Lista DAO

Security Review

Cantina Managed review by:
Slowfi, Lead Security Researcher

April 10, 2026

Contents

1 Introduction **2**
- 1.1 About Cantina . . . 2
- 1.2 Disclaimer . . . 2
- 1.3 Risk assessment . . . 2
  - 1.3.1 Severity Classification . . . 2

2 Security Review Summary **3**
- 2.1 Scope . . . 3

3 Findings **4**
- 3.1 Medium Risk . . . 4
  - 3.1.1 Stale exact asset repay can grief position migrations . . . 4
  - 3.1.2 Unconditional native repayment can break integration . . . 4
- 3.2 Gas Optimization . . . 4
  - 3.2.1 Revert strings can be replaced with custom errors . . . 4
- 3.3 Informational . . . 5
  - 3.3.1 Delegated borrow follows the existing authorization model . . . 5
  - 3.3.2 Allowed partial migration . . . 5
  - 3.3.3 Unsolicited native `BNB` lacks recovery path . . . 5
  - 3.3.4 Migration could support a deadline or APR bounds . . . 6

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

A security review is a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While the review endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that a security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity level	Impact: High	Impact: Medium	Impact: Low
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings are a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

Lista DAO functions as the open-source decentralized stablecoin lending protocol powered by LSDfi.

From Mar 26th to Mar 27th the security researchers conducted a review of moolah on commit hash dde336dd. A total of **7** issues were identified:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	2	2	0
Low Risk	0	0	0
Gas Optimizations	1	0	1
Informational	4	1	3
Total	**7**	**3**	**4**

2.1 Scope

The security review had the following components in scope for moolah on commit hash dde336dd:

```
src
├── broker/LendingBroker.sol
└── utils/PositionManager.sol
```

3 Findings

3.1 Medium Risk

3.1.1 Stale exact asset repay can grief position migrations

Severity: Medium Risk

Context: PositionManager.sol#L71-L97

Description: The function `onMoolahFlashLoan` from contract `PositionManager` repays the source market with `MOOLAH.repay(p.outMarket, assets, 0, p.user, "")`, where `assets` is the flash loaned `borrowAmount` supplied to `migrateCommonMarketToFixedTermMarket`. This uses Moolah's exact asset repay mode, while `IMoolah.repay` explicitly warns that a small front run repay can make that flow revert for underflow.

As a result, a third party can repay a small amount of the user's source debt before execution and cause a full migration that targets the previous debt amount to revert. This permissionlessly blocks the migration until the user resubmits with an updated amount.

Recommendation: Consider repaying the source position in shares mode when clearing the migrated debt, or otherwise derive the repay amount from the current position state at execution time instead of relying on a stale caller-supplied exact asset amount.

Lista DAO: Fixed in PR 152.

Slowfi: Fix verified.

3.1.2 Unconditional native repayment can break integration

Severity: Medium Risk

Context: LendingBroker.sol#L220-L292

Description: The function `_transferLoanToken` from contract `LendingBroker` unwraps WBNB and sends native BNB whenever a WBNB provider is registered for the market. Both user-facing `borrow(uint256)` and `borrow(uint256,uint256)` route through this helper, so contract users that do not implement a payable `receive` or fallback function revert on borrow instead of receiving the borrowed asset. The same native send path is also used when refunding excess value in both `repay` functions, which means an otherwise valid repay can revert when any surplus must be returned to a non-payable contract caller.

This changes the standard WBNB broker flow from ERC20 transfers to mandatory native transfers and blocks affected contract integrations.

Recommendation: Consider preserving WBNB ERC20 transfers on the standard borrow and refund paths, and reserve native BNB delivery for an explicit opt-in flow. If native refunds remain supported, consider avoiding unconditional native sends to arbitrary callers and fall back to returning WBNB when the recipient is not expected to receive native BNB.

Lista DAO: Fixed in PR 152.

Slowfi: Fix verified.

