Aura Finance C4 Contest Findings

Findings

Aura.sol (mint function)

- **Title of Finding:** No validation logic to check for 0 amount
- **Github Link:** Line 91
- **Proposed Recommendation:** Consider writing logic to check for zero amount `require(_amount != 0)`

AuraMinter.sol (constructor, mint function)

- **Title of Finding:** No validation logic to check `_aura` and `dao` addresses passed in constructor parameters
- **Details:** The absence of a validation logic to correctly determine `aura` and `dao` addresses could result in the invalid instantiation of contract with flawed reference to the targeted Aura and Dao contracts
- **Github Link:** Line 20
- **Proposed Recommendation:** Consider writing logic to check for passed in Aura and Dao contract addresses

AuraBalRewardPool.sol.sol (constructor)

- No validation logic to check `_stakingToken`, `_rewardToken`, `_rewardManager`, `_auraLocker` and `_penaltyForwarder` addresses passed in as constructor parameters
- **Details:** The absence of validation logic for the above parameters could result in the incorrect instantiation of contract with flawed reference to the targeted Pool LP, Aura, Depositor, Aura Lock and AuraPenaltyForwarder contracts respectively.
- **Github Link:** Line 62
- **Proposed Recommendation:** Consider writing logic to correctly evaluate the passed in contract address parameters in the constructor

AuraPenaltyForwarder.sol (constructor)

- **Title of Finding:** No validation logic to check `_token`, `_distributor` and `_delay` constructor parameters
- **Details:** Lack of validation logic to check `_token` and `_distributor` parameters could allow erroneous instantiation of contract with wrong/unintended addresses. Lack of logic to check the `_delay` parameter could flaw the contract during instantiation
- **Github Link:** Line 30
- **Proposed Recommendation:** Consider writing logic to correctly evaluate `_token`, `_distributor` and `_delay` parameters respectively in the constructor.

AuraLocker.sol (constructor, contract size)

- **Title of Finding:** No validation logic for `_stakingToken`, `_ cvxCrv` and `_cvxCrvStaking` constructor parameters
- **Details:** Given that no validation logic exists to accurately determine that `_stakingToken`, `_cvxCrv` and `_cvxCrvStaking` address parameters match their corresponding `StakingToken` , `RewardsToken` and `RewardStaking` contract addresses, incorrect addresses could be passed in the constructor and this could lead to invalid reference to the `staking token`, `reward token` and `reward staking` contracts.
- **Github Link:** Link to code
- **Proposed Recommendation:** Consider removing `updateReward` modifier, add this `updateReward` logic in a separate library

AuraClaimZap.sol (constructor)

- **Title of Finding:** No validation logic to evaluate `_crv`, `_cvx`, `_cvxCrv`, `_cvxCrvRewards` and `_locker` constructor parameters
- **Details:** The absence of validation logic for `_crv`, `_cvx`, `_cvxCrv`, `_cvxCrvRewards` and `_locker` constructor parameters could lead to flawed reference to the intended `CRV` token, `CVX` token, contract addresses.
- **Github Link:** Line 68
- **Proposed Recommendation:** Consider adding valid logic to correctly evaluate `_crv`, `_cvx`, `_cvxCrv`, `_cvxCrvRewards` and `_locker` constructor parameters to ensure these parameters match their corresponding contract addresses.

Inconsistent Staking Reward Reference in `AuraLocker.sol` and `AuraClaimZap.sol` Constructor Parameters

- **Details:** Whereas staking rewards is passed in as `_cvxCrvStaking` in the constructor parameter of `AuraLocker.sol` , it is used differently as `_cvxCrvRewards` in the constructor of `AuraClaimZap.sol` . This affects code readability.
- **Github Link:**
  - AuraLocker L152
  - AuraClaimZap L73
- **Proposed Solution:** Given that both constructor parameters reference the same `StakingReward` contract address, consider using a universal parameter name in all cases where they occur so as to enhance code readability.