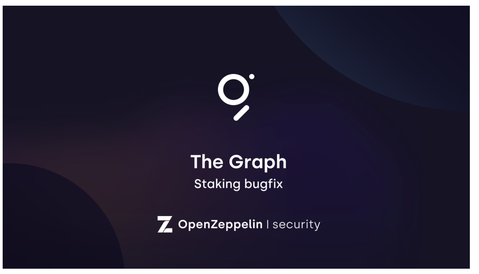
The Graph – Staking Bugfix #1 Audit

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



Introduction

The Graph team asked us to audit a bugfix of an issue regarding the `Staking` contract.

The pull request that we have audited is PR#457 at commit `01c891829d39e1d6adc30bf13a2c8bf64504f808` and the audited files are the following:

```
contracts/staking/Staking.sol
contracts/staking/libs/MathUtils.sol
contracts/staking/libs/Stakes.sol
```

Overview

The issue being fixed is that when indexers were used to call the `stakeTo` function (internally calling the `_stake` function) to stake from an external address, the delegation parameters of the delegation pool were incorrectly set. This was occurring because `setDelegationParameters` was always using `msg.sender` to initialize the delegation pool.

The solution adopted is to have an internal `_setDelegationParameters` that takes an address as input and that can be used either by the `stake` function or the `stakeTo` function without encountering the issue anymore.

Apart from the bugfix, the pull request introduces some minor changes in the style and format of the codebase. In particular:

- A new `MathUtils` library has been introduced providing the `diff` and `weightedAverage` functions.
- The `Stakes` library has been refactored, removing the `hasTokens` function and the `getLockingPeriod`. This last function has been replaced by the use of the `MathUtils` library.
- Two auxiliary functions, `_pushTokens` and `_pullTokens` have been introduced to better modularize the code base.

Summary

The bugfix is quite small and clear. However, other unrelated changes have been added to the same pull request. We strongly recommend using atomic pull requests for bugfix in order to avoid confusing future readers about the explicit purpose of the pull request. We must also notice that the PR in question is still not merged; we assume that The Graph team will merge it as it is and that no other bugs are introduced in eventual changes. The code has been audited by two auditors during the course of 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

None.

Low Severity

[L01] Lack of input validation

The `collect` function of the `Staking` contract does not validate whether the `_tokens` parameter is non-zero.

When the `_tokens` parameter is zero, the `collect` function can run without error, and emit its `AllocationCollected` event. This provides no useful feedback in the form of a revert message if the parameter was malformed by the client. Furthermore, the emission of the trivial `AllocationCollected` event may confuse off-chain services.

In the case that allowing `collect` to be called on zero `_tokens` was a design choice, consider properly documenting this in the code and other public-facing documentation. Otherwise, consider adding proper checks that the `_tokens` parameter is non-zero.

Update: *Fixed in commit `bd06a61e1055a5e0585e8ea64e618a8d6ce65d7c` where the `collect` function now includes documentation describing that zero values of `_tokens` are allowed.*

[L02] Unclean code

In the `lockTokens` function of the `Stakes` contract the `weightedAverage` function is called, but it is not clear what function parameters each of its four inputs denote. This is because as the inputs are passed in, their names are not suggestive of, or similar to, the parameters belonging to the function signature of the `weightedAverage` function.

This hinders readability and understanding of the code by auditors or other stakeholders.

Consider either documenting as comments in the code the correspondence of each input to parameter or defining and using intermediate variables with suggestive names as inputs.

Update: *Fixed in PR465 at commit `bead8f1e9f248764eec8f4ae5f627c86da33c78d`.*

Notes & Additional Information

[N01] Misleading naming

The `diff` function of the `MathUtils` library is performing a difference of the two numbers or returning zero if the result is negative.

Consider changing the name of the function to `diffOrZero` or some other name that can be more explicit on the purpose of the function.

Update: *Fixed in commit `3ef27446c7eef64590e133358119d2cbbb0cad65`.*

[N02] Erroneous docstrings

The private `_setDelegationParameters` function of the `Staking` contract is copy/pasting the docstrings from the above `setDelegationParameters` function.

Consider giving this private function appropriate docstrings or avoid duplicating them to improve code readability and clarity.

Update: *Fixed in commit `c8255c05f2dca0a4c7f1574c730255a8c0d39d8c`.*

[N03] Inconsistent style

The `_sendRewards` function of the `Staking` contract makes an ERC20 Graph Token transfer and checks its success with a require statement. Within this same pull request, the `_pushTokens` helper function was introduced, making the same transfer and checking its success. This helper function is used in all the `Staking` contract's functions, making a Graph Token transfer, except for the `_sendRewards` function.

Taking into consideration how much value a consistent coding style adds to the project's readability, enforcing a standard coding style is recommended. For consistency, consider using the `_pushTokens` function for the token transfer within the `_sendRewards` function.

Update: *Fixed in commit `e11d04183d98f06276cd3fd38c4ad90c4cfd9b65`.*

Conclusions

2 Low severity issues and other notes have been reported with recommended changes to improve the codebase.