
let expo = if i64::get_is_negative(&i64_expo) {
    i64::get_magnitude_if_negative(&i64_expo)
} else {
    i64::get_magnitude_if_positive(&i64_expo)
};
```

Resolution:

This issue has been fixed. The protocol now calls `get_magnitude_if_negative()` to retrieve the exponent.

SPO-1 The Administrator will Receive fewer Shares than Expected

Severity: Medium

Status: Fixed

Code Location:

kai-leverage/sources/supply_pool.move#402-407

Descriptions:

In the `supply_pool.repay_flash_loan()` function, if `equity::join()` is called after `share_registry.increase_value()`, the shares obtained will be reduced. From the project team's perspective, it is recommended to call `equity::join()` before `share_registry.increase_value()`.

```
let share_registry = pool.supply_equity.borrow_mut_registry();
    share_registry.increase_value(repay_amt - interest_fee);
    equity::join(
        &mut pool.collected_fees,
        share_registry.increase_value_and_issue(interest_fee)
    );
    pool.available_balance.join(balance);
```

If `registry.underlying_value_x64` and `registry.supply_x64` are both 100, and a user borrows 100 with an interest rate of 10%, the interest is 100 * 10% = 10. If `interest_fee_bps` is 50%, the `interest_fee` is 5. In the current implementation, the final `underlying_value` is 100 + 10 = 110, and `supply` is 100 + 5 * 100 / 105 = 104.7. If `equity::join()` is called before `share_registry.increase_value()`, the administrator can get `shares = 5 * 100 / 100 = 5` shares.

Suggestion:

It is recommended to call the `equity::join()` function before calling `share_registry.increase_value()`.

Resolution:

This issue has been fixed. The protocol has updated the value of `share_registry.increase_value()` to `interest - interest_fee`.

POS1-1 Create Position Fails

Severity: Major

Status: Fixed

Code Location:

kai-leverage/sources/clmm/position.move#642-778

Descriptions:

If the user executes `borrow_for_position_x` or `borrow_for_position_y` after creating a position ticket, in `borrow_for_position_x`, the user borrows an amount of `ticket.dx` into `borrowed_x` and then sets `ticket.dx` to 0:

```
let (balance, shares) = supply_pool.borrow(&config.lend_facil_cap, ticket.dy, $clock);
ticket.borrowed_y.join(balance);
ticket.dy = 0;
```

Then the creation of the position will fail because of the following checks:

```
assert!(ticket.borrowed_x.value() == ticket.dx, EInvalidBorrow);
assert!(ticket.borrowed_y.value() == ticket.dy, EInvalidBorrow);
```

Suggestion:

It is recommended to modify the appeal logic to resolve the issue.

Resolution:

This issue has been fixed. The client has modified the appeal logic to resolve the issue.

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.