Abex
Smart Contract
Audit Report

MOVEBIT

contact@movebit.xyz

https://twitter.com/movebit_

08/01/2023

Abex Smart Contract Audit Report

1 Executive Summary

1.1 Project Information

Description	ABEx is an on-chain derivatives & swap protocol.
Type	DeFi
Auditors	MoveBit
Timeline	July 03, 2023 – Aug 1, 2023
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/abex-finance/abex-contracts
Commits	cf341bc54843fff3a3152278e3f6e9dde22b6fdf 8ae0097a6285ed37e2f2fb30e50236beb7e9b2f4 c88ae942c351840675cc31331ba3a4042bbbf4e9

1.2 Files in Scope

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

ID	Files	SHA-1 Hash
PL	sources/pool.move	309e566168d1bf4371317499a3a1e27a1068d6ba

ADM	sources/admin.move	bb1c7c9c45b22455ae96df39780aec7d6bfab63f
AP	sources/agg_price.move	6475857a0e1943679516b11472a7da93a0e1afbc
SRT	sources/math/srate.move	dad2e5385bd3cb268a3454a76123b456b9f75f17
DEM	sources/math/decimal.move	6786d0676b8a3fe55a15ae6cd259866a5ea5f08a
RAT	sources/math/rate.move	233fabcc3dc350faf3f1e63232253d19f8893b91
SDE	sources/math/sdecimal.move	946081ed966cd030a551f818163a8c0ae8a2b89e
MDE	sources/model.move	5d3cfaf7eb2e6e2bae74c622e6452e261b7f490a
ALP	sources/alp.move	8fef4b0ef8e1c8e535191fc53875ee5a6f12b868
MKT	sources/market.move	7221e2bfd43244919dc56ef0c0480a5b502907e8
PST	sources/position.move	79334a4bc2cd004a69d74f0cddb9364ec7e57483
REF	sources/referral.move	6c6e2e26657d8a6682f90f0fcfa224d6e4ac69ab
ORD	sources/orders.move	a5797d5b7506d107135e9719aecacda9592b0d5b
MV	abex-core/Move.toml	3749d3b0f9e7825abd0fcde919db3414e79ba67c

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	8	7	1
Informational			
Minor	4	4	
Medium	2	2	
Major	1		1
Critical	1	1	

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 **Abex** to identify any potential issues and vulnerabilities in the source code of the **Abex** 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 **8** issues of varying severity, listed below.

ID	Title	Severity	Status

MKT-01	Lack of A Method to Add `referrals` in the `Market`	Medium	Fixed
MKT-02	Pending Order Fee Tokens not Tied to Valid Tokens	Major	Acknowledged
MKT-03	Unused Private Function	Minor	Fixed
MKT-04	Lack of Event	Minor	Fixed
MKT-05	Outdated Variable `VaultsValuation`	Critical	Fixed
REF-06	Unused Friend Function	Minor	Fixed
ORD-07	Unused Constants	Minor	Fixed
SRT-08	The Value of 0 for Both States	Medium	Fixed

3 Participant Process

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

Admin

- Admin can create a new `Vault` through `add_new_vault<L, C>()`.
- Admin can create a new `Symbol` through `add_new_symbol<L, I, D>()`.
- Admin can add a new type of collateral for the `Symbol` through `add_collateral_to_symbol<L, C, I, D>()`.
- Admin can remove a type of collateral for the `Symbol` through `remove_collateral_from_symbol<L, C, I, D>()`.

User

- User can open a new position or pending order open a new position through `open_position<L, C, I, D, F>()`.
- User can decrease their position through `decrease_position<L, C, I, D, F>()`.
- User can decrease their reserve amount in the positon through `decrease_reserved_from_position<L, C, I, D>()`.
- User can add collateral into their position through `pledge_in_position<L, C, I, D>()`.

- User can redeem collateral from their position through `redeem_from_position<L, C, I, D>()`.
- User can clear the closed position through `clear_closed_position<L, C, I, D>()`.
- User can set the amount of profit or loss through `take_profit_or_stop_loss<L, C, I, D>()`.
- User can clear the open position order through `clear_open_position_order<C, I, D, F>()`.
- User can clear the decrease position order through `clear_decrease_position_order<C, I, D, F>()`.
- User can clear the take profit or stop loss order through `clear_take_profit_or_stop_loss_order<C, I, D, F>()`.
- User can deposit `C` coin into the vault to inject liquidity and get `ALP` coin through `deposit<L, C>()`.
- User can burn the `ALP` coin to withdraw the liquidity an get the `C` coin through `withdraw<L, C>()`.
- User can swap the `S` coin and get the `D` coin in the vault through `swap<L, S, D>()`.

