SHERLOCK

SHERLOCK SECURITY REVIEW FOR

Contest type:	Public
Prepared for:	Kwenta
Prepared by:	Sherlock
Lead Security Expert:	g
Dates Audited:	July 31 - August 3, 2024
Prepared on:	August 23, 2024

Introduction

This audit focuses on a USDC fee streaming upgrade to our KWENTA staking rewards contract; adding USDC as a second claimable token type on the contract.

Scope

Repository: Kwenta/token

Branch: dualRewards

Commit: 38114f26b5cc19e59ab5ff78578855df7bc41f85

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

eeyore
g
aslanbek
tedox

Issue M-1: Attacker will prevent distribution of USDC to stakers through frequent reward updates

Source: https://github.com/sherlock-audit/2024-07-kwenta-staking-contracts-judging/issues/22

Found by

aslanbek, eeyore, g, tedox

Summary

USDC's lower precision of 6 decimals and frequent reward updates will cause stakers to receive 0 of the allotted 10K USDC weekly rewards.

> distributions should be done at the pace of 10k per week until fully distributed

Root Cause

Using the same reward-per-token calculation for USDC used for KWENTA is a mistake as USDC only has 6 decimals of precision compared to KWENTA's 18 decimals.

Internal pre-conditions

1. The rewardsDuration is 1 week. This is the default duration.
2. The `StakingRewardsNotifier` sends 10_000e6 USDC of rewards to `StakingRewards`. Kwenta Governance will be distributing 10K USDC per week as rewards.
3. There is at least 100_000e18 KWENTA staked in `StakingRewardsV2`. Note that KWENTA's current total supply is at ~631_000e18.

External pre-conditions

No response

Attack Path

1. The `StakingRewardsNotifier` sends 10_000e6 USDC to `StakingRewards` via `notifyRewardAmount()`. The rewardRateUSDC is now equal 16534 (`10_000e6 / 1 weeks`). This is scheduled to happen every week.

2. Every 1 to 3 blocks, the attacker will trigger an update of rewards by calling stake or unstake in `StakingRewards`. The reward-per-token for USDC is calculated:

```
((lastTimeRewardApplicable() - lastUpdateTime) * rewardRateUSDC * 1e18) /
↪  allTokensStaked
```

When 3 blocks have passed, the time since the last update would be 6 seconds because block time in Optimism is 2 seconds per block.

```
(6 * rewardRateUSDC * 1e18) / allTokensStaked
```

`rewardRateUSDC` is 16534 and `allTokensStaked` is 100_000e18.

```
(6 * 16534 * 1e18) / 100_000e18  // ==> this will return 0
```

Since the reward-per-token for USDC is 0, none of the 10_000e6 USDC will be distributed to the stakers.

Note that the attacker does not need to update rewards every 1-3 blocks if there is organic activity on every block from users interacting with `StakingRewardsV2` and trigger the rewards update.

Impact

All stakers will receive 0 USDC rewards of the 10_000e6 USDC weekly rewards.

PoC

The `test_rewardPerTokenUSDC` test in `StakingRewardsV2.t.sol` can be modified to the following test case.

```
function test_rewardPerTokenUSDC() public {
    // fund so that staking can succeed
    uint256 stakedAmount = 100_000e18; // @audit 100_000 staked KWENTA
    fundAndApproveAccountV2(address(this), stakedAmount);

    // check reward per token starts as 0
    assertEq(stakingRewardsV2.rewardPerTokenUSDC(), 0);

    // stake
    stakingRewardsV2.stake(stakedAmount);
    assertEq(stakingRewardsV2.totalSupply(), stakedAmount);

    // set rewards
```

```
    uint256 reward = 10_000e6; // @audit 10,000 USDC will be distributed every
↪   week
    vm.prank(address(rewardsNotifier));
    stakingRewardsV2.notifyRewardAmount(0, reward);

    // ff to end of period
    vm.warp(block.timestamp + 2);

    // check reward per token updated
    assertEq(stakingRewardsV2.rewardPerTokenUSDC(), 1 ether); // @audit this
↪   will return 0
}
```

The test shows that none of the rewards are getting distributed for the given values.

Mitigation

A possible mitigation is to convert the USDC rewards to an 18-decimal value when computing for `rewardRateUSDC`. When claiming rewards, convert the earned USDC rewards back to 6-decimal precision. If it is 0, store the unclaimed partial values of USDC until they can be claimed whole.

Discussion

sherlock-admin3

> Escalate The severity of this issue is medium due to the following reasons.
>
> 1. The griefing attack is not realistic. To prevent distribution of `10K` USDC rewards, attacker should should update rewards `86400 * 7s / 6s = 100_800` times per week. On the other hand, since the average transaction fee of Optimism is `0.1$ ~ 0.5$`, the attacker should pay `0.1$ * 100_800 = 10_080$` at minimum.
> 2. The count of organic activities per week will be not so much. Assuming the count of organic activities are 1000 at maximum, the loss of USDC rewards will be less than `0.1$ * 1000 = 100$`.
> 3. As I see from this issue itself, even `g` and `tedox` estimated the severity of their own issue as medium.
>
> From above, I insist that the severity is medium.

