Navi Protocol

# Audit Report

contact@bitslab.xyz

https://twitter.com/movebit_

Wed Jan 28 2026

# Navi Protocol Audit Report

# 1 Executive Summary

## 1.1 Project Information

| | |
|---|---|
| Description | Navi is a liquidity protocol, which supports borrow/lend top assets class in the Move ecosystem. The scope of this audit: Code that differs between commits bc86742307e6969ed14eb70219a69fe182f4488f and 2954d3b05aa8c2738375509d8895b967f28fd0fc. |
| Type | DeFi |
| Auditors | L1ght09, Leon@BitsLab |
| Timeline | Thu Jan 22 2026 - Sun Jan 25 2026 |
| Languages | Move |
| Platform | Sui |
| Methods | Architecture Review, Unit Testing, Manual Review |
| Source Code | https://github.com/naviprotocol/protocol/ |
| Commits | bc86742307e6969ed14eb70219a69fe182f4488f<br>2954d3b05aa8c2738375509d8895b967f28fd0fc<br>61d42926323b7a5479350a36ce1f21d7f545cf74 |

## 1.2 Files in Scope

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

| ID | File | SHA-1 Hash |
|---|---|---|
| ASW | oracle/sources/adaptor_swtichboard.move | ca36f7672c70e4b41efaa3d7f80227618c1d0abc |
| OPR | oracle/sources/oracle_provider.move | ce69b30f345837ba1a02e0933fe6461d4c1230e2 |
| OPR1 | oracle/sources/oracle_pro.move | 6571fab4bdbbe720edb0cc07652c68fc9e0502a7 |
| OMA | oracle/sources/oracle_manage.move | fb38adacee8945b2b5eec90a3efd8bc71855a862 |
| ACC | lending_core/sources/account.move | d8d073e01b62091d05ca12e6e89fada94d3b511f |
| DCA | lending_core/sources/dynamic_calculator.move | 5a71254230336d7ce047a35e6b1a80c78cae7d54 |
| LEN | lending_core/sources/lending.move | 5bb7a4490b84c2bb9bf44535cc577dfa5682faf4 |
| IV31 | lending_core/sources/incentive_v3.move | c8d04f414cd500d7de86a032dbe73252dfb818ed |
| LOG | lending_core/sources/logic.move | 56ce0c7145a32a76ed56fe76a223a25802bc401a |
| POO | lending_core/sources/pool.move | 12eb60747a80576f824bdc729567f048e22396ca |

| | | |
|---|---|---|
| STO | lending_core/sources/storage.move | 640026933fe00c2c815b8b6e33e8b1292c9794b5 |
| FLO | lending_core/sources/flash_loan.move | 9a01f7e772b0c8751093815e272f4cc886f55ec1 |
| MAN1 | lending_core/sources/manage.move | 60cee26ff08d0a05f85b42f29da070ec75b70bf3 |
| PMA | lending_core/sources/pool_manager.move | bd4f55c7fd4c940788e34edd6f6e006a7b750e92 |

## 1.3 Issue Statistic

| Item | Count | Fixed | Acknowledged |
|---|---|---|---|
| Total | 5 | 3 | 2 |
| Critical | 0 | 0 | 0 |
| Major | 0 | 0 | 0 |
| Medium | 1 | 1 | 0 |
| Minor | 1 | 1 | 0 |
| Informational | 3 | 1 | 2 |

## 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 Navi Protocol to identify any potential issues and vulnerabilities in the source code of the Navi Protocol 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 5 issues of varying severity, listed below.

| ID | Title | Severity | Status |
|---|---|---|---|
| ASW-5 | Redundant Code in `get_price_native()` | Informational | Acknowledged |
| LOG-3 | Missing E-Mode Compatibility in `calculate_avg_threshold()` | Minor | Fixed |
| STO-1 | Lack of Event Emit | Informational | Fixed |
| STO-4 | Missing State Check in `set_emode_config_active()` | Informational | Acknowledged |
| STO-6 | Missing Validation in `withdraw_treasury_v2()` | Medium | Fixed |

# 3 Participant Process

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

**Admin**

- Admin can create flash loan config through the `create_flash_loan_config_with_storage()` function.
- Admin can create flash loan asset through the `create_flash_loan_asset()` function.
- Admin can initialize a reserve through the `init_reserve()` function.
- Admin can set pause status through the `set_pause()` function.
- Admin can withdraw treasury through the `withdraw_treasury_v2()` function.
- Admin can initialize emode for main market through the `init_emode_for_main_market()` function.
- Admin can initialize for main market through the `init_for_main_market()` function.
- Admin can initialize borrow weight for main market through the `init_borrow_weight_for_main_market()` function.
- Admin can create emode asset through the `create_emode_asset()` function.
- Admin can create emode pair through the `create_emode_pair()` function.
- Admin can set emode config active through the `set_emode_config_active()` function.
- Admin can set emode asset liquidation threshold through the `set_emode_asset_lt()` function.
- Admin can set emode asset loan to value through the `set_emode_asset_ltv()` function.
- Admin can set emode asset liquidation bonus through the `set_emode_asset_liquidation_bonus()` function.
- Admin can create new market through the `create_new_market()` function.
- Admin can set borrow weight through the `set_borrow_weight()` function.

