YieldNest Default Asset Index Audit

May 2025
Prepared for yieldnest.finance by NFR Audits
nofrontrunning@gmail.com

NFR
AUDITS

Scope

- https://github.com/yieldnest/yieldnest-vault/pull/125 commit 084cd4269cf8acf1289a349c49be8ab1f10f7acd
- https://github.com/yieldnest/wrapped-token/pull/1 commit 87736dc7e788cc50a876d347ad71d41353e78e17

Codebase Overview

This update addresses a major issue - YN vaults convert all supporting asset to baseAsset value for accounting, if baseAsset has <18 decimals, this conversion will lose precision because of truncation.

Solution - introduce default asset in vault - an 18 decimal wrapper for baseAsset to be the base asset in index 0, the idea is all asset will be converted to this 18 decimal asset which has the same price as base asset just more decimals. The original base asset still needs to be supported, so it is set in the the asset list in a random index defined by the vault.

This update also fixed 4 other bugs in the vault.

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 ulnerabilities and issue validation.

Trust Assumptions

There is a privileged roles `Owner` in the hook who is capable of changing the fee collector and fee commission. We assume this privileged role is not malicious.

Severity Classification

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

Issues

Critical

None.

High

H01 Default asset index might change due to asset deletion

For a vault configured with a default asset, it is important to ensure the following condition are always true for the vault to function as expected

1. both base and default assets should not be deleted from the vault
2. both base and default assets' indexes should remain the same all the time

The `deleteAsset` function ensures point 1 by not allowing deletion of either base or default asset in line 146, however point 2 for default asset can not be guaranteed when deleting an asset if the following conditions are met

- the defaultAssetIndex is the last in the list
- the deleting asset's index < defaultAssetIndex this action will cause the default asset being copied to the deleting asset's index and the original default asset popped from the list, leading to change of default asset's index value which violates condition 2, this might cause critical vault functions to revert.

Consider enforcing the `defaultAssetIndex` to always be 1 similar to how base asset always occupies index 0, since base asset can not be deleted, this will make both two assets' index unchangeable.

Status update: Fixed in commit 89d6c.

Medium

M01 Vault might be initialised to wrong states

For a vault to function properly, the base asset should be added to the index 0 of the vault's asset list, similarly the default asset should be added to the vault's `defaultAssetIndex` . Howev er this process is not enforced nor is it checked when initialising the vault, the system relies on the admin to correctly add these assets in after the vault is initialised. This means a vault might be functional before it is properly initialised or it might enter a wrong state after it is initialised (e.g. by deletion of assets as reported in H01).

Consider adding an internal function to do an invariant check requiring the vault's base asset and default asset to be in the right index, and call this internal function before any critical operation is carried out e.g. deposit/withdraw, alternatively consider setting the base and default assets during vault initialisation with a check to ensure the vault system is in the right state at the end of the process.

Status update: Acknowledged, won't fix. Client stated - The vaults are configured and verified thoroughly at launch time, and only made public once verifications pass. Automation with Verification scripts, and running the entire test suite for the new vault + manual verification are employed to verify the vault is configured correctly. This is to keep the vault core code light - and rely on the fact that this initial configuration is a one-time thing.

Low

L01 No relationship checks between base and default asset

The base asset is supposed to be a "virtual" wrapped token of the default asset, this is critical for the vault to function properly. However the vault does not enforcing any relationship checks between these two assets. This means any mis-configurations would be allowed which could cause vault functions to revert or even vault asset tracking to be messed up with potential loss if an oracle is configured for this asset.

Consider enforcing a check whenever a default asset is configured to the vault, requiring default asset is the underlying asset of current base asset.

Status update: Acknowledged, won't fix. Client stated the same reason form M01.

L02 Redundant check when adding asset to storage list

The vault lib function `addAsset` has a check in line 92 to ensure the base asset added has 18 decimals if `countNativeAsset` is true. However this check is redundant after this update because the new base asset is always gonna be 18 decimals from now on, even for index asset that has less than 18 decimals. This is also reflected in the added check in line 97 given the vault decimal is always gonna be 18.

Consider removing this redundant check.

Status update: The client explained they are expecting none 18 decimal vaults, this means this check is needed.

L03 Not enforcing checks on defaultAssetIndex when adding default asset

Under the new design, whenever a vault has a default asset defined (initialized with `defaultAssetIndex != 0`), it is expecting the default asset at this index to have less than 18 decimals. However this is not enforced when adding an asset to the vault, the system does not check if the asset at index `defaultAssetIndex` has less than 18 decimals. This means a default asset of 18 could be added by mistake.

Consider adding an invariant check that the specified asset at index `defaultAssetIndex` has less than 18 decimals to ensure the correctness of the system state.

Status update: Acknowledged, won't fix. Client stated - it is okay to have default asset with 18 decimals, or just same number of decimals as base asset and still be a separate asset. One can argue it's pointless, but it would not cause incorrect behaviour. So currently as long as it's checked one can't add an asset with a number of decimals greater than base asset, one should be fine.

Notes

N01 Unnecessary functions in wrapped token contract

The newly introduced base asset is supposed to be a virtual asset to allow more accurate math computations when dealing with a default asset with less than 18 decimals. This asset will function well as long as it has a price oracle against other assets in the vault, there is no need to actually have balance of this asset for any users at all, same for wrapping or unwrapping functions.

Consider simplifying the wrapped token contract by removing any `mint` related function including wrapping and unwrapping, so the vault does not need to deal with any balances of this asset ever.

Status update: Acknowledged, won't fix. Client stated - We might want that functionality when sending value across max vault and strategy vaults also having WUSDC as base vault so we'd like to to have this functionality in.

N02 Lack of and Incorrect Docstrings

Some of the newly added functions in this update does not have docstrings, for example `totalBaseAssets()` . Some docstrings are incorrect, for example line 57 states the underlying asset is the first asset added to the storage list, but it is incorrect after this update since the Default Asset is now defined by the `defaultAssetIndex` .

Consider adding proper docstrings for all newly added functions and fix all incorrect docstrings.

Status update: Fixed in commit d69dc.

N03 Confusing Naming

Multiple comments in the codebase uses the word "underlying asset", but often referring to different assets in the vault, for example in line 55 it refers to the Default Asset, but in line 64 it refers to the vault share token.

Also multiple comments in `VaultLib.sol` uses primary asset to refer to the Base Asset.

Considering consolidating the terms throughout the codebase into the standard ones - Default Asset, Base Asset, Native Asset and Supported Assets.

Status update: Fixed in commit d69dc.

N04 Missing docstrings

The `initialize` and `_initialize` functions in Vault.sol are missing docstrings for the newly added `defaultAssetIndex_` parameter.

Consider adding docstrings for these functions.

Status update: Fixed in commit e7948.