Deeptrade

Audit Report

MOVEBIT

contact@bitslab.xyz

https://twitter.com/movebit_

Thu Oct 09 2025

Deeptrade Audit Report

1 Executive Summary

1.1 Project Information

Description	Deeptrade Core introduces a self-sustaining economic model centered around a core Treasury that manages protocol-owned liquidity for DEEP tokens. This allows for a more accessible trading experience by abstracting the native fee token requirements of the underlying order book
Type	DEX
Auditors	Alex,PeiQi,hyer,Light
Timeline	Sun Sep 07 2025 - Thu Oct 09 2025
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/DeeptradeProtocol/deeptrade-core
Commits	aa073bd515b92af99e155a7a92b1a8a593aa99bf 33a1b96b742cef5c11fddbc5a145267bad22f315 f2846cfd3eb5baf58ae929b1ed157643e420db80 03f31c1f0788eecbaf99f511172426f7a1735c6e

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MOV	packages/deeptrade-core/Move.toml	be312c732ab7a931ef2ed511161de996ae8e0216
ADM	packages/deeptrade-core/sources/admin.move	e1a77a368bb709d5dbba1214f7f45a0657bd48be
MAT	packages/deeptrade-core/sources/math.move	043978207440a32c7d316bf1834035d916b3abc3
FEE	packages/deeptrade-core/sources/fee.move	2d095f02ac7eb8199b43bcb2b13e857a3b60ed83
POO	packages/deeptrade-core/sources/pool.move	dfc91842b712230762c6432cb390de6606156442
ORA	packages/deeptrade-core/sources/oracle.move	3cac7da9f1730502b35dc6e120800c42e912ceab
SWA	packages/deeptrade-core/sources/swap.move	ba2a796a146359ffe082a46f99334c313dda2287
FMA	packages/deeptrade-core/sources/fee_manager.move	e062b9e447c027e40e796287cb9d58f31619b9a4
LOY	packages/deeptrade-core/sources/loyalty.move	cc0c2f76799366b597fad56e0bf73238615877de
TIC	packages/deeptrade-core/sources/ticket.move	e32c5fe386aa141406b2935310ab610060ab657b

TRE	packages/deeptrade-core/sources/treasury.move	4aab7de386ee81c12d173c9b879b26bf24442170
HEL	packages/deeptrade-core/sources/helper.move	dfee54e6966a9825414acddf4ee5889972fe1afc
MCO	packages/deeptrade-core/sources/multisig_config.move	34b8d4f29a42cba7e424be687b704a56fca52905
ORD	packages/deeptrade-core/sources/order.move	871ec507220412fb475d6288127bb647b04ac2c1

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	1	1	0
Informational	0	0	0
Minor	1	1	0
Medium	0	0	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 Deeptrade to identify any potential issues and vulnerabilities in the source code of the Deeptrade 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 1 issues of varying severity, listed below.

ID	Title	Severity	Status
FMA-1	`check_if_sender_is_multisig_address` Usage Method Is Incorrect	Minor	Fixed

3 Participant Process

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

Admin:

- `enable_version` : Enables a new package version.
- `disable_version` : Permanently disables an old package version.
- `add_loyalty_level` / `remove_loyalty_level` : Manages the available loyalty tiers.
- `update_loyalty_admin_cap_owner` : Transfers ownership of the `LoyaltyAdminCap` .
- `claim_user_unsettled_fee_storage_rebate_admin` : Claims storage rebate for a user's settled fee.
- `claim_protocol_unsettled_fee_storage_rebate_admin` : Claims storage rebate for a settled protocol fee.
- `update_pool_creation_protocol_fee` : Changes the protocol fee required to create a new pool.
- `update_default_fees` : Changes the global, default trading fee configuration.
- `update_pool_specific_fees` : Sets or updates a custom trading fee configuration for a specific pool.
- `grant_user_level` : Assigns a loyalty tier to a specific user.
- `revoke_user_level` : Removes a loyalty tier from a specific user.
- `update_loyalty_admin_cap_owner` : Update the owner of the LoyaltyAdminCap (requires multisig + AdminCap).
- `grant_user_level` : Assign a loyalty level to a user.
- `revoke_user_level` : Revoke a user’s loyalty level.
- `add_loyalty_level` : Add a new loyalty level and define its discount rate.
- `remove_loyalty_level` : Remove a loyalty level (only if no users are assigned to it).