3.2 Gas Optimization

3.2.1 Revert strings can be replaced with custom errors

Severity: Gas Optimization

Context: *(No context files were provided by the reviewer)*

Description: The function `migrateCommonMarketToFixedTermMarket` from contract `PositionManager` and multiple functions from contract `LendingBroker` still rely on string-based `require` and `revert` messages for input validation and state checks. This pattern is used repeatedly across both contracts for common conditions such as zero address checks, authorization checks, invalid market state, and unsupported operations.

While this is not a functional issue, replacing repeated revert strings with custom errors would reduce deployed bytecode size and lower revert-path gas costs, while also making shared failure cases easier to standardize across the codebase.

Recommendation: Consider replacing repeated revert strings with custom errors, especially for shared validation paths that appear multiple times across `PositionManager` and `LendingBroker`. Consider grouping common errors in a shared error definition file if the same conditions are used across contracts.

Lista DAO: Acknowledged.

Slowfi: Acknowledged.

3.3 Informational

3.3.1 Delegated borrow follows the existing authorization model

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: The function `borrow` from contract `LendingBroker` allows an authorized caller to open a fixed-term borrow position for `user` and send the borrowed tokens to an arbitrary `receiver`, gated by `MOOLAH.isAuthorized(user, msg.sender)`. This does not introduce a new privilege boundary by itself, because `IMoolah.isAuthorized` is already defined as an approval to modify the author's positions on all markets.

The behavior is therefore consistent with the existing trust model, but it is worth documenting because integrations may otherwise interpret the new delegated borrow entrypoint as a narrower approval than the underlying Moolah authorization actually provides.

Recommendation: Consider documenting that delegated fixed-term borrowing inherits Moolah's existing all markets authorization scope, and consider making this trust assumption explicit in user-facing documentation and integration guides.

Lista DAO: Acknowledged.

Slowfi: Acknowledged.

3.3.2 Allowed partial migration

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: The function `migrateCommonMarketToFixedTermMarket` from contract `PositionManager` accepts caller-controlled `collateralAmount` and `borrowAmount`, and the interface and tests already exercise migrations that move only part of the source position. This means the flow is not limited to full migrations: it can intentionally leave residual debt and collateral in the source market as long as the remaining position stays valid after the migration.

This is an informational behavior note rather than a vulnerability, but it is important to document so users and integrators do not assume the function always closes the source position completely.

Recommendation: Consider documenting that the migration flow supports both full and partial position moves, and consider describing the expected post-migration health requirements for any residual source position.

Lista DAO: Acknowledged.

Slowfi: Acknowledged.

3.3.3 Unsolicited native `BNB` lacks recovery path

Severity: Informational

Context: LendingBroker.sol#L489

Description: The function `receive` from contract `LendingBroker` accepts arbitrary native BNB. This is needed for the WBNB unwrap flow, but it also allows accidental transfers and forced native sends to

leave a native balance on the broker. The current implementation does not provide a native rescue or sweep function, so unsolicited native BNB cannot be recovered within the current contract version and may remain stranded until an upgrade is performed.

The impact is limited to operational dust or mistakenly sent funds.

Recommendation: Consider adding a manager-only native recovery function for unsolicited balances, and consider documenting that direct native transfers to the broker are unsupported outside the intended WBNB unwrap flow.

Lista DAO: Fixed in PR 152.

Slowfi: Fix verified.

3.3.4 Migration could support a deadline or APR bounds

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: The function `migrateCommonMarketToFixedTermMarket` from contract `PositionManager` selects the destination product only by `termId`, while the corresponding fixed-term APR can be updated by the broker bot before execution. As a result, a delayed migration transaction can still execute against the same `termId` after its APR has changed, without any caller specified deadline or maximum acceptable APR.

This is a UX hardening note rather than a security issue, but it can lead to a less favorable fixed-term rate than the user expected when the transaction was submitted.

Recommendation: Consider adding a deadline and/or maximum acceptable APR to the migration parameters, and consider validating the current term data against those bounds before starting the flash loan.

Lista DAO: Acknowledged.

Slowfi: Acknowledged.