Order Executor

- Order Excutor can execute the open position order through `execute_open_position_order<L, C, I, D, F>()`.
- Order Excutor can execute the decrease position order through `execute_decrease_position_order<L, C, I, D, F>()`.
- Order Excutor can execute the take profit or stop loss order through `execute_take_profit_or_stop_loss_order<L, C, I, D, F>`.

Liquidator

- Liquidator can liquidate the position through `liquidate_position<L, C, I, D>()`.

4 Findings

MKT-01 Lack of A Method to Add `referrals` in the `Market`

Severity: Medium

Status: Fixed

Code Location: sources/market.move#L335

Descriptions: There is a mechanism to get rebates in the contract, but in the `Market`, the referrals are empty by default, with no method to add, which equals that no one can get rebates, and there is a problem with the setup of this mechanism.

Suggestion: It is recommended to add a method to add rebate users.

Resolution: The client followed our suggestion and fixed this issue.

MKT-02 Pending Order Fee Tokens not Tied to Valid Tokens

Severity: Major

Status: Acknowledged

Code Location: sources/market.move#L461, 610, 748, 981

Descriptions: The method of opening and reducing a position in a contract requires the passing of a pending order's fee `Coin<F>`, but this fee token can be passed in a fake coin minted by itself, so that when a pending order executor comes to execute the pending order it receives a fake fee and loses the interests of the pending order executor.

Suggestion: It is recommended to bind the token for the pending order fee to a valid token in the vault.

MKT-03 Unused Private Function

Severity: Minor

Status: Fixed

Code Location: sources/market.move#L224

Descriptions: The private function `burn_lp` is not used.

Suggestion: It is recommended to remove the private function, or change the code to make it useful.

Resolution: The client followed our suggestion and fixed this issue.

MKT-04 Lack of Event

Severity: Minor

Status: Fixed

Code Location: sources/market.move#L345

Descriptions: The smart contract lacks appropriate events for monitoring sensitive operations, which could make it difficult to track important actions or detect potential issues. Such as: `add_new_vault` , `add_new_symbol` , `add_collateral_to_symbol` , `remove_collateral_from_symbol` .

Suggestion: It is recommended to emit an event for these update functions.

Resolution: The client followed our suggestion and fixed this issue.

MKT-05 Outdated Variable `VaultsValuation`

Severity: Critical

Status: Fixed

Code Location: sources/market.move#L1245

Descriptions: Create two `VaultsValuation` in a single transaction, and `valuate_vault` is called sequentially to update them, ensuring that both created `VaultsValuation` meet the validation criteria. At this point, the first `VaultsValuation` is used for deposit while the second one remains unchanged. After completing the deposit, the total value in the market will increase, but the data in the second `VaultsValuation` remains unchanged. Subsequently, using the second `VaultsValuation` for another deposit will result in a larger number of LP tokens than what would normally be obtained through regular operations. This creates an arbitrage opportunity and can lead to the depletion of assets.

Suggestion: It is recommended to take measures to avoid such issues.

Resolution: The client added a lock to avoid re-valuation and fixed this issue.

REF-06 Unused Friend Function

Severity: Minor

Status: Fixed

Code Location: sources/referral.move#L12, 19

Descriptions: The friend functions `new_referral` and `refresh_rebate_rate` are not used.

Suggestion: It is recommended to remove the friend function, or change the code to make it useful.

Resolution: The client followed our suggestion and fixed this issue.

ORD-07 Unused Constants

Severity: Minor

Status: Fixed

Code Location: sources/orders.move#L20, L23

Descriptions: The constants `ERR_MISMATCHED_DECREASE_INTENTION` and `ERR_INVALID_DECREASE_AMOUNT` are not used.

Suggestion: It is recommended to remove the unused constans.

Resolution: The client followed our suggestion and fixed this issue.

SRT-08 The Value of 0 for Both States

Severity: Medium

Status: Fixed

Code Location: sources/math/srate.move#L49

Descriptions: When the result of a calculation is 0, the state of the returned sRate is negative, which may result in two states of 0, positive 0 and negative 0. The same problem exists for add and sub. The same problem exists with add and sub. The same applies to sdecimal.

Suggestion: It is recommended to confirm that this meets the purpose of the design and check if it affects some of the calculations .

Resolution: The client followed our suggestion and fixed this 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.