# AuraBAL compounder report

Date: 2023-03-24
Author: Facundo Spagnuolo
Commit: `7eb8063948dd6dd0a46d2f243b7c2d2fab91b841`
Amendment: 2023-03-27

## High

*None*

## Medium

### M1. Limit the number of reward tokens in the Strategy

Similarly to how the number of extra rewards is limited in the `GenericVault`, consider limiting the number of rewards in the `Strategy`. These can be added indefinitely which could make the harvest logic impossible to be computed in a single block. Consider testing the limit and adding a reasonable cap according to it. Alternatively, consider adding a way to remove reward tokens.

**Answer: “**This is just for harvests we blocked it for `extraRewards` because if there is an overflow there is blocks deposits/withdrawals but there is no risk of that here.”

## Low

### L1. Verify that the max number of extra rewards can be computed successfully

Even though the number of extra rewards is limited in the `GenericVault`, consider having an automated test that to verify this limit actually holds.

**Answer:** “Noted”

### L2. Strategy reward tokens might be useless

The `AuraBalStrategy` contract allows to add any number of reward tokens that will be processed every time the `harvest` function is triggered. However, the way these

reward tokens are processed depends on a separate contract called `tokenHandler`. These are not validated when a new reward token is added, allowing also to be set to the zeroed address. In that case, reward tokens without a token handle are simply skipped. Even though this has no impact, it ends up consuming unnecessary gas units. Consider implementing a better way to handle this situation like adding a way to remove token rewards in case it's necessary.

**Answer:** "Like M1 this is just on harvest as forked from the llama.airforce so we've left it as is."

### L3. `AuraBalStrategy` lacks emitted events

This contract has a few setter methods that do not trigger any events making it hard to track changes on the contract off-chain. Consider adding events for each of those setter functions.

**Answer:** "Noted"

### L4. `CallerIncentiveUpdated` event is not being used

There is an event defined in `GenericVault` called `CallerIncentiveUpdated` that it's not being used in any part of the codebase. Consider removing it.

**Answer:** "Event removed"

### L5. `AuraBalVault` lacks emitted events

This contract has a few setter methods that do not trigger any events making it hard to track changes on the contract off-chain. Consider adding events for each of those setter functions.

**Answer:** "Noted"

## Notes

### N1. Extract magic numbers to readable constants

The `AuraBalVault` contract implements a max extra rewards number that can be extracted to a constant to improve its readability.

### N2. Outdated documentation

The inline documentation in `GenericVault` mentions that withdrawal penalties are removed, while these are actually still part of the implementation.

### N3. Inconsistent naming files convention

There are a few smart contracts that are called differently to their files:

- `AuraBalStrategy` ⇒ `Strategy`
- `AuraBalStrategyBase` ⇒ `StrategyBase`
- `BBUSDHandlerv2` ⇒ `BBUSDHandler`

### N4. Lack of inline documentation

Most parts of the smart contracts changes do not include documentation explaining what's the intention behind them. Consider adding at least some inline documentation to denote what was the intention of the developers on each contract.

### N5. Avoid re-implementing 2-step ownable

Many parts of the codebase rely on different implementations imported from OpenZeppelin. There is another one that might be useful in the `HandlerBase` contract. This contract implements what's known as "2-step ownable" logic which is also provided by OpenZeppelin. Consider reusing that implementation to avoid re-implementing it manually.

### N6. `BBUSDHandlerv2` relies on hardcoded contract addresses

There are a few token addresses and Balancer pool IDs that are hardcoded in the `BBUSDHandlerv2` contract. Consider following the same convention used for the other contracts where these addresses can be passed as arguments in the contract constructor.