Security Review

2024-06-21 by SECBIT Labs

At the request of the AladdinDAO team, we conducted a security review of two changes to AladdinDAO-related products.

These two changes are:

- https://github.com/AladdinDAO/aladdin-v3-contracts/pull/191 CLever: add paladin harvest and optimize votium harvest
- https://github.com/AladdinDAO/aladdin-v3-contracts/pull/202 f(x): add token wrapper for decimals < 18

The following introduces the background, change details, security review results, and current status of these two changes.

PR#191 CLever: add paladin harvest and optimize votium harvest

Change Background

> CLever has added the ability to close CVX repayment, with the purpose of keeping the burn ratio of Furnace relatively stable (CVX repayment will recalculate the 14-day release rate of Furnace).
>
> https://github.com/AladdinDAO/aladdin-v3-contracts/pull/191

Potential Issues Found

In the CLeverCVXLocker contract, there are low-level calls that can only be invoked by the keeper role. Attention should be paid to the permissions and parameter settings of the keeper role. In the worst case, it may be possible to transfer tokens in the contract and authorized by users.

https://github.com/AladdinDAO/aladdin-v3-contracts/blob/f5b9baa0ec1d9bee571a7086aedeb26135a2f4fa/contracts/clever/CLeverCVXLocker.sol#L597

https://github.com/AladdinDAO/aladdin-v3-contracts/blob/f5b9baa0ec1d9bee571a7086aedeb26135a2f4fa/contracts/clever/CLeverCVXLocker.sol#L982

Status

Fixed in https://github.com/AladdinDAO/aladdin-v3-contracts/pull/191/commits/f0461e3a49553c37981c3136a35d282ec7f1b449.

The team added permission control, and in fact, only the MultiPathConverter contract will be used.

PR#202 f(x): add token wrapper for decimals < 18

Change Background

> Since the contract for depositing Convex into the Stability pool only supports claiming rewards with 18 decimals, and wbtc has 8 decimals, we specifically created a wrapped wbtc and added it to the reward token of the stability pool to support convex. Relevant contracts were modified https://github.com/AladdinDAO/aladdin-v3-contracts/pull/202
>
> "Convex creates a personal vault for each person, such as 0xF26eedf505D9d32F7dB3c1F1A9f19b540AD485bc, which is used to deposit into the fx Stability pool. The personal vault does not support 8-decimal rewards and calculates everything based on 18 decimals."

Taking the code logic of the given contract address 0xF26eedf505D9d32F7dB3c1F1A9f19b540AD485bc as an example, the parameter rewards represents extra rewards on Convex, currently at the address: https://etherscan.io/address/0x424426916b19A85430C0A95230cA59A081Af038c#code, used for distributing additional rewards. Looking at its reward distribution function notifyRewardAmount(), it can be seen that the rewards are linearly distributed over a given period, currently 7 days (604800). Since the decimals of wbtc is 8, if wbtc is used as a reward for linear release, there will be a large rounding error when calculating rewardRate. Therefore, it is said here that "the personal vault does not support 8-decimal rewards and calculates everything based on 18 decimals."

The given reference address 0xF26eedf505D9d32F7dB3c1F1A9f19b540AD485bc currently does not have any additional rewards set. In actual use, the code logic should be the same as this.

```
function _notifyReward(address _rewardsToken, uint256 _reward) internal {
    Reward storage rdata = rewardData[_rewardsToken];

    if (block.timestamp >= rdata.periodFinish) {
        // @audit When the decimals of the reward token is too small and rewardsDuration
is large,
        // it will produce a large rounding error
        rdata.rewardRate = _reward / rewardsDuration;
    } else {
        uint256 remaining = rdata.periodFinish - block.timestamp;
        uint256 leftover = remaining * rdata.rewardRate;
        rdata.rewardRate = (_reward + leftover) / rewardsDuration;
    }

    rdata.lastUpdateTime = block.timestamp;
    rdata.periodFinish = block.timestamp + rewardsDuration;
}

function notifyRewardAmount(address _rewardsToken, uint256 _reward) external
updateReward(address(0)) {
    require(rewardDistributors[_rewardsToken][msg.sender]);
    require(_reward > 0 && _reward < 1e30, "bad reward value");

    _notifyReward(_rewardsToken, _reward);

    // handle the transfer of reward tokens via `transferFrom` to reduce the number
    // of transactions required and ensure correctness of the _reward amount
    IERC20(_rewardsToken).safeTransferFrom(msg.sender, address(this), _reward);
    emit RewardAdded(_rewardsToken, _reward);
}
```

Potential Issues Found

In the `transfer()` function of the RewardTokenWrapper contract, the `_burn()` function burns the `amount` of wrapped wbtc tokens, and the amount of wbtc tokens that can be retrieved is `amount / scale`. Due to the existence of rounding errors, it may cause the value of the actually burned wrapped wbtc tokens to be higher than the value of the wbtc tokens retrieved by the user, and the excess portion will be permanently left in the contract and cannot be claimed. **It is expected that the actual value of this portion should be negligible.**

```
// @audit RewardTokenWrapper.sol contract
function transfer(address to, uint256 amount) public virtual override returns (bool) {
    address owner = _msgSender();

    if (hasRole(REWARD_POOL_ROLE, owner)) {
      // @audit Burn wrapped wbtc
      _burn(owner, amount);

      // @audit Send the wbtc under the contract to the reward pool
      IERC20Upgradeable(token).safeTransfer(to, amount / scale);
    } else {
      _transfer(owner, to, amount);
    }
    return true;
  }
```

Status

Since the actual amount of assets ultimately left in the contract is almost negligible, both we and the development team believe that this issue does not require special handling.