Moar Market

Audit Report

contact@bitslab.xyz

https://twitter.com/movebit_

Wed Dec 11 2024

Moar Market Audit Report

1 Executive Summary

1.1 Project Information

Description	Moar Market is the Credit Layer for Aptos DeFi. It introduces the concept of Composable Leverage - giving you credit (up-to 25x) to use across DeFi Protocols
Type	DeFi
Auditors	MoveBit
Timeline	Mon Oct 14 2024 - Wed Dec 11 2024
Languages	Move
Platform	Aptos
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/moar-market/cerebro https://github.com/moar-market/tiered-oracle
Commits	https://github.com/moar-market/cerebro: 8b85a2e94c4133d52d13eb58bd8c9541e46f71f6 504749c88ade6e11817c9102fbcf7b20a4e86446 c16764e3f8c99f72d0d29efc073311524ba63551 d31c4f090011615190adf8cbcb0cbb1dbbcc0fc9 96b7d024dbb8c228cfe34d8f60905ce85ae362bb 4ada426f82d2a61304929a324798c8c55bbf2038 https://github.com/moar-market/tiered-oracle: b0975e9452e50147bea8343c6f8a07b2858ec5c2 28ac0ca1ed236319e68bf832365d1410cd684f39

1.2 Files in Scope

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

ID	File	SHA-1 Hash
TOR	sources/tiered_oracle.move	20abd7c11203a05dd971d1772a10975310c87d58
MAT	sources/math.move	668465400e2e79f6581fda11340dd6d3791eb66f
SOR	sources/switchboard_oracle.move	8d51190f558cc1bfdd6c081fcf8ca53cc2c91c64
POR	sources/pyth_oracle.move	ca773aa13d56014019ecc8d711e7c180fbe45c62
STA	sources/status.move	0a84e3920ef8e5f08b78e6f170b25a859002ac09
MOV	moar-liquidator/Move.toml	3fa46b66fa0364ff76277b3c69f6a676bac50622
MOV3	Move.toml	7408f94bf1b5b3e7dd82fde11703ab409c676389
UTI	moar_strats/sources/utils.move	fc9f5d2266c2e858023e7a0a8c8bba0a0786f7a3
CWR	sources/coin_wrapper.move	7352d1e03a0fd3324522fb661e8e192baa3dd290
ORA	sources/oracle.move	95efb0e9b6d14e492a7d409b8b37d31dafd0eb02
PMA	sources/package_manager.move	a706c7052718687a007592d92a6293b5dbdaf13c

LTE	moar-liquidator/sources/tests/liquidator_tests.move	0660df3e50944e903ca1b78331fb100fafdda63e
LIQ	moar-liquidator/sources/liquidator.move	3ae95c8b29fec79a6eb9a1acbdf4e71d16988455
TAD	moar_strats/sources/adapters/thala_adapter.move	9ef7077d97df8b4100c01aeb8f23c40779ab37e5
PAD	moar_strats/sources/adapters/panora_adapter.move	e6d903ad6930e3d15294cd26a6f784baf9788cf9
ROU1	sources/router.move	cd99f1d7394f5910e1bc81fd1aa2b84316a6aeec
LEN	sources/lens.move	30bed8e65e87aed6caa1db8afdc7e4e36ed1cfd0
CMA	sources/credit_manager.move	12348a4fc8164b9f533006373cd0e3268c238b41
POO1	sources/pool.move	690a7d78093499d2d67fc78fdc5b3fe0b2889de2
RMA	sources/risk_manager.move	6ac41cd688300a66367122ec8a2e791e3cb3cc44
IRM	sources/interest_rate_model.move	05e2320f18ac4b770d3bf3dbb22e685924abc918
UTI1	sources/utils.move	e07342f235f50ea232d57e80596fee6106204b29
MOV8	moar_strats/Move.toml	08fbc38e063907d07f500efa0fc9a40eaeca5193
LAD	moar_strats/sources/adapters/liquidswap_adapter.move	7613c43cac0c153e5a1a6d03cba4b7cbb885691f

ROU	moar_strats/sources/router.move	36822c369cf96c41267a4b2899357b61a1178297

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	11	10	1
Informational	1	1	0
Minor	2	1	1
Medium	7	7	0
Major	1	1	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 Moar to identify any potential issues and vulnerabilities in the source code of the Moar Market 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 11 issues of varying severity, listed below.