You've deleted an escalation for this issue.

0xklapouchy

@debugging3 recheck on my finding #18 and you will be informed that cost of this attack is like 18$ for whole week.

olaoyesalem

@0xRandluck this is a duplicate for this issue https://github.com/sherlock-audit/2024-07-kwenta-staking-contracts-judging/issues/4

debugging3

@0xklapouchy I tested the following function to calculate the gas amount

```
function test_updateRewards() public {
    stakingRewardsV2.getReward();
}
```

And the result is

```
[PASS] test_updateRewards() (gas: 37884)
```

So the transaction fee on Optimism is `37884 * 0.1e9 / 1e18 * 2528$  0.01$` and the total cost per week is `604800 / 45 * 0.01$ = 134.4$` which is larger than your `18$`. Anyway, the cost is much small than `10K`, so I will retreat my escalation. Thank you for your advice.

MGF15

#124 is also a dup @debugging3

twicek

I could not escalate in time so feel free to ignore my comment if you want. This issue is clearly valid but should not be high severity in my opinion because USDC can be recovered and it needs about 1/10 of the total supply of KWENTA staked to happen. On top of that, 2/4 issues duplicated with this report agreed with a medium severity. What was the reason behind the choice of high severity? @0xRandluck

0xklapouchy

@twicek This is not about the USDC being unrecoverable, but about the loss of user rewards. These rewards, which should go to users, are not recoverable, resulting in a direct loss of funds for each staker.

All accounting that should be tracked on-chain is lost.

twicek

@0xklapouchy USDC being recoverable is important as it reduces the impact of the issue, users can be paid back later. I agree, there is a loss of rewards for the users but only under specific conditions. Also, a loss of rewards is not a direct loss of funds. Therefore, it qualifies as medium severity.

whitestrong0820

@0xRandluck This issue does not satisfy the Sherlock's high severity criteria. So I think this is medium severity

gjaldon

@twicek in Sherlock, loss of rewards is not treated as less severe as any other type of loss of funds. The protocol can recover the USDC but not the stakers. The stakers lose their rewards (loss of funds) and the attacker's actions mess up the accounting for the rewards.

WangSecurity

Firstly, I agree that loss of rewards doesn't decrease the severity by default and it's also a loss of funds.

Secondly, the tokens are not lost completely, can be recovered from the contract and distributed later. I believe this indeed decreases the severity. Hence, I agree that the medium is more appropriate here. Planning to downgrade the severity to medium.

About #4, being a duplicate, it's not. It talks about admins not sending enough tokens causing the rounding down. I believe it's an admin mistake to not send enough tokens. Hence, it's not a sufficient duplicate, cause it's invalid based on the admin's rules and not having a valid attack path.

0xklapouchy

@WangSecurity

I would like to disagree on classifying the severity as `Medium`. The `distributed later` approach is difficult to implement, and the development costs in terms of man-days could exceed the value of the rewards not distributed.

Additionally, it was mentioned on Discord that USDC should, in fact, not be recoverable. I even requested an update to the README to reflect this, but it hasn't been done. While I agree that USDC may be recoverable in the current code, this will not be the case in the final code, as the protocol will address this issue.

WangSecurity

> I would like to disagree on classifying the severity as Medium. The distributed later approach is difficult to implement, and the development costs in terms of man-days could exceed the value of the rewards not distributed

Fair enough, but since the rewards are not permanently locked in the contract and can be retrieved decreases the severity. Similar to upgradeability of the contracts, i.e. if the funds are stuck in the contract, but the contract is upgradeable, I believe it decreases the severity, but none of these 2 reasons can solely invalidate an issue.

> Additionally, it was mentioned on Discord that USDC should, in fact, not be recoverable. I even requested an update to the README to reflect this, but it hasn't been done. While I agree that USDC may be recoverable in the current code, this will not be the case in the final code, as the protocol will address this issue.

Hm, thank you for pointing this out and excuse me if you tagged me, but I missed the notification. For future reference, in those cases DM me. But, now we cannot exclude the fact that USDC can be recovered as the code has the implementation for it.

Hence, I'm still planning to decrease the severity to medium.

sherlock-admin2

The protocol team fixed this issue in the following PRs/commits:
https://github.com/Kwenta/token/pull/257

WangSecurity

Plan to apply my decision in a couple of hours.

gjaldon

The issue has been fixed as recommended. `rewardRateUSDC` is now scaled up to 18-decimal precision, just like `rewardRate` (for KWENTA). When rewards are claimed, the USDC rewards are downscaled to USDC decimals.

gjaldon

Relevant lines for the fix: https://github.com/sherlock-audit/2024-07-kwenta-staking-contracts/blob/e72a8031b317705813c48252f3d1a6b9a25e8d8a/token/contracts/StakingRewardsV2.sol#L378 https://github.com/sherlock-audit/2024-07-kwenta-staking-contracts/blob/e72a8031b317705813c48252f3d1a6b9a25e8d8a/token/contracts/StakingRewardsV2.sol#L665-L674

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.