OpenZeppelin | security

The Graph Dispute Manager Incremental Audit

The Graph

August 30, 2023

Table of Contents

Summary

Type	DeFi	Total Issues	4 (0 resolved)
Timeline	From 2023-07-31 To 2023-08-11	Critical Severity Issues	0 (0 resolved)
Languages	Solidity	High Severity Issues	1 (0 resolved)
		Medium Severity Issues	0 (0 resolved)
		Low Severity Issues	0 (0 resolved)
		Notes & Additional Information	3 (0 resolved)

Scope

We audited pull request #766 at commit `a928e0b`.

In scope were the following contracts:

```
contracts
└── disputes
    ├── DisputeManager.sol
    └── IDisputeManager.sol
```

System Overview

The `DisputeManager` contract enables fishermen to create three different types of disputes: indexer disputes, query disputes, and query conflict disputes. In the indexer and query dispute case, fishermen must provide a minimum deposit, currently set at 10,000 GRT, in order to create the dispute, with its status set as `Pending`. Disputes can be either accepted, rejected, or drawn by a selected arbitrator. Performing any of the previous actions will change the dispute status from `Pending` to `Accepted`, `Rejected`, or `Drawn` respectively. Originally, once disputes were resolved, they would be deleted from the `DisputeManager` contract. Removing the dispute from the contract made it difficult for third parties to determine the outcome of a dispute and the relevant information of each dispute as they were required to filter events on the blockchain to ascertain all previous disputes and their resolutions. The pull request that was audited removes the logic that deletes a dispute upon resolution. Additionally, the status of each resolved dispute is stored, and additional logic has been added in order to prevent disputes from being re-used now that they continue to exist in the contract after being resolved.

Security Model and Trust Assumptions

Upgrade Is Not Backwards-Compatible

The changeset adds new variables and logic that are not backwards-compatible with disputes that are still in progress in the original `DisputeManager` contract. Disputes that have not been resolved before the upgrade will be unresolvable. Therefore, it is important that all disputes are resolved before the upgrade, and that fishermen are aware of the pending upgrade so that they hold off on creating new disputes.

Some Assumptions in Arbitration Charter Will Change

The Arbitration Charter states that disputes against the same attestation cannot happen more than once (double jeopardy). Currently, this is an implicit assumption that requires the Arbitrator to have historical knowledge of past disputes, since dispute state is deleted upon resolution. With this change, that implicit assumption will become an explicit assumption, as now the dispute state will be stored along with the status of the resolution. The Arbitrator can look up certain past dispute information on-chain. This explicitly prevents the same fisherman from creating duplicate disputes once the original dispute is resolved. However, it is possible for different fishermen to create query disputes against the same resolved attestation. In such a case, the responsibility falls on the arbitrator for justified arbitration.

Privileged Roles

The Graph Foundation is the governor of the `DisputeManager` contract, and it can:

- Upgrade the `DisputeManager` contract as well as the `Staking` contract whose states and slashing logic are used here.
- Change the Arbitrator's address.
- Set the minimum required GRT deposit for disputes.
- Set the fisherman reward percentage.
- Set the slashing percentages.

The Arbitrator can:

- Accept, reject, or draw disputes.

High Severity

H-01 Upgrade Will Cause Active Disputes to Be Unresolvable

As part of the changeset, all disputes created after the upgrade will contain a `DisputeStatus`. Additionally, the `DisputeStatus` enum value will be used in the new `onlyPendingDispute` modifier to determine whether certain actions such as `acceptDispute`, `rejectDispute`, and `drawDispute` can be called on the passed-in `disputeID`. However, disputes that already exist do not have the `DisputeStatus` enum, and as such, looking up the value of the enum will return a default value of 0 (`Null` in `DisputeStatus`). Consequently, actions against existing disputes will fail as the `onlyPendingDispute` modifier will incorrectly determine that the dispute does not exist, and revert. This will prevent active disputes from being resolved. For indexer and query disputes, this will also cause the minimum deposit of 10,000 GRT to become stuck in the `DisputeManager` contract.

Adding logic for a two-step handoff to handle active disputes on-chain would be cumbersome and increase the overall complexity of the contract. Therefore, consider notifying fishermen of the impending upgrade, and ensuring that any disputes are resolved by the Arbitrator before upgrading.

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

> *Fishermen are generally anonymous and it is a permissionless role, so there is generally no way to notify them that the upgrade is happening, other than announcing through Discord channels - most likely participants that will be Fishermen would be Indexers, so we can notify in Indexer channels. However, this risk can be mitigated by being careful in the deployment plan: disputes have only occasionally happened, so we can make a point in the plan to wait for any active disputes' resolution before executing the upgrade transaction. We will ensure to document this in the deployment plan and have made a note in the pull request in the meantime.*

Notes & Additional Information

N-01 Using Implicit Imports

The `DisputeManager` contract uses the following implicit imports. This can make it difficult to determine exactly what is being used in the imports and can lead to namespace collisions.

Consider being explicit when importing by using named import syntax (e.g., `import {A, B} from "X.sol;") to increase the clarity of the codebase.

Update: *Acknowledged, not resolved. The Graph team stated:*

> *Noted, but since these imports were already in this form before the audited PR, we will leave them unchanged for the moment.*

N-02 Incomplete Tests

Only a subset of dispute tests were updated as part of this changeset. Consider updating all relevant tests, and adding new tests, to ensure that the new `DisputeStatus` enum is being set correctly for new disputes throughout the full dispute lifecycle.

Update: *Acknowledged, not resolved. The Graph team stated:*

> *Acknowledged, will note the test coverage as something to improve but keep it as it is for now.*

N-03 Unused Return Value

The boolean return value in the `_drawDisputeInConflict` function is unused. Consider removing the return value to improve the clarity of the codebase.

Update: *Acknowledged, not resolved. The Graph team stated:*

> *Acknowledged, but keeping it like this for now.*

Conclusions

One high-severity issue was identified during this audit. The issue stemmed from the complexity of upgrading the protocol's logic while correctly accounting for disputes that are in-progress.