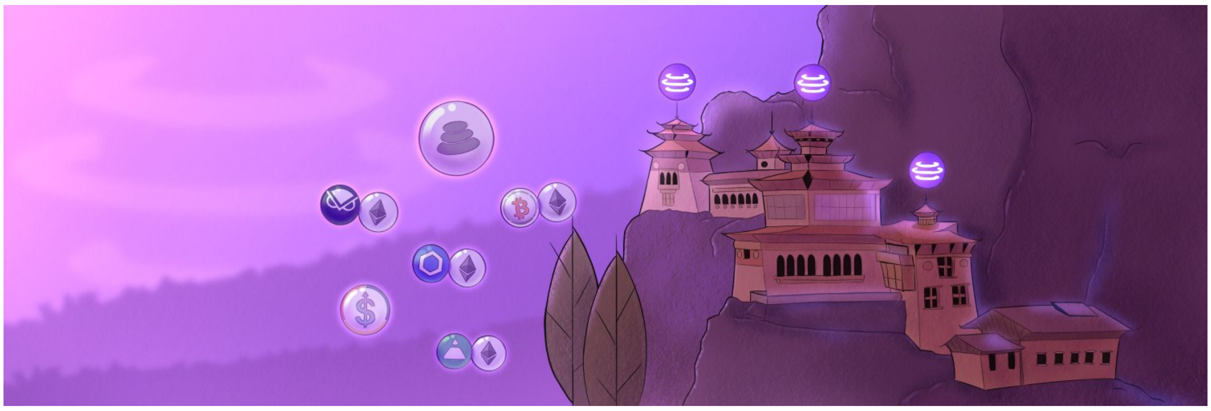

Aura Booster migration report

1. Context

Aura Finance is a protocol built on top of Balancer to provide maximum incentives to Balancer liquidity providers through social aggregation of deposits and Aura’s native token. Aura can be seen as a spin-off of Convex Finance, which implements a similar model for Curve.

One of the key components of this protocol is the Booster contract, the main deposit contract for liquidity pool tokens. Its main responsibility is to allow liquidity providers from different pools to stake their tokens all together in a single place, collect the corresponding rewards, and track the accountancy properly to redistribute these among all the participants.

2. Issues

2.1. Donations

The Booster’s reward pool contracts had a donate function that basically allowed any account to increase the number of rewards to be distributed among all the liquidity providers, but it was also used to determine how much AURA was minted for each liquidity provider.

2.2. Earmark rewards

In a similar way, AURA emissions could have been manipulated since the Booster didn’t take into account possible previous BAL balance when earmarking rewards. When the Booster claimed rewards from the pool’s gauge, it simply checked its current balance to track these in the corresponding rewards pool. This means, anyone could have sent BAL tokens to the Booster to increase the number of rewards to be distributed among all the liquidity providers, but at the same time affecting the number of AURA tokens to be rewarded as well.

2.3. Governance

A few different exploitable scenarios were reported where governance could abuse specific aspects of the code to grief tokens from the protocol. All these required privileged rights, therefore the chances of occurrence were considered low.

3. Approach

Right after these issues were reported to the team, Aura contributors put their biggest effort to work out a migration plan. Along with this, Aura contributors reached out to some external reviewers with large expertise to analyze, review, provide feedback, and follow up closely on every detail of the migration plan.

The migration plan proposed by Aura contributors was the following:

1. Development
    a. Fix all the reported smart contract issues
    b. Work on small tweaks related to the reported issues to improve some protocol aspects
    c. Work on a pool migrator contract to allow users to move their tokens smoothly
    d. Code-review and audit each of these changes
    e. Implement the corresponding deployment scripts
2. Testing
    a. Write unit tests for each of the mentioned changes
    b. Write fork tests to test the pool migrator contract
    c. Write fork tests to test the protocol kept working as expected after the migration
3. Deployment
    a. Craft multisig transactions to shut down the previous contracts that were affected
    b. Craft multisig transactions to migrate pools metadata to the newly deployed contracts
    c. Run fork tests
    d. Deploy pool migrator and offer it through the app to users

4. Fixes

4.1. Drop donation function

The `BaseRewardPool#donate` function was removed entirely to make sure external accounts could not manipulate the number of reward tokens to be distributed among liquidity providers, therefore affecting the number of AURA rewards to be minted.

4.2. Compute BAL balance correctly

A fix was provided to make sure `Booster`'s BAL balance was computed correctly when earmarking rewards. As can be noticed, instead of querying the current BAL balance of the `Booster`, a diff is computed to measure exactly the amount that was claimed from the gauge. This way external accounts cannot manipulate the number of rewards reported to the reward contracts by transferring BAL directly to the `Booster`.

