# Navi Smart Contract Audit Report

## 1 Executive Summary

### 1.1 Project Information

| | |
|---|---|
| **Description** | Navi is a one-stop Liquidity Protocol on Sui. It enables users to participate as liquidity providers or borrowers within the Sui Ecosystem. Liquidity providers supply assets to the market, earning passive income through yields, while borrowers have the flexibility to obtain loans for different assets. |
| **Type** | Lending |
| **Auditors** | MoveBit |
| **Timeline** | May 24, 2023 – July 13, 2023 |
| **Languages** | Move |
| **Platform** | Sui |
| **Methods** | Architecture Review, Unit Testing, Manual Review |
| **Source Code** | https://github.com/naviprotocol/protocol-core |
| **Commits** | 838650c8471c5a6401022852222362e70dd455d1<br>1e888f75fc385657bef13d4991655d5b29455346<br>50b69ce6ed44b5b0da313a6689182992e63654b7<br>6f7fe8f3241fbed6ca41a11537704fe60cd0bbda |

### 1.2 Files in Scope

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

| ID | Files | SHA-1 Hash |
| --- | --- | --- |
| CAL | lending_core/sources/calculator.move | e8d7eeb688ea8fdac789d1250dab882dbb39182e |
| LEND | lending_core/sources/lending.move | 95c9c81332af01a7e4715f5e15fa1e9c12b2a737 |
| LGC | lending_core/sources/logic.move | 028bdc6cf13bd25f3904a7b7f9ce409c9abc86e5 |
| POOL | lending_core/sources/pool.move | aec00964424b2f1d4934e54fc5573ba20c98856f |
| STG | lending_core/sources/storage.move | 04e503beb4569e0b8f593a8283084b31756adeb8 |
| UTL | lending_core/sources/validation.move | 1b9cf17cd88bd9240a1bb20d5a37dcb69c93a3f3 |
| MTL | math/sources/math_utils.move | 62c86254e345ada776b87e684a4fadcf0eac2a46 |
| PTM | math/sources/percentage_math.move | 578ceb14800f6e5c033898702b9a7e36b1304205 |
| RMT | math/sources/ray_math.move | fd5d6b6024fddf0c9d82ba20a82f430c7fd7712a |
| SMT | math/sources/safe_math.move | dd3232c7d7b1117c9d106060d251ff180d162998 |
| ORC | oracle/sources/oracle.move | 0c12f8d4a1d704a12d63186581c26342baf7ca73 |
| UTL | utils/sources/utils.move | 06570ed2cb200337807337f3ea85db714172aa5f |

## 1.3 Issue Statistic

| Item | Count | Fixed | Acknowledged |
| --- | --- | --- | --- |
| **Total** | 16 | 12 | 4 |
| **Informational** | | | |

| | | | |
|---|---|---|---|
| **Minor** | 7 | 4 | 3 |
| **Medium** | 1 | | 1 |
| **Major** | 4 | 4 | |
| **Critical** | 4 | 4 | |

## 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 **Navi Protocol** to identify any potential issues and vulnerabilities in the source code of the **Navi** 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 **16** issues of varying severity, listed below.

| ID | Title | Severity | Status |
|---|---|---|---|
| POOL-01 | Missing Permission Validation | Critical | Fixed |
| POOL-02 | Missing State Validation | Major | Fixed |
| STG-03 | Function Visibility Error | Critical | Fixed |

| | | | |
|---|---|---|---|
| CAL-04 | Interest Rate Calculation Error | Critical | Fixed |
| ORC-05 | Oracle Lacks Update Cycle Verification | Major | Fixed |
| LEND-06 | Lacking Validation for the Generic Parameter CoinType | Major | Fixed |
| STG-07 | There are Unused Fields in the Structure | Minor | Acknowledged |
| POOL-08 | Centralization Risk | Medium | Acknowledged |
| LEND-09 | Withdrawal and Repayment Lack of Validation for Zero Amounts | Minor | Fixed |
| LEND-10 | Incorrect Data in Event | Minor | Fixed |
| VLD-11 | Unused Constants | Minor | Fixed |
| STG-12 | The Method for Obtaining `coin_decimals` is Incorrect | Minor | Fixed |
| LGC-13 | The Data is Not Updated Synchronously | Major | Fixed |
| LEND-14 | Numerical Precision Error | Critical | Fixed |
| ORC-15 | Lacking Events | Minor | Acknowledged |
| ORC-16 | The Price in the Oracle Lacks Validation | Minor | Acknowledged |

