Lista DAO Contracts

Security Review

Cantina Managed review by:

0xWeiss, Security Researcher
Víctor Martínez, Security Researcher

September 17, 2025

Contents

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

Cantina Managed provides 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 Cantina Managed 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 the Cantina Managed 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 Sep 1st to Sep 8th the Cantina team conducted a review of lista-dao-contracts on commit hash 50b1e7b1. The team identified a total of **13** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	3	3	0
Medium Risk	3	1	2
Low Risk	5	4	1
Gas Optimizations	0	0	0
Informational	2	2	0
Total	**13**	**10**	**3**

3 Findings

3.1 High Risk

3.1.1 Bypass of `nonReentrant` in provide via Direct PCS V3 Position NFT Transfer

Severity: High Risk

Context: PancakeSwapV3LpProvider.sol#572-577

Description: The `nonReentrant` modifier on the provide function can be bypassed by directly transferring a PCS V3 Position NFT to the Provider. This action triggers the ERC721 callback, which does not enforce the `nonReentrant` modifier, leaving the contract exposed to potential reentrancy.

Recommendation: Simplify the deposit process by restricting deposits to the `provide` function, removing support for direct NFT safeTransfers.

Lista DAO: Resolved in commit 86976aee. Users can now deposit exclusively via the `provide` function.

Cantina Managed: Fixed. Direct NFT transfers for deposits are no longer allowed. The new check in `onERC721Received` ensures that NFT transfer operators are limited to either `PancakeStakingHub` or the provider itself.

3.1.2 Withdrawals and harvests fail when `MasterChefV3` has insufficient CAKE

Severity: High Risk

Context: PancakeSwapV3LpStakingHub.sol#160, PancakeSwapV3LpStakingHub.sol#233, MasterChefV3.sol#L812-L829

Description: The current `require(postBalance == preBalance + rewards, "PancakeSwapStakingHub: invalid-reward-balance")` statement on `PancakeSwapV3LpStakingHub::withdraw` and `PancakeSwapV3LpStakingHub::harvest` can cause a denial-of-service (DoS) when MasterChefV3 does not have enough CAKE balance.

`MasterChef::_safeTransfer` caps transfers to the available balance but still returns the originally calculated reward, which causes the `require` check to fail.

Recommendation: Use the actual balance difference (amount successfully transferred) instead of the returned reward value in the `require` check to prevent DoS issues.

Lista DAO: Fixed in commit edda1fdf.

Cantina Managed: Fix verified.

3.1.3 `getLpValue` misapplies decimal scaling for tokens with decimals other than 18

Severity: High Risk

Context: PancakeSwapV3LpProvider.sol#L758-L757

Description: The `getAmounts` function normalizes `amount0` and `amount1` into 18 decimals, regardless of the underlying ERC20 token decimals.

However, `getTokenPrice` further scales the return value by `10 (18 - tokenDecimals)` using the token's decimals as a scaling factor. Since the amounts have already been normalized to 18 decimals, this introduces a double scaling issue** when values are used in `getLpValue`.

If `token0` has 6 decimals:

- `getAmounts` scales `amount0` up by `10 ** 12` to reach 18 decimals.
- `getTokenPrice` multiplies the oracle's 8-decimal price by `10 (18 - 6) = 10 12`.
- When multiplied together, the effective scaling is `10^24`, causing nflated LP valuations.

Impact: High. `getLpValue` will return severely inflated or deflated values, for tokens with decimals other than 18.

Recommendation: Do not use 18 decimal normalised prices in `getLpValue`, since token amounts are already normalised to 18 decimals.

Lista DAO: Fixed in commit 75a12577.

Cantina Managed: The fix has been verified, prices are now taken directly with 8-decimal precision.

3.2 Medium Risk

3.2.1 Token price scaling mechanics will not work for tokens with high decimals

Severity: Medium Risk

Context: PcsV3LpNumbersHelper.sol#142

Description: The current method of scaling token prices normalizes both to 18 decimals. This fails for tokens with more than 18 decimals, unlike the original GUniLP implementation, which scales based on `token0.decimals() * token1.decimals()`. That approach correctly handles tokens exceeding 18 decimals. If ListaDAO plans to support (or may later support) collateral tokens with more than 18 decimals, the current implementation will break for those cases.