- Admin can remove borrow weight through the `remove_borrow_weight()` function.
- Admin can create Switchboard oracle provider config through the `create_switchboard_oracle_provider_config()` function.
- Admin can set Switchboard price source pair ID through the `set_switchboard_price_source_pair_id()` function.
- Admin can enable Switchboard oracle provider through the `enable_switchboard_oracle_provider()` function.
- Admin can disable Switchboard oracle provider through the `disable_switchboard_oracle_provider()` function.

**IncentiveOwner**

- IncentiveOwner can create incentive V3 reward fund through the `create_incentive_v3_reward_fund_with_storage()` function.
- IncentiveOwner can create incentive V3 through the `create_incentive_v3_with_storage()` function.

**User**

- User can set an account name through the `set_account_name()` function.
- User can set an account description through the `set_account_description()` function.
- User can set the last update time for his account through the `set_last_update_time()` function.
- User can set the market balance for his account through the `set_market_balance()` function.
- User can enter emode through `enter_emode()` and `enter_emode_with_account_cap()` functions.
- User can exit emode through `exit_emode()` and `exit_emode_with_account_cap()` functions.
- User can update his state through the `update_state_of_user()` function.
- User can update all states through the `update_state_of_all()` function.

- User can update the reward state by asset through the `update_reward_state_by_asset()` function.
- User can update single price through the `update_single_price_v2()` function.

# 4 Findings

## ASW-5 Redundant Code in `get_price_native()`

Severity: Informational

Status: Acknowledged

Code Location:

oracle/sources/adaptor_swtichboard.move#9,12

Descriptions:

The `get_price_native()` function in the adapter switch contract is used to fetch the latest price data from the oracle aggregator. This function fetches the `current_result` data twice. First, it fetches `current_result` to get the timestamp. Then, it calls `aggregator::current_result(aggregator)` again to get the price and index results. This results in code redundancy.

Suggestion:

Please consider whether this code conforms to project design. If not, it is recommended to fetch `current_result` only once.

## LOG-3 Missing E-Mode Compatibility in `calculate_avg_threshold()`

Severity: Minor

Status: Fixed

Code Location:

lending_core/sources/logic.move#729

Descriptions:

The `calculate_avg_threshold()` function is used to compute the average liquidation threshold for a user's collateral assets. This average threshold is important for determining the overall health and risk level of a user's position. However, the function does not consider the user's E-Mode status. In E-Mode, different liquidation thresholds may apply to specific assets. The function currently uses the standard liquidation factors for all assets, ignoring any E-Mode-specific thresholds. This can lead to an incorrect average threshold calculation.

Suggestion:

It is recommended to update the `calculate_avg_threshold()` function to check the user's E-Mode status and use the appropriate liquidation threshold (either standard or E-Mode-specific) for each collateral asset during the calculation.

Resolution:

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

## STO-1 Lack of Event Emit

Severity: Informational

Status: Fixed

Code Location:

lending_core/sources/storage.move#913

Descriptions:

The `set_emode_config_active()` function lacks event logging, which is essential for blockchain transparency, off-chain data tracking, and frontend integration. Event logs allow external systems to monitor contract activities without querying the blockchain state directly.

Suggestion:

It is recommended to add event emission for these operations.

Resolution:

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

## STO-4 Missing State Check in `set_emode_config_active()`

Severity: Informational

Status: Acknowledged

Code Location:

lending_core/sources/storage.move#910

Descriptions:

The `set_emode_config_active()` function in the `storage.move` contract is responsible for updating the active status ( `isActive` ) of a specific `e-mode` configuration. This function directly sets the `isActive` field to the provided boolean value without first checking its current state. The problem is that it does not verify whether the new `is_active` value is different from the current `isActive` value. This oversight means the function can be called to set the status to the same value it already has. This can lead to unnecessary state changes being recorded on the blockchain, wasting gas fees for the transaction caller, and potentially causing confusion in off-chain monitoring tools that track state changes.

Suggestion:

It is recommended to add a check at the beginning of the function to compare the new `is_active` value with the current `isActive` value of the configuration. The function should proceed with the update only if the values are different, otherwise it should exit early to prevent an unnecessary state change.

## STO-6 Missing Validation in `withdraw_treasury_v2()`

Severity: Medium

Status: Fixed

Code Location:

lending_core/sources/storage.move#724

Descriptions:

The `withdraw_treasury_v2()` function is responsible for withdrawing assets from the treasury. It checks the coin type but does not verify if the market ID from the storage matches the market ID from the pool. This missing check means the function might process withdrawals using incorrect or mismatched market data. As a result, assets could be withdrawn from the wrong treasury pool, leading to incorrect accounting and potential loss of funds.

Suggestion:

It is recommended to add a validation step to ensure the market IDs from the storage and the pool are identical before proceeding with the withdrawal.

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