# 3 Participant Process

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

**Admin**

- Admin can withdraw treasury balance of protocol(not user's balance) through `withdraw_treasury()` .
- Admin or the account designated by the admin can manipulate the token price through `upda`

`te_token_price()` / `update_token_price_batch()` functions.

- Admin can incentive pool through `add_pool()`.
- Admin can upgrade the version through `version_migrate()`.
- Admin can add new reserve through `init_reserve()`.
- Admin can update the pause status through `set_pause()`.
- Admin can update some values in the `Storage` through `set_borrow_cap()` / `set_supply_cap()` / `set_borrow_cap()` / `set_ltv()` / `set_treasury_factor()` / `set_base_rate()` / `set_multiplier()` / `set_jump_rate_multiplier()` / `set_reserve_factor()` / `set_optimal_utilization()` / `set_liquidation_ratio()` / `set_liquidation_threshold()`.

**User**

- Users can add liquidity to the pool through the `deposit()` function.
- Users can withdraw their own assets from the pool through the `withdraw()` function.
- Users can borrow/repay assets from the pool through `borrow()` / `repay()` functions.
- Users can liquidate assets from the pool through the `liquidation_call()` function.

# 4 Findings

## POOL-01 Missing Permission Validation

**Severity: Critical**

**Status: Fixed**

**Code Location:** pool/sources/pool.move#L103

**Descriptions:** There is a lack of permission checking in the `withdraw()` function, allowing any user to withdraw a specified amount of tokens to a specified address without paying any cost, which directly leads to the loss of assets in the pool.

**Suggestion:** Add appropriate permission checks in the `withdraw()` function to ensure that only authorized users can execute the withdrawal operation. You can use access control modifiers (such as `friend`) to restrict access.

**Resolution:** The client has followed our suggestion and fixed the issue.

## POOL-02 Missing State Validation

**Severity: Major**

**Status: Fixed**

**Code Location:** pool/sources/pool.move

**Descriptions:** There is a lack of state validation of `Storage` in many public functions, which allows modifying data via some public functions even when the administrator has suspended transactions.

**Suggestion:** In all public functions, add appropriate state validation to ensure that the transaction has been lifted by the administrator before performing any modifying operations.

**Resolution:** The client has followed our suggestion and fixed the issue.

## STG-03 Function Visibility Error

**Severity: Critical**

**Status: Fixed**

**Code Location:** lending_core/sources/storage.move#L229, L236, L243, L249, L316, L431, L450, L469, L477, L485, L491; lending_core/sources/logic.move#L144, L179, L192, L199, L206, L213

**Descriptions:** The function `increase_supply_balance()` is a public function that allows the caller to modify the data in `Storage` and profit from it, causing the entire contract to fail to function properly. The same issue exists in many other parts of the code, which can be found based on the `Code Location` mentioned earlier. Also, the function `borrow_reserve_mut()` has a return value that is modifiable, so it is not recommended to set it as a public function.

**Suggestion:** Add appropriate permission checks in the `increase_supply_balance()` function to ensure that only authorized users can execute the withdrawal operation. You can use access control modifiers (such as `friend`) to restrict access.

**Resolution:** The client has followed our suggestion and fixed the issue.

## CAL-04 Interest Rate Calculation Error

**Severity: Critical**

**Status: Fixed**

**Code Location:** lending_core/sources/calculator.move#L6

**Descriptions:** The value of the constant `SECOND_PER_YEAR` is the number of seconds in a year, but in some places where it is used, it is incorrectly calculated with milliseconds, resulting in a value that is 1000 times larger than it should be, causing significant differences between the calculated results and the expected results.

