OpenZeppelin | security

Pull Request #1183 Review

The Graph

June 25, 2025

Table of Contents

Summary

Type	Governance
Timeline	From 2025-06-19 To 2025-06-20

Languages

Scope

OpenZeppelin audited pull request #1183 of the graphprotocol/contracts/ repository. In scope were all the non-test Solidity files present in the pull request.

Overview

On June 5th, The Graph team informed OpenZeppelin about several potential issues overlooked by the Horizon audit, as well as a slight change of behavior. Subsequently, a two-day review of pull request #1183 was conducted to validate the correctness and security outlook of the fixes that had been implemented to address these issues. The review team confirmed that pull request #1183 had correctly fixed the identified issue.

Reported Issues

Issue #1

In the `createIndexingDispute` function of the `DisputeManager` contract, the `allocationId` and `poi` parameters are used to create IDs for indexing disputes. The uniqueness of these arguments is then enforced to prevent the same dispute from being posted multiple times. However, this logic does not take into account the fact that the Horizon architecture actually permits repeatedly posting the same POI for a given allocation. This would result in the indexer only being subject to slashing the first time they are disputed, but any further violations would not be punished because the dispute creation would revert.

***Update**: Fixed in commit 97dceb1 of pull request #1183. An extra `blockNumber` argument has been added to the `createIndexingDispute` function. This allows for creating one dispute each time a POI is posted, regardless of its uniqueness.*

Issue #2

The `DisputeManager` contract checks an indexer's stake to decide if they are disputable or not. However, it only checks the indexer's provisioned tokens amount. If it is 0, then the indexer is deemed not disputable and any dispute creation calls will revert. This fails to consider delegation which might also be slashable.

***Update**: Fixed in commit d89440a of pull request #1183. Now, the delegation is also considered during the dispute creation.*

Issue #3

A subtle discrepancy exists between the legacy `closeAllocation()` function and the current rewards distribution behavior in Horizon. Legacy allocations that are opened and closed within the same epoch experience different reward calculation outcomes compared to the current system, creating inconsistent behavior despite the functions being intended to operate identically. It is important to note that Graph Horizon removes the ability to create legacy allocations. Hence, the inconsistency mentioned above can only be leveraged if an attacker opens an allocation right before the Horizon update and closes it right after, without exceeding one epoch.

***Update**: Fixed in commit 1c7e4cf of pull request #1183. The reward distribution behavior has been modified to ensure that the legacy function exactly matches the current implementation.*

Conclusion

Pull request #1183 addresses each of the three reported issues and fixes them correctly.