ID	Title	Severity	Status
CMA-1	Lacks A Function To Remove Allowed Assets	Medium	Fixed
CMA-2	Missing a Function to Handle Bad Debt	Medium	Fixed
CMA-3	Unused Constants	Minor	Fixed
MOV-1	The Tiered Oracle has no Upgrade Strategy	Medium	Fixed
POO-1	The `deposit()` Function does not Account for Exceptional Cases	Medium	Fixed
POO-2	The Calculation of Shares during Withdrawal should Round Up	Medium	Fixed
POO-3	The `icon_url` Has no Value	Minor	Acknowledged
POR-1	Variable Naming Inconsistency in `set` Function	Informational	Fixed
TOR-1	The Protocol should Abort when `get_price()` Returns Stale Prices	Major	Fixed

TOR-2	The `last_price` Is Not Updated	Medium	Fixed
TOR-3	Different Assets Should Have Varying Expiration Times for Their Prices	Medium	Fixed

3 Participant Process

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

User

- Users can provide liquidity through the `deposit_entry` function.
- Users can remove liquidity through the `withdraw` function.
- Users can execute actions through the `execute_actions` function.
- Users can create a credit account through the `create_credit_account` function.
- Users can deposit collateral through the `deposit_collateral_entry` function.
- Users can withdraw collateral through the `withdraw_entry` function.
- Users can add a credit account asset through the `add_credit_account_asset_entry` function.
- Users can remove a credit account asset through the `remove_credit_account_asset_entry` function.
- Users can borrow through the `borrow_entry` function.
- Users can repay through the `repay` function.
- Users can execute strategies through the `execute_strategy_public` function.
- Users can liquidate through the `liquidate_entry` function.
- Users can close a credit account through the `close_credit_account` function.

Governance

- Governance can add an allowed asset through the `add_allowed_asset` function.
- Governance can set the piecewise linear model through the `set_piecewise_linear_model` function.
- Governance can set the liquidation discount through the `set_liquidation_discount` function.
- Governance can set the LTV through the `set_ltv` function.

4 Findings

CMA-1 Lacks A Function To Remove Allowed Assets

Severity: Medium

Status: Fixed

Code Location:

sources/credit_manager.move#695-699

Descriptions:

This function's purpose is to add allowed assets, but once added, there's no way to remove them. If an allowed asset is attacked or encounters an emergency situation, the protocol will be unable to remove it.

```
    public entry fun add_allowed_asset(sender: &signer, asset: Object<Metadata>)
acquires CreditManagerConfig {
        utils::governance_check(signer::address_of(sender));
        let credit_manager_config = unchecked_mut_credit_manager_config();
        smart_table::upsert(&mut credit_manager_config.allowed_assets, asset, true);
    }
```

Suggestion:

It is recommended to add a function for removing allowed assets.

Resolution:

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

CMA-2 Missing a Function to Handle Bad Debt

Severity: Medium

Status: Fixed

Code Location:

sources/credit_manager.move#1

Descriptions:

The protocol currently lacks a method to handle bad debt effectively, which could lead to unresolved debts impacting the protocol's overall health and users' funds.

https://github.com/sentimentxyz/protocol-v2/blob/master/src/PositionManager.sol#L454-L469

```
    /// @notice Liquidate a position with bad debt
    /// @dev Bad debt positions cannot be liquidated partially
    function liquidateBadDebt(address position, DebtData[] calldata debtData) external
nonReentrant {
        (DebtData[] memory repayData, AssetData[] memory seizeData) =
            riskEngine.validateBadDebtLiquidation(position, debtData);

        // liquidator repays some of the bad debt, and receives all of the position assets
        _transferAssetsToLiquidator(position, BAD_DEBT_LIQUIDATION_FEE, seizeData); //
zero protocol fee
        _repayPositionDebt(position, repayData);

        // settle remaining bad debt for the given position
        uint256[] memory debtPools = Position(payable(position)).getDebtPools();
        uint256 debtPoolsLength = debtPools.length;
        for (uint256 i; i < debtPoolsLength; ++i) {
            pool.rebalanceBadDebt(debtPools[i], position);
            Position(payable(position)).repay(debtPools[i], type(uint256).max);
        }
    }
```

Suggestion:

It is recommended to implement a liquidateBadDebt() function, similar to the approach used in the Sentiment contract.