Additionally, if there was any previous amount of BAL held by the `Booster`, it is transferred to the treasury address. Note that there shouldn't be any BAL held in the `Booster` as it is transferred right away to the rewards contract once claimed to the gauge.

4.3. Reentrancy guards

Several reentrancy guards were added to different functions in the `Booster` to make sure any users and in particular governance cannot manipulate these:

- Set fees info
- Set fees
- Shut down pool
- Deposit
- Withdraw
- Earmark rewards
- Earmark fees
- Distribute L2 fees

4.4. Avoid governance to manipulate valid votes

The `Booster` contract had a function called `setVote` that allows the voting delegate to mark any vote hash as valid or not on the `VoterProxy`. This could have been manipulated by governance, therefore Aura contributors decided to block valid votes.

4.5. Avoid BAL entirely for Booster extra rewards

The `Booster` contract can handle extra rewards for non-BAL tokens to be distributed among all liquidity providers. Aura contributors added an explicit validation to make sure BAL cannot be computed as one of these contracts.

5. Features

5.1. Allow distributing fees for L2 pools

A big issue not easy to solve was how to distribute fees corresponding to L2 pools. A new functionality was included in the new Booster contract to allow a delegate to receive the corresponding amount of AURA rewards in exchange for the claimed rewards collected from those pools.

These BAL rewards are forwarded to the reward pools by the booster, and the corresponding AURA rewards are immediately minter to the delegate.

5.2. Add `Transfer` events to reward pools

A `Transfer` event was added to `BaseRewardPool` to improve off-chain accounting processes. This event is not emitted every time balances are affected in the reward contract.

5.3. Implement AURA reward multipliers

A new concept was introduced to allow Aura to control the ratio of AURA rewards designated per pool. Before this, the number of AURA rewards was tied to the number of rewards collected by the Booster. Now, a multiplier can be defined so that the number of AURA rewards can be decreased or increased per pool.

5.4. Increase `Booster` max fees

The maximum value of fees that can be charged by the `Booster` was extended from 2500 to 4000.

5.5. Control max amount of auraBAL to be deposited by `AuraClaimZap`

This contract acts as a helper to allow users to manage several deposits in a single place. Therefore, different options are offered to its interface to allow the user to tell how they want to treat each stake. A new option was added to let them decide the maximum amount of auraBAL to deposit.

5.6. Pool migrator

This one is probably the contract that puts all the pieces together in this migration. It's a simple contract that allows users to withdraw their staked balance from the previous `Booster` and deposit it in the new `Booster`. It allows users to migrate staked balances for multiple pools at once. The user only needs to give this contract allowance to do that.

6. Results

All the changes mentioned above were merged against `aura-contracts` and `convex-platform` repositories under `48b27921236df784dc407e7ea4ee0faa43eed217` and `c17d05039f8ed5cda8fdebb72e0a17f0119521b9` commits once tested and reviewed by the team and the external reviewers.

On December 13th new contracts were deployed to Ethereum mainnet:

- Booster — `0xA57b8d98dAE62B26Ec3bcC4a365338157060B234`
- CvxCrvRewards — `0x00A7BA8Ae7bca0B10A32Ea1f8e2a1Da980c6CAd2`
- AuraClaimZap — `0x2E307704EfaE244c4aae6B63B601ee8DA69E92A9`
- PoolMigrator — `0x12addE99768a82871EAaecFbDB065b12C56F0578`

Before starting with the entire migration process, a thorough fork test was executed based on the newly deployed contracts to check once again the expected outcome. Aura contributors did a great job here, a long list of protocol aspects was covered here, and not only the fixes and new features were checked, but they also made sure the rest of the functionalities provided by the protocol remained unaffected.

Right after that, the following list of events was executed from Aura's multisig:

- Shut down 38 existing pools
- Shut down previous `PoolManager` and `BoosterOwner` contracts
- Set AURA's operator as the new deployed `Booster` instance

- Set new deployed peripheral contracts
- Add back all pools to the new `Booster` instance

These events can be explored through Aura's safe multisig.

Finally, the successful setup was announced on Discord and users started migrating their balances successfully.

7. Conclusions

Aura contributors acted in an extremely professional way during the entire process. All the reported vulnerabilities were analyzed in an isolated way, treating them carefully, while providing not only a fix to cover the reported issues but thinking out-of-the-box to mitigate other potential related issues.

The quality of the fixes and the whole migration process matches perfectly with the quality delivered since this project was born. Several verification steps were taken into consideration in the middle of the migration process to make sure everything was happening as expected.

Congrats to 0xMaharishi and Fry for taking full responsibility for the entire process.

Facundo Spagnuolo

Ethereum developer and security researcher