**Suggestion:** Corrected the calculation of the constant `SECOND_PER_YEAR` to ensure its value is seconds per year rather than milliseconds. Set it to the correct value based on your needs and usage.

**Resolution:** The client has followed our suggestion and fixed the issue.

## ORC-05 Oracle Lacks Update Cycle Verification

**Severity: Major**

**Status: Fixed**

**Code Location:** oracle/sources/oracle.move

**Descriptions:** Oracle does not have validation for the maximum interval period when obtaining external prices. As external prices can change rapidly, there may be significant fluctuations in a short time period. When Oracle does not update prices for a long time, users may receive assets that do not match their expectations when performing operations such as borrowing, withdrawing, and settling.
**Resolution:** The client has followed our suggestion and fixed the issue.

## LEND-06 Lacking Validation for the Generic Parameter CoinType

**Severity: Major**

**Status: Fixed**

**Code Location:** lending_core/sources/lending.move

**Descriptions:** In the `lending` module, all functions lack validation of whether the generic parameter `CoinType` corresponds with the `coin_type` of the asset. If the parameter is passed incorrectly, it will cause incorrect asset calculation in `Storage` and the entire contract will have a significant vulnerability that will prevent it from functioning properly.

**Suggestion:** To ensure the `coin_type` consistency between the generic parameter `CoinType` and the parameter asset, add validation for both parameters.

**Resolution:** The client has followed our suggestion and fixed the issue.

## STG-07 There are Unused Fields in the Structure

**Severity: Minor**

**Status: Acknowledged**

**Code Location:** lending_core/sources/storage.move

**Descriptions:** The `users` field in struct `Storage` is only modified in the entire contract and is not used for any other logic. Similarly, the `is_isolated` field in struct `ReserveData` also has the same issue. If these fields are not needed, it is recommended to remove them.

**Suggestion:** Delete the unused fields.

**Resolution:** The client response reason is to prevent upgrade failures caused by adding fields later.

## POOL-08 Centralization Risk

**Severity: Medium**

**Status: Acknowledged**

**Code Location:** lending_core/sources/pool.move; oracle/sources/oracle.move; lending_core/sources/storage.move

**Descriptions:**

Centralization Risk is identified:

- Admin can withdraw treasury balance of protocol(not user's balance) through `withdraw_treasury()` .
- Admin or the account designated by the admin can manipulate the token price through `update_token_price()` / `update_token_price_batch()` functions.
- Admin can incentive pool through `add_pool()` .
- Admin can upgrade the version through `version_migrate()` .
- Admin can add new reserve through `init_reserve()` .
- Admin can update the pause status through `set_pause()` .

- Admin can update some values in the `Storage` through `set_borrow_cap()` / `set_supply_cap()` / `set_borrow_cap()` / `set_ltv()` / `set_treasury_factor()` / `set_base_rate()` / `set_multiplier()` / `set_jump_rate_multiplier()` / `set_reserve_factor()` / `set_optimal_utilization()` / `set_liquidation_ratio()` / `set_liquidation_threshold()` .

**Suggestion:** It is recommended to take some measures to mitigate the Centralization Risk.

**Resolution:** The client replied:

- We adopt mult-sig wallet to mitigate this issue.

Regarding `withdraw_treasury()` :

- This is the correct behavior. This method can only withdraw treasury funds, not users'.

Regarding Oracle:

- There was no reliable decentralized Oracle on Sui at the time of audit (Pyth and Supra were not mainnet ready), NAVI consulted Movebit about building our own oracle aggregator and aligned on this decision
- NAVI built an Oracle that aggregates price feed from Pyth, Binance, OKX and a total of 5 centralized exchanges to build a more robust pricing feed.

This has been fixed and addressed in NAVI Oracle design.