Resolution:

This issue has been fixed, and the client has implemented the `liquidate_bad_debt_entry()` functionality.

CMA-3 Unused Constants

Severity: Minor

Status: Fixed

Code Location:

sources/credit_manager.move#44

Descriptions:

There are unused constants in the contract.

```
const E_NOT_FEE_RECIPIENT: u64 = 13;
```

```
const E_INVALID_DEBT_POOL_COUNT: u64 = 18;
```

Suggestion:

It is recommended to remove unused constants if there's no further design.

Resolution:

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

MOV-1 The Tiered Oracle has no Upgrade Strategy

Severity: Medium

Status: Fixed

Code Location:

Move.toml#3

Descriptions:

In the Moar Market protocol, we found that the upgrade strategy is compatible.

```
[package]
name = "moar"
version = "0.0.0"
upgrade_policy = "compatible"
authors = ["Moar Market (tech@moar.market)"]
```

However, in the Tiered Oracle protocol, there is no upgrade strategy.

```
[package]
name = 'TieredOracle'
version = '1.0.0'
```

Suggestion:

It is recommended to add a compatible upgrade strategy.

Resolution:

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

POO-1 The `deposit()` Function does not Account for Exceptional Cases

Severity: Medium

Status: Fixed

Code Location:

sources/pool.move#661-674

Descriptions:

The `deposit()` function allows users to deposit FA and receive LP tokens. The protocol calls `convert_amount_to_shares()` to calculate the shares, with the formula: `shares = amount * total_shares / total_amount`.

```
inline fun convert_amount_to_shares(
    amount: u64,
    total_amount: u128,
    total_shares: u128,
    is_round_up: bool,
): u64 {
    if (total_amount == 0) {
        amount
    } else if (is_round_up) {
        (utils::mul_div_with_ceil((amount as u128), total_shares, total_amount) as u64)
    } else {
        (math128::mul_div((amount as u128), total_shares, total_amount) as u64)
    }
}
```

However, the protocol might encounter an extreme case where `total_amount = 0` but `total_shares != 0`, causing the protocol to always mint at a 1:1 ratio. The Sentiment contract, which is nearly identical to this contract, has accounted for this scenario.

https://github.com/sentimentxyz/protocol-v2/blob/master/src/Pool.sol#L373

Suggestion:

It is recommended to revert when `pool.totalDepositAssets == 0 && pool.totalDepositShares != 0`.

Resolution:

This issue has been fixed, and the client handles the extreme scenario where the total pool assets are 0, but the total shares are not 0.

POO-2 The Calculation of Shares during Withdrawal should Round Up

Severity: Medium

Status: Fixed

Code Location:

sources/pool.move#408

Descriptions:

The purpose of the `withdraw()` function is to burn the given amount of LP tokens and transfer the underlying assets to the receiver's primary fungible store. In this function, the protocol calls `convert_amount_to_shares()` to calculate the shares that need to be burned.

```
let lp_address = signer::address_of(lp);
        let pool = get_pool_by_id(pool_id);
        let pool_data = pool_data(&pool);
        // Calculate the amounts of tokens redeemed from the pool.
        let lp_tokens_supply = option::destroy_some(fungible_asset::supply(pool));
        let shares = convert_amount_to_shares(amount, pool_data.total_deposited,
lp_tokens_supply, false);
        assert!(shares > 0, E_LP_ZERO_SHARES);
        assert!(!pool_data.is_paused, E_POOL_PAUSED);
```

It is important to note that the current implementation uses round down, but it should use round up.

Suggestion:

It is recommended to round up when calculating the shares to be burned during the withdrawal process.

Resolution:

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

POO-3 The `icon_url` Has no Value

Severity: Minor

Status: Acknowledged

Code Location:

sources/pool.move#622

Descriptions:

In the `create_lp_token()` function, the `icon_uri` is an empty string and should be a valid URI.

```
primary_fungible_store::create_primary_store_enabled_fungible_asset(
        lp_token_constructor_ref,
        option::none(),
        token_name,
        token_name,
        fungible_asset::decimals(token), // lp token decimals must be same as the
underlying token
        string::utf8(b""), // @todo change this
        string::utf8(b"https://moar.market/")
    );
```

Suggestion:

It is recommended to modify the value of the `icon_uri` string.

