YieldNest clisBNB Strategy Audit Report

May 2025

Prepared for yieldnest.finance by NFR Audits
nofrontrunning@gmail.com

NFR AUDITS

Scope

PR https://github.com/yieldnest/yieldnest-clisBNB-strategy/pull/1 Commit 71026b08b4b2016cbc8bde99f80f974a97e68511

- ~~src/ClisBnbStrategy.sol
- ~~src/module/ClisBnbStrategyRateProvider.sol
- ~~lib/yieldnest-vault/src/strategy/BaseStrategy.sol
- ~~src/interfaces/ISlisBnbProvider.sol
- ~~src/interfaces/Interaction.sol

Codebase Overview

This strategy takes Lista's slisBNB in as base asset, and deposit it to Lista CDP which will mint un-transferrable clisBNB back to the receiving address (set to YN's MPC wallet address). The base asset is slisBNB, 18 decimals, no default asset.

Assets

Lista is a protocol offers Defi products around BNB like liquid staking, lending markets, CDP

- slisBNB - Lista's liquid staked BNB, interest bearing asset
- clisBNB - Lista's CDP receipt token, pegged 1:1 to BNB, non-transferrable. designed to allow slisBNB as collateral and borrow stable lisUSD, for looping etc. Also MPC wallet holder of slisBNB will have access to Binance launchpool - exclusive new token launch, earnings etc.

The codebase is well-written, documented, and organised. The code structure is solid, and the quality is good. We recommend fixing the issues identified in this report before deployment. It will also be beneficial to set up off-chain monitoring of the system after deployment, especially for critical events and state updates. Further audits are highly recommended if new updates are made to the codebase in the future.

Auditing Methods

The codebase is manually audited line by line. Auto detectors and static analysers are also used to ensure best coverage of vulnerabilities and issue validation.

Severity Classification

Severity classification in this report uses Immunefi Vulnerability Severity Classification System - v2.3 for smart contracts.

Issues

C - Critical
H - High
M - Medium
L - Low
N - Notes

M01 Inconsistent design logic

According to the specs, the clisBNB strategy is designed to only support `slisBNB` as depositing asset for now, with future potential to open up for other assets. However the codebase has inconsistent implementation of logic regarding this, for example

- while the `_deposit()` function allows potential depositing assets other than `slisBNB` if they are added as supported assets, the `_withdrawAsset()` function strictly reverts in line 201 if the asset is not slisBNB when the strategy needs to unstake `slisBNB`.
- The `_withdrawAsset()` function allows withdraw of any assets (if more assets are supported) when the strategy has enough balance of them, but only allows unstaking `slisBNB` and all other potential assets are not supported.

The current design does not pose immediate threat because the strategy only supports `slisBNB` as deposit, but this could change if more assets are added in the future.

Consider making the logic clear on policies regarding extra assets. If the intention is to support more assets in the future, consider changing the logic in `_withdrawAsset`, put the release logic in an if clause for `IERC20(asset_) == slisBnb()`. If the intention is to revert for all other assets for now, consider reverting at the beginning of `_withdrawAsset` function to fail early, also add the same revert condition to `_deposit()` function.

Status update: Fixed in commit 42b137.

L01 Semantic overload for function return values

The `_maxWithdrawAsset()` function is designed to return the smaller value of the `availableAssets` and `maxAssets`. It also returns 0 if the vault is paused or the given `asset_` is not withdrawable. This is a case of semantic overloading on the result of this function where value 0 could mean there is 0 asset available for withdraw or it could mean the vault is paused/given asset is not withdrawable, which makes the return value hard to deal with.

Similar case can also be found in function `_maxRedeemAsset()`.

Consider using an err instead if the vault is paused/given asset is not withdrawable.

Status update: Acknowledged, won't fix. Client stated - Design decision to make maxRedeem not revert. If contract is paused or asset is not withdrawable then how much you can withdraw or redeem is 0. While the standard doesn't specify this with precision, our design choice is to have this behaviour.

N01 Unused named return variables

Named return variables are a way to declare variables that are meant to be used within a function body for the purpose of being returned as the function's output. They are an alternative to explicit in-line `return` statements. However the `hasAllocators` return variable in the `getHasAllocator` function was never used.

Consider either using or removing any unused named return variables.

Status update: Acknowledged, won't fix. Client stated - Won't fix for now, as we have already done a release of BaseStrategy. We can address this in a future release.

N02 Withdraw function in BaseStrategy bypass critical checks

The `_withdraw()` function in BaseStrategy bypasses some critical checks inside `_withdrawAsset()` function , unlike the logic of the redeem function group, where all of them will call `_redeemAsset()` function where critical checks are performed, which is a much cleaner and conventional pattern.

Consider carefully document the reason and intended usage of this function to avoid future confusions.

Status update: Acknowledged, won't fix. Client stated - This is more related to BaseStrategy.sol, we'll publish another better documented version of it later there.