## LEND-09 Withdrawal and Repayment Lack of Validation for Zero Amounts

**Severity: Minor**

**Status: Fixed**

**Code Location:** lending_core/sources/lending.move#L83, L136

**Descriptions:** When using the `repay()` function to repay debts, if the current user does not have any debt, the current logic saves tokens to the pool and then takes them out of the pool. We believe this operation is meaningless and can cause more gas losses. We recommend using an `assert` function to validate and block this transaction. Similarly, when using the `withdraw()` function, validation for whether the withdrawable amount is zero is missing.

**Suggestion:** Add `assert` functions to prevent situations like this.

**Resolution:** The client has followed our suggestion and fixed the issue.

## LEND-10 Incorrect Data in Event

**Severity: Minor**

**Status: Fixed**

**Code Location:** lending_core/sources/lending.move#L108, L164, L209

**Descriptions:** In functions `withdraw()`, `repay()`, and `liquidation_call()`, the `amount` emitted in `emit` should represent the actual amount of funds withdrawn, repaid, or liquidated, rather than what the user inputs. Otherwise, it will send incorrect information to external event listeners.

**Suggestion:** Please check again and recalculate the amount.

**Resolution:** The client has followed our suggestion and fixed the issue.

## VLD-11 Unused Constants

**Severity: Minor**

**Status: Fixed**

**Code Location:** lending_core/sources/validation.move#L7, L8

lending_core/sources/storage.move#L118.

**Descriptions:** There are unused constants in the contract. It is recommended to remove them.

**Suggestion:** Remove unused constants in contracts.

**Resolution:** The client has followed our suggestion and fixed the issue.

## STG-12 The Method for Obtaining `coin_decimals` is Incorrect

**Severity: Minor**

**Status: Fixed**

**Code Location:** lending_core/sources/storage.move#L188

**Descriptions:** In the `init_reserve()` function, `coin_decimals` should not be passed in as a parameter, but should be obtained through the `CoinType`.

**Suggestion:** Get `coin_decimals` through the type parameter `CoinType`.

**Resolution:** The client has followed our suggestion and fixed the issue.

## LGC-13 The Data is Not Updated Synchronously

**Severity: Major**

**Status: Fixed**

**Code Location:** lending_core/sources/logic.move#L89

**Descriptions:** In the `execute_withdraw()` function, if the remaining amount after withdrawing is small, it will be saved in the treasury. However, the user's asset data and reserve balance in `ReserveData` are not updated, resulting in data desynchronization and affecting the calculation logic.

**Resolution:** The client has followed our suggestion and fixed the issue.

## LEND-14 Numerical Precision Error

**Severity: Critical**

**Status: Fixed**

**Code Location:** lending_core/sources/lending.move#L158

**Descriptions:** In function `repay()`, if there is an excess amount after repayment, it will be returned to the account through `pool::withdraw`. However, the returned value `excess_amount` is not converted to decimal precision through `pool::unnormal_amount`, which will result in an incorrect amount being returned to the user.

**Suggestion:** Convert the return value `excess_amount` to decimal precision by `pool::unnormal_amount`.

**Resolution:** The client has followed our suggestion and fixed the issue.

## ORC-15 Lacking Events

**Severity: Minor**

**Status: Acknowledged**

**Code Location:** lending_core/sources/oracle.move#L65, L83, L98, L129;

lending_core/sources/storage.move#L220, L227, L234, L241, L248, L255, L262, L269, L276, L283, L290, L297, L304

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

**Suggestion:** Add events for these functions.

## ORC-16 The Price in the Oracle Lacks Validation

**Severity: Minor**

**Status: Acknowledged**

**Code Location:** lending_core/sources/oracle.move#L77, L95, L117

**Descriptions:** When setting the price in the oracle, it is not validated whether the price is equal to 0. If the price is set to 0, then the value of the token will also be 0, making the current `ReserveData` meaningless.

**Suggestion:** Add assertion validation price must be greater than 0.

# 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.