POR-1 Variable Naming Inconsistency in `set` Function

Severity: Informational

Status: Fixed

Code Location:

sources/pyth_oracle.move#22-23

Descriptions:

The variable name `switchboard_oracle` in the `set` function does not accurately reflect its purpose. It is intended to represent an instance of the `PythOracle` struct, but the naming could lead to confusion regarding its functionality and association with the Pyth oracle system.

```
public fun set(asset_oracle_signer: &signer, feed: vector<u8>) acquires PythOracle {
    let asset_oracle_address = signer::address_of(asset_oracle_signer);
    if (exists<PythOracle>(asset_oracle_address)) {
        let switchboard_oracle = borrow_global_mut<PythOracle>(asset_oracle_address);
        switchboard_oracle.feed = feed;
    } else {
        move_to(asset_oracle_signer, PythOracle {
            feed
        })
    }
}
```

Suggestion:

It is recommended to rename the variable `switchboard_oracle` to `pyth_oracle` within the `set` function.

Resolution:

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

TOR-1 The Protocol should Abort when `get_price()` Returns Stale Prices

Severity: Major

Status: Fixed

Code Location:

sources/tiered_oracle.move#109-147

Descriptions:

In the `get_price()` function, if `tier_1_status` is not normal and there is no tier_2 oracle, the protocol still returns the swap price.

```
    if (status::is_normal_status(tier_1_status) || asset_oracle.tier_2 == ORACLE_NULL) {
        return (tier_1_status, tier_1_price)
    };
```

This is incorrect, as stale prices can lead to improper estimation of users' collateral and potentially result in liquidation of their positions. Similarly, if there is a tier_2 oracle and both return stale prices, the function should also revert.

Suggestion:

It is recommended to revert when the protocol returns stale prices.

Resolution:

This issue has been fixed. The client has adjusted the staleness and brokenness durations according to our tolerance levels.

TOR-2 The `last_price` Is Not Updated

Severity: Medium

Status: Fixed

Code Location:

sources/tiered_oracle.move#41

Descriptions:

The `last_price` is not updated. Throughout the entire price retrieval process, there is no update to the `last_price`. This means that the `deviate_largely_from_old_price` function consistently returns the initial value of 0.

```
    smart_table::add(&mut oracle.assets, asset, AssetOracle {
        extend_ref: object::generate_extend_ref(&asset_constructor_ref),
        tier_1: 0,
        tier_2: 0,
        last_price: fixed_point64::zero(),
        staleness_seconds: DEFAULT_STALENESS_SECONDS,
        broken_seconds: DEFAULT_BROKEN_SECONDS,
        price_deviate_reject_pct: DEFAULT_PRICE_DEVIATE_REJECT_PCT,
    });
```

```
  public fun deviate_largely_from_old_price(
      price_deviate_reject_pct: u64,
      new_price: FixedPoint64,
      old_price: FixedPoint64
  ): bool {
      fixed_point64::to_u128(old_price) > 0 && get_price_diff_pct(new_price, old_price) >=
price_deviate_reject_pct
  }
```

Suggestion:

It is recommended to update the `last_price` when a valid price is obtained.

Resolution:

This issue has been fixed. The client has removed this feature.

TOR-3 Different Assets Should Have Varying Expiration Times for Their Prices

Severity: Medium

Status: Fixed

Code Location:

sources/tiered_oracle.move#88-90

Descriptions:

In the `register` function, the oracle's `staleness_seconds`, `broken_seconds`, and `price_deviate_reject_pct` for each asset are currently set using constant default configurations. However, the expiration times and related parameters for different assets should be adjustable individually, rather than being controlled solely by these constants. Different assets have varying levels of volatility.

```
const DEFAULT_STALENESS_SECONDS: u64 = 900;
const DEFAULT_BROKEN_SECONDS: u64 = 3600;
const DEFAULT_PRICE_DEVIATE_REJECT_PCT: u64 = 20;
```

Suggestion:

It is recommended to set different staleness_seconds, broken_seconds, and price_deviate_reject_pct values for different assets.

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.