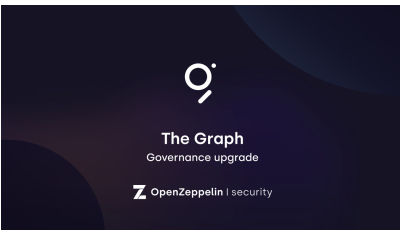
The Graph – Governance Upgrade Audit

APRIL 27, 2021 | IN SECURITY AUDITS | BY OPENZEPPELIN SECURITY



Introduction

The Graph team asked us to audit a new set of contracts that should enhance the existing governance system by enabling the storage of proposal outcomes and votes to better address traceability and provide a trustful source of information regarding protocol proposals.

The pull request that we have audited is the PR#362 at commit `d51553b3d70c6110885 2f0ffe1ed249254c91e9a` and the audited files are the following:

```
contracts/governance/GraphGovernance.sol
contracts/governance/GraphGovernanceStorage.sol
contracts/governance/IGraphGovernance.sol
```

Overview

The introduced changes are modular and consist of one main `GraphGovernance` contract.

This contract will be governed by The Graph multi-sig and will be upgradeable using the protocol upgradeability pattern. It exposes two functions, the `createProposal` function that gives the possibility to store a `Proposal` in a mapping, and an `updateProposal` function that also gives the chance to update an existing proposal.

Each proposal is stored as a struct and can be looked up by the IPFS hash of its content in a `bytes32` mapping. The struct data is composed of the Proposal's votes and resolution.

Summary

We are happy to see that the proposed code is clear and modular to enhance the protocol functionalities. We must note that the PR in question is still not merged; we assume that The Graph team will merge the code as it is and that no other bugs are introduced in later changes. Two auditors have audited the code over three days, with the findings presented below.

Update: *All of the following issues have been either fixed or acknowledged by the Graph team. Our analysis of the mitigations is limited to the specific changes made to cover the issues, and disregards all other unrelated changes in the codebase.*

Critical Severity

None.

High Severity

None.

Medium Severity

[M01] Lack of event emission after sensitive action

The `_initialize` function of the `Governed` contract does not emit the `NewOwnership` event after setting the value of the `governor` to be the `_initGovernor`.

Consider emitting events after sensitive changes occur to facilitate tracking and notify off-chain clients following the contracts' activity.

Update: *Fixed in PR462 at commit `1714b78d1243a824f36106539b34f8a79ebf14f3`.*

[M02] Proposal's update can assume prior states

The `updateProposal` function of the `GraphGovernance` contract is designed to update either the votes or the resolution of a proposal.

The function is not checking whether the new parameters for the proposal, passed as input values, are different from those stored in the `proposals` mapping, or even if they have been used previously.

Specifically, the `_votes` parameter can be reused multiple times. Even if `_votes` is an IPFS hash of a collection of signatures for each vote for the `_proposalId`, this doesn't amount to a replay vulnerability. However, the fact that the `updateProposal` function call is revisiting prior data may lead to confusion.

Even worse, a proposal's resolution can be changed from `Accepted` to `Rejected` and vice versa as many times as this function is called.

Whether this is a design choice or an unexpected outcome, consider either properly documenting this design choice or avoiding having a non-permanent or repetitive resolution on a specific proposal.

Moreover, consider adding some checks to verify that the values passed as input parameters are different from the stored or previously referenced data.

Update: *Fixed in PR463 at commit `8a4fb6e5aeaf77787925 8ef0b4bdacde23ae30d7` where documentation describing the consequences of this design choice were added.*

Low Severity

[L01] Lack of input validation

The `initialize` function of the `GraphGovernance` contract is not validating the input parameter passed in.

Consider adding proper checks to determine if the zero address is passed as an input parameter to avoid mistakenly setting the `governor` to a null address.

Update: *Fixed in PR463 at commit `f01518d1b669aeed89d3ceaa112b315cbdfb8f85`.*

[L02] Lack of docstrings

The `IGraphGovernance` interface, as well as the `ProposalCreated` and `ProposalUpdated` events, are lacking documentation in the form of docstrings or comments.

In the `GraphGovernance` contract, there is no documentation specifying which encoding or representation the parameters taking on IPFS hash values realize.

Consider thoroughly documenting all events and files in the codebase. When writing docstrings, consider following the Ethereum Natural Specification Format (NatSpec).

Update: *Fixed in PR463 at commit `e3cd5e35896b66ea6bbc794b874f98c462cf8cc21`.*

Notes & Additional Information

[N01] Lack of indexed parameters in events

The `ProposalCreated` and `ProposalUpdated` events of the `GraphGovernance` contract are lacking indexed parameters.

Consider indexing event parameters to avoid hindering the task of off-chain services searching and filtering for specific events.

Update: *Fixed in PR463 at commit `d7b25e5158f31b46e16daae043e1eafc7e715b989`.*

[N02] Useless event parameter

The `ProposalCreated` and `ProposalUpdated` events of the `GraphGovernance` contract are emitting the address of the `msg.sender` as the first parameter.

The functions emitting those events are only callable by the `governor` due to the `onlyGovernor` modifier. For this reason, there is no way that the `msg.sender` can be different at some point.

Since it can be known beforehand who the `msg.sender` is when emitting such events, consider removing this parameter from the event definitions and emissions.

Update: *Fixed in PR363 at commit `b1439ac9ca806234d3a4721d7594af8cf753c358`.*

Conclusions

2 Medium and other lower severity issues were found with changes recommended to improve the codebase.