Rewards Manager Audit - TheGraph

OpenZeppelin Security

The Graph team asked us to review updates to the `RewardsManager` contract. We looked at the code and now publish our results.

Scope

We audited PR528 up to commit `fdbc60c1e64450123c142d5710a0ae54f489eb8b`. We also took the opportunity to review:

- `RewardsManager`
- `IRewardsManager`
- `RewardsManagerV1Storage`
- `RewardsManagerV2Storage`

We also reviewed related parts of the existing code base to the extent that additional context was necessary to understand the code under audit. All other code and contract dependencies were assumed to work as documented.

System Overview

The `RewardsManager` contract performs the internal accounting necessary to assign rewards to subgraph indexers, funded through inflation of the Graph Token supply. Specifically, new rewards are accrued every block and assigned to subgraphs in proportion to the amount of their signalled tokens. Those rewards are divided amongst the indexers in proportion to their allocation sizes. When an allocation is closed, the assigned tokens are minted and transferred to the indexer. The `RewardsManager` contract tracks how reward assignments change as users change their subgraph signalling and indexer allocations.

Privileged Role

There are a few administrator powers that affect reward distributions:

- the governance structure is able to change the issuance (inflation) rate. The only restriction is that it must be positive.
- there is a Subgraph Availability Oracle address that manages a blacklist of subgraphs. Indexers that allocate tokens to a blacklisted subgraph will be unable to retrieve their rewards. It should be noted that blacklisted subgraphs are still assigned rewards internally, which has two interesting consequences:
    - the rewards issued to valid subgraphs may be lower than expected.
    - if a subgraph is removed from the blacklist, the indexers will be able to claim rewards accrued while it was on the blacklist.
- the governance structure or the Subgraph Availability Oracle address can choose a minimum subgraph signal threshold. Subgraphs do not accrue rewards while their signal is below this threshold.

Critical Severity

None.

High Severity

None.

Medium Severity

[M01] Reward issuance excludes pending rewards

The amount of new rewards assigned to token holders is proportional to the token supply. However, the tokens are only minted when closing a subgraph allocation, which occurs at the choice of the users. This has the unusual side effect that the

inflation rate depends on how often users extract their rewards. In the interest of predictability, consider applying the issuance rate to all assigned tokens, which includes the token supply as well as any withdrawable rewards. If this recommendation is implemented, it should be decided and documented whether "withdrawable rewards" includes the rewards assigned to subgraphs on the deny list (which may become withdrawable in the future) and rewards assigned to subgraphs with no allocations (which cannot be withdrawn in practice).

Low Severity

[L01] Reward getter functions may overestimate rewards

The `getAccRewardPerSignal`, `getAccRewardsForSubgraph`, `getAccRewardsPerAllocatedToken` and `getRewards` functions of the `RewardManager` contract rely on the `getNewRewardsPerSignal` function. In all cases, the getter functions return a projected value that would be appropriate if a user were to take their rewards immediately. Since additional rewards are distributed over time, these getters will typically underestimate the rewards that a user will eventually receive.

Moreover, the contract caches the reward assignments before signal or allocations updates, which effectively locks in the rewards accrued so far. However, the contract does not cache the reward assignments when the Graph token supply changes, which also affects the reward calculations. When new tokens are minted, the rewards to be distributed increases, so the getter functions still underestimate the eventual rewards. However, the Graph Token is also `Burnable` and if tokens are burned, the projected rewards will reduce, so previous calls to the getter function would overestimate the eventual rewards received. This may be confusing to users who assume the getters are monotonically increasing and the claimed rewards are guaranteed.

In the interest of predictability, consider either:

- disabling the ability to burn graph tokens,
- caching the accumulated rewards before any burn operation, or

- documenting that the getters are merely projections that may overestimate the actual rewards

[L02] Incomplete NatSpec

The following Ethereum Natural Specification (NatSpec) comments were found to be incomplete:

- the `isDenied` function of the `RewardManager` does not have a `@return` statement.
- the `getAccRewardsPerSignal` function of the `RewardManager` does not have a `@return` statement.

[L03] Misleading comments

The following misleading comments were identified:

- The `@dev` comment on the `_setDenied` function of the `RewardsManager` says "internal" instead of "private".
- The comment on the `_pow` function of the `RewardsManager` contract points to the `master` branch of the `makerdao/dss` repository. Since this is subject to change, it should point to the particular commit that was used.

[L04] No rewards at minimum threshold

The `minimumSubgraphSignal` is described as the minimum number of signaled tokens for a subgraph to accrue rewards. However, rewards are only accrued if the number of signaled tokens is strictly greater than this bound. Consider using an inclusive bound so rewards accrue at the minimum threshold, or renaming the variable and clarifying the description to match the current behavior.

Notes & Additional Information

[N01] Typographical errors

The following typographical errors were identified:

- In the `RewardsManager` contract, "subgraph" should be "subgraphs"
- In the `RewardsManager` contract, on lines 237, 263, 314 and 337, "Subgraph deployment" should be "Subgraph deployment ID"
- In the `RewardsManager` contract, on lines 357 and 394, "Allocation" should be "Allocation ID"

Conclusions

No critical or high severity issues have been found. Recommendations have been proposed to improve code quality, reduce the attack surface, and minimize errors.