- `update_pool_creation_protocol_fee` : Update the protocol fee for creating a pool.
- `create_ticket` :Creates a new admin ticket with multi-signature verification for future execution.
- `destroy_ticket` : Consumes and deletes a ticket, marking it as expired or executed.
- `validate_ticket` : Validates a ticket before execution (ownership, type, readiness, expiration).
- `enable_version` : Enables a specific package version for the treasury with multisig verification.
- `disable_version` : Permanently disables a specific package version for the treasury with multisig verification.
- `withdraw_deep_reserves` : Withdraws a specified amount of DEEP tokens from reserves using an admin ticket.
- `withdraw_protocol_fee` : Withdraws all accumulated protocol fees of a given coin type using an admin ticket.
- `withdraw_coverage_fee` : Withdraws all accumulated coverage fees of a given coin type using an admin ticket.

User:

- `estimated_deep_required` :User's estimate of DEEP tokens needed for an order creation.
- `estimated_deep_required_slippage` :Maximum acceptable slippage.
- `estimated_sui_fee` - User's estimate of SUI coverage fee.
- `estimated_sui_fee_slippage` :Maximum acceptable slippage for the coverage fee.
- `get_user_loyalty_level` : Get the loyalty level assigned to a user.
- `get_loyalty_level_fee_discount_rate` : Get the fee discount rate of a given loyalty level.
- `get_user_discount_rate` : Get the discount rate applied to a user (0 if no loyalty level).
- `get_level_member_count` : Get the number of users in a specific loyalty level.

- `total_loyalty_program_members` : Get the total number of members in the loyalty program.
- `create_permissionless_pool` : Create a new permissionless trading pool (pays DeepBook fee + protocol fee in DEEP).
- `pool_creation_protocol_fee` : Get the current protocol fee for creating a pool.
- `swap_exact_base_for_quote_input_fee` : Swap base tokens for quote tokens.
- `swap_exact_quote_for_base_input_fee` : Swap quote tokens for base tokens.
- `get_quantity_out_input_fee` : Estimate expected output amounts for a swap.
- `cleanup_expired_ticket` : Allows anyone to remove an expired ticket from the system.
- `ticket_active_duration` : Returns the active duration window after the delay.
- `deposit_into_reserves` : Deposits DEEP tokens into the treasury reserves.
- `deep_reserves` : Returns the current DEEP balance in the reserves.
- `create_limit_order` : Creates a limit - order on DeepBook using coins from wallet and treasury reserves, applying DEEP and SUI fees with slippage checks.
- `create_market_order` : Creates a market order on DeepBook using coins from wallet and treasury reserves, applying DEEP and SUI fees with slippage checks.
- `create_limit_order_whitelisted` : Creates a limit order on DeepBook for whitelisted pools using only wallet coins, optimized without treasury reserves.
- `create_market_order_whitelisted` : Creates a market order on DeepBook for whitelisted pools using only wallet coins, optimized without treasury reserves.
- `create_limit_order_input_fee` : Creates a limit order on DeepBook using input coins as fees instead of DEEP, applying loyalty discount only.
- `create_market_order_input_fee` : User places a market order using input coins to pay fees.
- `cancel_order_and_settle_fees` : User cancels an order and settles any outstanding fees.

4 Findings

FMA-1 `check_if_sender_is_multisig_address` Usage Method Is Incorrect

Severity: Minor

Status: Fixed

Code Location:

packages/deeptrade-core/sources/fee_manager.move#293

Descriptions:

Multiple functions verify whether the caller is a valid multi-signature address by calling multisig::check_if_sender_is_multisig_address. However, all the key parameters used for verification, such as the public key list pks, weights, and thresholds, are directly provided by the function caller in the transaction. An attacker can construct a set of multi-signature parameters of their own. For instance, they can create a multi-signature configuration that only contains their own public key and set the threshold threshold to 1. Since the contract itself does not store or reference an authoritative, pre-configured multi-signature address configuration. It fully trusts that the parameters provided by the caller are correct, resulting in the construction of a 1-of-1 multi-signature configuration pointing to itself and successfully passing the verification··

Suggestion:

When calling multisig::check_if_sender_is_multisig_address, verify using trusted configuration data loaded from within the contract

Resolution:

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

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.