Recommendation: Adopt the scaling logic from GUniLP that uses the product of *token0.decimals()* and *token1.decimals()*, ensuring compatibility with tokens beyond 18 decimals. Otherwise, ensure that LPs from pools containing such assets are not enabled as collateral.

Lista DAO: Acknowledged. The decimal places of all LP's token should less than or equals to 18 in our case.

Cantina Managed: The issue has been acknowledged, the protocol will follow the recommendation of not using over 18 decimals tokens as collateral.

3.2.2 LP NFTs deposited during emergency mode cannot earn rewards after normal operations resume

Severity: Medium Risk

Context: PancakeSwapV3LpStakingHub.sol#128, MasterChefV3.sol#L245-L249

Description: If a deposit is made while the MasterChefV3 contract is in emergency mode, the LP NFT remains in the staking hub. Once emergency mode is disabled, that NFT cannot automatically be restaked into MasterChef to start earning rewards again.

Recommendation: Implement a `restake` function that allows idle NFTs in the hub to be transferred back into the MasterChef contract once normal operations resume. This ensures users can resume earning rewards after emergency mode deactivation.

Lista DAO: Fixed in commit f12da920.

Cantina Managed: The fix has been verified. A permissioned manager-only `restake` function has been introduced to re-stake idle NFTs once emergency mode is lifted.

3.2.3 Liquidation DoS risk from deny-list tokens

Severity: Medium Risk

Context: PcsV3LpLiquidationHelper.sol#L124-L119

Description: The `postLiquidation` function directly transfers leftover tokens to the liquidated user without considering whether the recipient can receive them. For tokens with deny/blacklist functionality (e.g., USDC, USDT), this call will revert if the recipient is blacklisted. As a result, the entire liquidation transaction fails.

Impact: High. Liquidations involving deny-list tokens can be blocked entirely if the user is blacklisted. This denial-of-service risk may lead to protocol insolvency from accumulating bad debt, while also creates incentivizing malicious users to intentionally get blacklisted in order to avoid liquidation.

Recommendation: Implement a pull-based system where leftover tokens are credited to user accounts instead of directly transferred, allowing users to claim when eligible while ensuring liquidations always complete successfully.

Lista DAO: This scenario is covered by our monitoring and operational runbooks. The security team handles alerts in real time, pauses liquidation if needed, and escalates cases for manual review before deciding next steps.

Cantina Managed: The issue has been acknowledged. The project team confirmed they will handle occurrences using the process described above.

3.3 Low Risk

3.3.1 Stale `LP_USD` syncing on deposit

Severity: Low Risk

Context: PancakeSwapV3LpProvider.sol#656

Description: On deposit, when a user holds multiple LPs, `_syncUserCdpPosition` may operate on stale data. The newly deposited LP is updated via `_syncLpValue(tokenId)`, but the other LPs rely on cached values in `_userLpTotalValue`. This can result in incorrect syncing of `LP_USD`. Given this behavior, `syncLpPrice == true` should be enforced to ensure all LP values are refreshed during deposit.

Recommendation: Enforce `syncLpPrice == true` in `_syncUserCdpPosition` during deposit to guarantee accurate position syncing for users with multiple LPs.

Lista DAO: We have decided not to address this within the contract because we rely on an off-chain bot that monitors LP values at one-minute intervals. The bot compares each LP's value to its corresponding `LP_USD` in the CDP, and if any LP deviates by more than 3% (configurable), it will synchronize all LPs at once. This approach ensures that `LP_USD` remains accurate without requiring on-chain enforcement.

Cantina Managed: The issue has been acknowledged, the protocol uses an off-chain bot to maintain LPs synchronised.

3.3.2 Wrong rounding direction causes protocol fees to be underestimated

Severity: Low Risk

Context: PancakeSwapV3LpStakingVault.sol#169, PancakeSwapV3LpStakingVault.sol#121

Description: The current rounding direction to calculate rewards fee results in fee being rounded down. In most DeFi protocols, protocol fees are typically rounded up to prevent revenue loss.

