OpenZeppelin | security

Availability Manager and Minimum Allocation Duration Removal Audit

The Graph

February 12, 2024

Table of Contents

Summary

Type	Infrastructure
Timeline	From 2024-01-22 To 2024-01-29
Languages	Solidity

Total Issues	8 (6 resolved, 1 partially resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	0 (0 resolved)
Low Severity Issues	3 (2 resolved, 1 partially resolved)
Notes & Additional Information	4 (3 resolved)
Client Reported Issues	1 (1 resolved)

Scope

We audited the following pull requests of the graphprotocol/contracts repository:

- Pull request #882 at commit 1ad7935.
- Pull request #902 at commit 0a7dca8.

Overview of Changes

Subgraph Availability Manager

The current `subgraphAvailabilityOracle` (SAO) in the `RewardsManager` contract is responsible for verifying the availability of subgraph files, conducting validity checks, and reporting the status on-chain by either denying or allowing rewards on the subgraph. The proposed changes aim to enhance the decentralization and robustness of the protocol by introducing five separate SAO accounts and requiring a minimum of three votes among the SAOs before a subgraph is denied/allowed rewards.

The pull request introduces the `SubgraphAvailabilityManager` (SAM), a new contract responsible for aggregating votes, enforcing the execution threshold, and discarding old votes after a configurable interval. The newly deployed contract will replace the `subgraphAvailabilityOracle` address of the `RewardsManager` contract, requiring no changes to the `RewardsManager`. Every 5 minutes (oracle cycle), the oracles decide whether to deny or allow indexing rewards for certain subgraphs if a change in availability is detected. This is done by checking that the subgraph manifest is available on IPFS (through the `subgraphDeploymentID`). Subgraphs will be denied if the manifest misses, is incorrect (cannot be indexed) or contains features not allowed by governance.

The system works under the assumption that the majority of the oracles are honest. Note that an oracle discovered to be dishonest can be removed through the `setOracle` function. The `governor` of the SAM will be set as The Graph Council. It is expected that oracles will all vote the same unless they are malfunctioning since they run the same software and have the same data to look at. If some oracles malfunction or temporarily misbehave, they will revert to correct behavior within the next oracle cycle, or will be replaced by governance.

Remove Minimum Allocation Duration Restriction

The proposed pull request is the implementation of Graph Improvement Proposal 60. It removes the restriction that indexers must wait for at least one epoch before they can close an allocation.

Security Model and Trust Assumptions

Within `SubgraphAvailabilityManager.sol`, there are several functions under access control:

- The `setVoteTimeLimit` function can only be called by the `governor` role.
- The `setOracle` function can only be called by the `governor` role.
- The `vote` function can only be called by the `oracle` role.
- The `voteMany` function can only be called by the `oracle` role.

We assume that the governor is non-malicious, will act in the protocol's best interest, will correctly set parameters and will not set the same oracle twice. In addition, we assume that the majority of oracles are honest and will vote truthfully.

Low Severity

L-01 Missing Input Validations

There are several validations that could be introduced to further reduce potential user error.

When constructing the SAM, consider validating that `_executionThreshold <= NUM_ORACLES` to avoid redeployment if a bad value is supplied. Furthermore, consider validating that oracles are not duplicated inside the `oracles` array, both during construction and in the `setOracle` function.

Update: *Partially resolved in pull request #935 at commit fbdf6af. The Graph team stated:*

> *We discussed with the team and we feel the impact of setting duplicated oracles is not severe and it is easily solvable by replacing that oracle. Checking for duplicates in the array would increase gas consumption and complexity.*

L-02 Votes From Earlier Round Can Remain Valid Resulting in Premature Consensus

When oracles vote to allow or deny a subgraph, the `checkVotes` function will return `true` as soon as the execution threshold is reached and the `RewardsManager` state will be updated. It is possible that the state may be updated prematurely if valid votes from a previous round still exist. For example, suppose that under ideal network conditions all oracles perform their availability checks and vote if needed at every time interval `T`. A certain subgraph is having issues with its availability and is being repeatedly allowed/denied.

At time `T`, all oracles cast an allow vote, updating the `RewardsManager` state. However, at time `2T`, the subgraph goes offline again and the oracles vote to deny, updating the state again. The votes cast at time `T` still remain valid up to time `T + voteTimeLimit`, which may be greater than `2T`. As such, there can exist a window in which the state can be reset to that of an earlier round using a single vote.

Consider modifying the voting logic to ensure that votes that occurred in a previous round can never be seen as valid in the most recent round.

