SHERLOCK

SHERLOCK SECURITY REVIEW FOR

Contest type:	Public
Prepared for:	Beefy
Prepared by:	Sherlock
Lead Security Expert:	bughuntoor
Dates Audited:	May 23 - May 28, 2024
Prepared on:	July 2, 2024

Introduction

Beefy is preparing to revolutionize the market for automated liquidity management with its new cowcentrated liquidity manager (CLM) product. Contribute to their novel product design and win prizes for your help by participating in this Beefy Sherlock Audit Contest

Scope

Repository: beefyfinance/cowcentrated-contracts

Branch: main

Commit: e1e5bc81c830700501624d6b2643b2e8ad5ecb91

For the detailed scope, see the contest details.

Findings

Each issue has an assigned severity:

- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- High issues are directly exploitable security vulnerabilities that need to be fixed.

Issues found

Medium	High
1	0

Issues not fixed or acknowledged

Medium	High
0	0

Security experts who found valid issues

SHERLOCK

DenTonylifer	bughuntoor
Dliteofficial	iamnmt

Issue M-1: Accounting will be broken if `output` token is one of the `lpTokens`

Source: https://github.com/sherlock-audit/2024-05-beefy-cowcentrated-liquidity-manager-judging/issues/71

Found by

DenTonylifer, Dliteofficial, bughuntoor, iamnmt

Summary

Accounting will be broken if `output` token is one of the `lpTokens`

Vulnerability Detail

When the strategy's balances is calculated, it counts both the funds in the LP positions and the funds within the strategy contract, fetched via regular `erc20.balanceOf`.

```
function balances() public view returns (uint256 token0Bal, uint256 token1Bal) {
    (uint256 thisBal0, uint256 thisBal1) = balancesOfThis();
    (uint256 poolBal0, uint256 poolBal1,,,,) = balancesOfPool();

    uint256 total0 = thisBal0 + poolBal0;
    uint256 total1 = thisBal1 + poolBal1;

    // For token0 and token1 we return balance of this contract + balance of
↪   positions - feesUnharvested.
    return (total0, total1);
}
```

```
function balancesOfThis() public view returns (uint256 token0Bal, uint256
↪   token1Bal) {
    return (IERC20Metadata(lpToken0).balanceOf(address(this)),
↪   IERC20Metadata(lpToken1).balanceOf(address(this)));
}
```

The problem is that the `output` token might be one of the `lpTokens` too and any accrued fees that are not yet harvested will be included in this number.

This would unfairly inflate share value when people are depositing via the Vault. Once rewards are collected though, these same depositors would suffer all the losses. Furthermore, it would lead to insolvency as the accrued fees might be deposited in the LP position, hence it will be hard to harvest them in order to temporarily fix accounting.

Impact

Broken accounting, loss of funds, insolvency

Code Snippet

https://github.com/sherlock-audit/2024-05-beefy-cowcentrated-liquidity-manager/blob/main/cowcentrated-contracts/contracts/strategies/velodrome/StrategyPassiveManagerVelodrome.sol#L536C1-L538C6

Tool used

Manual Review

Recommendation

If one of the `lpTokens` is `output` token, deduct the `fees` from the token balance

Discussion

MirthFutures

This is a nice catch, although this can only happen if output is one of the tokens it should be accounted for.

sherlock-admin3

> Escalate I believe I missed this during the judging period. This issue should not be Medium. The root cause is that the `output` token is one of the `lpTokens`, which are set in the `initialize` function, a TRUSTED admin function. This issue is input validation of a trusted function. Therefore, in my opinion, it should be downgraded to Low/Informational, because admin can check this before deployment.

You've deleted an escalation for this issue.

sherlock-admin2

The protocol team fixed this issue in the following PRs/commits: https://github.com/beefyfinance/cowcentrated-contracts/pull/17/files

0502lian

Escalate This issue should invalid. The root cause is that the output token is one of the lpTokens, which are set in the initialize function, a TRUSTED admin(owner) function. This issue is input validation of a trusted function. Therefore, it should be downgraded to Low/Informational, because admin can check this before deployment and will let the output token is not one of the lpTokens. By the way, I brought up this escalation mainly because the current rules for TRUSTED admin seem unclear. There are too many issues about TRUSTED admin. Some are judged as valid, while others are deemed invalid.

sherlock-admin3

> Escalate This issue should invalid. The root cause is that the output token is one of the lpTokens, which are set in the initialize function, a TRUSTED admin(owner) function. This issue is input validation of a trusted function. Therefore, it should be downgraded to Low/Informational, because admin can check this before deployment and will let the output token is not one of the lpTokens. By the way, I brought up this escalation mainly because the current rules for TRUSTED admin seem unclear. There are too many issues about TRUSTED admin. Some are judged as valid, while others are deemed invalid.