Recommendation: Adjust the rounding logic so that protocol fees are always rounded up.

Lista DAO: Fixed in commit 3ad794b7.

Cantina Managed: Fix verified.

3.3.3 Missing max `feeRate` check

Severity: Low Risk

Context: PancakeSwapV3LpStakingVault.sol#L227

Description: There are two functions that update the liquidity provider fee rate. `setLpProviderFeeRate` and `registerLpProvider`. Only `setLpProviderFeeRate` validates the fee rate not to exceed 10000 bps:

```
function setLpProviderFeeRate(address provider, uint256 feeRate) external onlyRole(MANAGER) {
    require(provider != address(0) && lpProviders[provider], "PancakeSwapLpStakingVault:
    ↪  provider-not-registered");
    uint256 oldFeeRate = feeRates[provider];
    require(feeRate != oldFeeRate && feeRate <= DENOMINATOR, "PancakeSwapLpStakingVault: invalid-fee-rate");
    lpProviders[provider] = true;
    feeRates[provider] = feeRate;
    emit FeeRateUpdated(provider, oldFeeRate, feeRate);
  }
```

Recommendation: Add a check so that the feeRate value is also validated when registering the LP:

```
  function registerLpProvider(address provider, uint256 feeRate) external onlyRole(MANAGER) {
     require(
       provider != address(0) &&
```

```
      !lpProviders[provider],
      "PancakeSwapLpStakingVault: provider-already-registered"
    );
+   require(feeRate <= DENOMINATOR, "PancakeSwapLpStakingVault: invalid-fee-rate");
    lpProviders[provider] = true;
    feeRates[provider] = feeRate;
```

Lista DAO: Fixed in commit bd53ceeb.

Cantina Managed: Fix verified.

3.3.4 Missing token validation

Severity: Low Risk

Context: PancakeSwapV3LpProvider.sol#L135

Description: On the initialization of the PancakeSwapV3LpProvider contract, the token addresses are not validated properly. According to the pool creation in pancake swap v3, the tokens should be in a specific order depending on their addresses. This is not enforced currently in the code and could cause DoS issues down the line if they were to be initialized in the opposite way.

Recommendation: Consider adding the same validation as the `PancakeSwap` factory:

```
require(tokenA != tokenB);
    (address token0, address token1) = tokenA < tokenB ? (tokenA, tokenB) : (tokenB, tokenA);
    require(token0 != address(0));
```

Lista DAO: Fixed in commit 00b8b5c5.

Cantina Managed: Fix verified.s

3.3.5 Inconsistent ownership validation for direct ERC721 transfers

Severity: Low Risk

Context: PancakeSwapV3LpProvider.sol#L213

Description: The ownership check in `provide` does not protect against direct transfers. When a PCS V3 Position NFT is sent directly, it triggers onERC721Received, which currently lacks equivalent validation. This creates an inconsistency between deposit methods and could allow unintended deposits to bypass the intended ownership check.

Recommendation: Move the ownership validation into onERC721Received and enforce `operator == from || operator == address(this)` to ensure consistent checks across both `provide` and direct transfer deposit paths.

Lista DAO: Fixed in commit 5cfff7a2.

Cantina Managed: Fixed. A check `require(from == pancakeStakingHub || operator == address(this), "PcsV3LpProvider: use-provide()")` has been added to the `onERC721Received` function in order to only accept NFTs sent from PancakeSwapV3LpStakingHub or the provider contract itself through the `provide` function.

3.4 Informational

3.4.1 Overly complicated rewards calculation function can be simplified

Severity: Informational

Context: PancakeSwapV3LpProvider.sol#L598, PancakeSwapV3LpStakingVault.sol#L115

Description: The `_sendRewardAfterFeeCut` function:

- `safeIncreaseAllowance` to the `pancakeLpStakingVault` contract.
- Calls `feeCut` which transfers the entire reward amount to the contract.
- It calculates the fees for the manager.

- It sends back to the lp provider the entire reward minus the fee for the manager.

All of this could be refactored to:

- Calls `feeCut` which calculates the fees for the manager.
- It then sends only the fees for the manager after calcualtion.