Update: *Resolved in pull request #939 at commit 1ce46f0.*

L-03 Incomplete Docstrings

Within `SubgraphAvailabilityManager.sol`, there are several parts that have an incomplete docstring:

- The `index` and `oracle` parameters of the `OracleSet` event are not documented.
- The `voteTimeLimit` parameter in the `VoteTimeLimitSet` event is not documented.
- The `subgraphDeploymentID`, `deny`, `oracleIndex`, and `timestamp` parameters of the `OracleVote` event are not documented.

Consider thoroughly documenting all functions/events (and their parameters or return values) that are part of any contract's public API. When writing docstrings, consider following the Ethereum Natural Specification Format (NatSpec).

***Update:** Resolved in pull request #936 at commit 94e6d05.*

Notes & Additional Information

N-01 `OracleVote` Event Emitted in Unnatural Order

If an oracle vote triggers the majority consensus, it will deny/allow the subgraph and emit a `RewardsDenylistUpdated` event in the process. When examining emitted events, it would seem that first the subgraph was denied/allowed, and only after that did the oracle cast a vote since the `OracleVote` event is emitted at the end of the `_vote` function. This does not follow the natural course of events, which could be confusing to an observer.

Consider emitting the `OracleVote` event before calling the `rewardsManager` to update subgraph availability.

***Update:** Resolved in pull request #937 at commit 9f9db2e.*

N-02 Gas Inefficiencies

There are several places across the codebase where changes can be made to improve gas consumption:

- The assignment of variables to their default value is unnecessary: #1, #2, #3, #4.
- The `OracleVote` event emits the timestamp of the current block. This is unnecessary since it could be deduced off-chain from the timestamp of the block the event was emitted in.
- In the `SubgraphAvailabilityManager` contract, the `SafeMathUpgradeable` library is only used for a calculation which is not expected to cause overflows or division-by-zero errors. Consider removing the library to simplify the code and reduce deployment cost.

When performing these changes, aim to reach an optimal trade-off between gas optimization and readability. Having a codebase that is easy to understand reduces the chance of errors in the future and improves transparency for the community.

Update: *Resolved in pull request #938 at commit f2aa2fa.*

N-03 Subgraph Availability Manager Deprecates `RewardsManager.setDeniedMany`

After the introduction of the SAM, the `setDeniedMany` function of the `RewardsManager` will become unused.

If there is no intention of using it in the future, to increase code clarity, consider removing this function both from interface and implementation.

Update: *Resolved in pull request #940 at commit ef35364.*

N-04 Lack of Security Contact

Providing a specific security contact (such as an email or ENS name) within a smart contract significantly simplifies the process for individuals to communicate if they identify a vulnerability in the code. This practice is quite beneficial as it permits the code owners to dictate the communication channel for vulnerability disclosure, eliminating the risk of miscommunication or failure to report due to a lack of knowledge on how to do so. Moreover, if the contract incorporates third-party libraries and a bug surfaces in those, it becomes easier for the maintainers of those libraries to contact the appropriate person about the problem and provide mitigation instructions.

The `SubgraphAvailabilityManager` contract does not have a security contact.

Consider adding a NatSpec comment containing a security contact above the contract definitions. Using the `@custom:security-contact` convention is recommended as it has been adopted by the OpenZeppelin Wizard and the ethereum-lists.

Update: *Acknowledged, will resolve. The Graph team stated:*

> *We will have this in mind when deploying the smart contract. We currently do not have an email account to use for this.*

Client Reported

CR-01 Rounding Error in L2 Curation

A rounding error has been discovered when minting signal on L2: if less than 100 wei GRT is deposited, the curation tax rounds down to 0 thereby allowing an attacker to mint signal without incurring a fee. The proposed fix changes the calculations such that the curation tax rounds up instead of down. Note that while the same vulnerability also exists on L1, a minimum curation deposit of 1 GRT (equal to 1e18 wei) is currently enforced, so rounding down to 0 can never happen.

Update: *Resolved in pull request #1 of the GHSA-p4j4-4h8c-rrc6 security advisory.*

Conclusion

The Graph aims to update its oracle design for reward management, moving towards a more decentralized voting system. The design is straightforward and well-documented, and only a few low-severity issues were found during the audit. In addition, recommendations aimed at improving the readability and clarity of the codebase have also been made.

No issues were found with pull request #902.

The Graph core developers were highly responsive throughout the audit period and answered all questions regarding the protocol.