You've created a valid escalation!

To remove the escalation from consideration: Delete your comment.

You may delete or edit your escalation comment anytime before the 48-hour escalation window closes. After that, the escalation becomes final.

z3s

I asked the sponsor about this on Discord. Their decision is to support this scenario and fix the accounting issue. Therefore, having the output token as one of the LP tokens is an expected outcome. The root cause is the wrong accounting, which makes this a valid Medium issue. So, if they decided it should revert when the output token is one of the LP tokens, then it is an input validation issue and should be classified as Informational.

IWildSniperI

@z3s we judge according to sherlock judging rules so a sponser that could make a mistake himself and got a value from the report makes this issue in best cases `Low` severity quoting this from sherlock rules

> List of Issue categories that are not considered valid: Admin Input/call validation: Protocol admin is considered to be trusted in most cases, hence issues where Admin incorrectly enters an input parameter. Example: Make sure interestPerMin > 1 ether as it is an important parameter. This is not a valid issue. Admin could have an incorrect call order. Example: If an Admin forgets to setWithdrawAddress() before calling withdrawAll() This is not a valid issue. An admin action can break certain assumptions about the functioning of the code. Example:

> Pausing a collateral causes some users to be unfairly liquidated or any other action causing loss of funds. This is not considered a valid issue. As mentioned in the standards observed, in the case of a restricted admin, the restriction must be clearly mentioned for any issue in this category to be considered valid

so its not mentioned restrictively that a check of `output` is in place, hence we are assuming the admin to do the righ thing

Dliteofficial

I believe the bone of contention is relating to whether the admin will knowingly or unknowingly pass in a CLPool with one of the LpTokens as the output/reward token. Being trusted IMO has no relevance here. Adding a check to ensure that this scenario is addressed is similar to what you would find in a staking pool where only the owner can add a reward token into the pool yet there's a check to ensure that the reward token is not the staking token. This honest mistake will cause catastrophic damage to the protocol. If the impact here doesn't entirely describe the severity of the issue, you should see #77.

Because there's no clear delineation, the accounting for both the LP Token and the output is now mushed into each other and there's at least one party who stands to lose everything.

DenTonylifer

Let me give an example that will help to solve this question. Here is a similar report from a previous audit: the problem described in the report will ONLY occur because of the value entered by the TRUSTED admin. But this report was accepted for the following reason:

> there's a specific code path that not just allows but expects the admins to set configs[i].start in the future So, if the admins were to prevent the bug in issue 245, that piece of code would become dead code, which doesn't really make sense.

Thus, if the contract clearly expects the administrator to enter certain data, but after entering this data an unexpected problem occurs - this is a valid issue.

If we look at `_chargeFees()` and `balances()` functions, we will clearly see that they expect that `output` might be the same as one of the `lptokens`. If we look at `beforeAction()` function, we will clearly see that protocol aware that unharvested fees can cause accounting problem and takes measures to prevent this. If the protocol expected that `output` will never be the same as one of the `lptokens`, then `that piece of code would become dead code, which doesn't really make sense.`

And also:

> So, if they decided it should revert when the output token is one of the

> LP tokens, then it is an input validation issue and should be classified as Informational.

This fix was made after the end of the audit and cannot affect the validity of the issue.

Dliteofficial

So, I looked into your submission again, @DenTonylifer. Your escalation is predicated on the fact that token0 could be the same as the output tokens on which the fees are being charged, which could result in broken accounting. If that's true, then I agree that it should be a duplicate of this report then.

spacegliderrrr

As per the readme, protocol should be able to work with any non-weird ERC20s (such one is `VELO`). Velodrome's gauges have `VELO` as their output token and hence, creating a strategy on top of a pool which has `VELO` as one of the `lpTokens` will break accounting.

Given that there are currently 37 pools which have `VELO` as one of their `lpTokens` it is reasonable that strategies will be built on top of them.

nevillehuang

Agree with @spacegliderrrr, planning to reject escalation and leave issue as it is.

WangSecurity

Result: Medium Has duplicates

sherlock-admin4

Escalations have been resolved successfully!

Escalation status:

- 0502lian: rejected

spacegliderrrr

Fix looks good. `balanceOfThis` now checks if one of the `lpTokens` is the same as `output` token and deducts fee amount if necessary.

sherlock-admin2

The Lead Senior Watson signed off on the fix.

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users’ sole risk and is the users’ responsibility.