Recommendation: Apply the following:

- `_sendRewardAfterFeeCut` function refactor:

```
    function _sendRewardAfterFeeCut(uint256 amount, address to) internal {
        require(to != address(0), "PcsV3LpProvider: invalid-recipient");
        if (amount > 0) {
-           // approve rewardToken to the vault
-           IERC20(rewardToken).safeIncreaseAllowance(pancakeLpStakingVault, amount);
           // transfer rewards to the vault
-           uint256 rewardAfterCut = IPancakeSwapV3LpStakingVault(pancakeLpStakingVault).feeCut(amount);
+           (uint256 rewardAfterCut, uint256 fee) =
↪   IPancakeSwapV3LpStakingVault(pancakeLpStakingVault).feeCut(amount);
+          //transfer fees to the staking vault
+          IERC20(rewardToken).safeTransfer(pancakeLpStakingVault, rewardAfterCut);
           // transfer rewards to the user
           IERC20(rewardToken).safeTransfer(to, rewardAfterCut);
        }
    }
```

- `feeCut` function refactor:

```
    function feeCut(uint256 amount) override external onlyLpProvider whenNotPaused nonReentrant returns
↪   (uint256) {
      require(amount > 0, "PancakeSwapLpStakingVault: zero-amount-provided");
-     IERC20(rewardToken).safeTransferFrom(msg.sender, address(this), amount);
      // cut fee
      uint256 feeRate = feeRates[msg.sender];
      if (feeRate > 0) {
        uint256 fee = FullMath.mulDiv(amount, feeRate, DENOMINATOR);
        availableFees += fee;
        amount -= fee;
      }
-     // transfer remaining amount to lpProxy
-     IERC20(rewardToken).safeTransfer(msg.sender, amount);
      // emit Fee cut event (provider address, amount, fee rate)
      emit FeeCut(msg.sender, amount, feeRate);
-     return amount;
+    return amount, fee;
    }
```

Lista DAO: Fixed in commit 7ddabe85.

Cantina Managed: Fix verified.

3.4.2 Minor improvements to code and comments

Severity: Informational

Context: *(See each case below)*

- BaseTokenProvider.sol#L275, mBTCProvider.sol#L163, PumpBTCProvider.sol#L163 - The `onlyRole(PROXY)` modifier is redundant in the external liquidation function since access control is already enforced in the internal call to the primary liquidation function.
- PancakeSwapV3LpProvider.sol#L326 - The comment incorrectly states LP_USD is burned, but the function only updates liquidation state and emits an event.
- PancakeSwapV3LpProvider.sol#L340 - Typos: should read 'pays the debt' (not 'and paid the debt'), 'transformed' (not 'transform'), and 'transferred' (not 'transfer').
- PancakeSwapV3LpProvider.sol#L366, PancakeSwapV3LpProvider.sol#L493 - Comment incorrectly states that 3 variables are encoded in 'data'; it should mention 4, including the missing 'deadline' parameter.

- PancakeSwapV3LpStakingVault.sol#L217 - `setLpProviderFeeRate` redundantly sets `lpProviders[provider] = true`, even though the provider is already required to be registered.
- PancakeSwapV3LpProvider.sol#L468 - `userTotalLpValue[owner] = 0` is unnecessarily set inside the loop and should be moved outside to avoid redundant writes.
- PancakeSwapV3LpStakingHub.sol#L45, PancakeSwapV3LpStakingHub.sol#L48 - TODOs left in the code for emergencyMode and tokenIds that should be addressed or removed.

Lista DAO:

- 1 - We have deployed the `mBTCProvider` and PumpBTCProvider already, we decided not to change this at the moment.
- 2 to 5 - Fixed in commit d0acc8b7.
- 6 - Fixed in commits b5e2df3 and 1dc4fec.
- 7 - Fixed in commit afc96c60.

Cantina Managed: Pending.

- 1 - The issue has been acknowledged.
- 2 to 5 - Verified fixed in commit `d0acc8b`.
- 6 - Confirmed as fixed through the architectural change from push-based to pull-based leftover fund distribution.
- 7 - Fixed in commit `afc96c60`.