

The Graph - PR-491, 475, 492

Date	August 2021
Auditors	David Oz Kashi, Martin Ortner

1 Executive Summary

This report presents the results of our engagement with **the-graph** to review **PR-491, PR-457, and PR-492**.

The review was conducted over the course of 5 days, from **16 Aug 2021** to **20 Aug 2021**. A total of 2x5 person-days were spent.

2 Scope

Our review focused on the following pull requests:

- [PR-491: ariel/delegation-no-shares](#) at (`059f634b8b31ca97bbe565bb959c26322a97d2d6`)
- [PR-475: ariel/multicall](#) at (`2e5921a61943c27ca15ce1ebcdacbfd11f613966`)
- [PR-492: ariel/delegate-thawing](#) at (`dedfa31cd4200fa3fc83afba954f1b4cbadf022c`)

2.1 Objectives

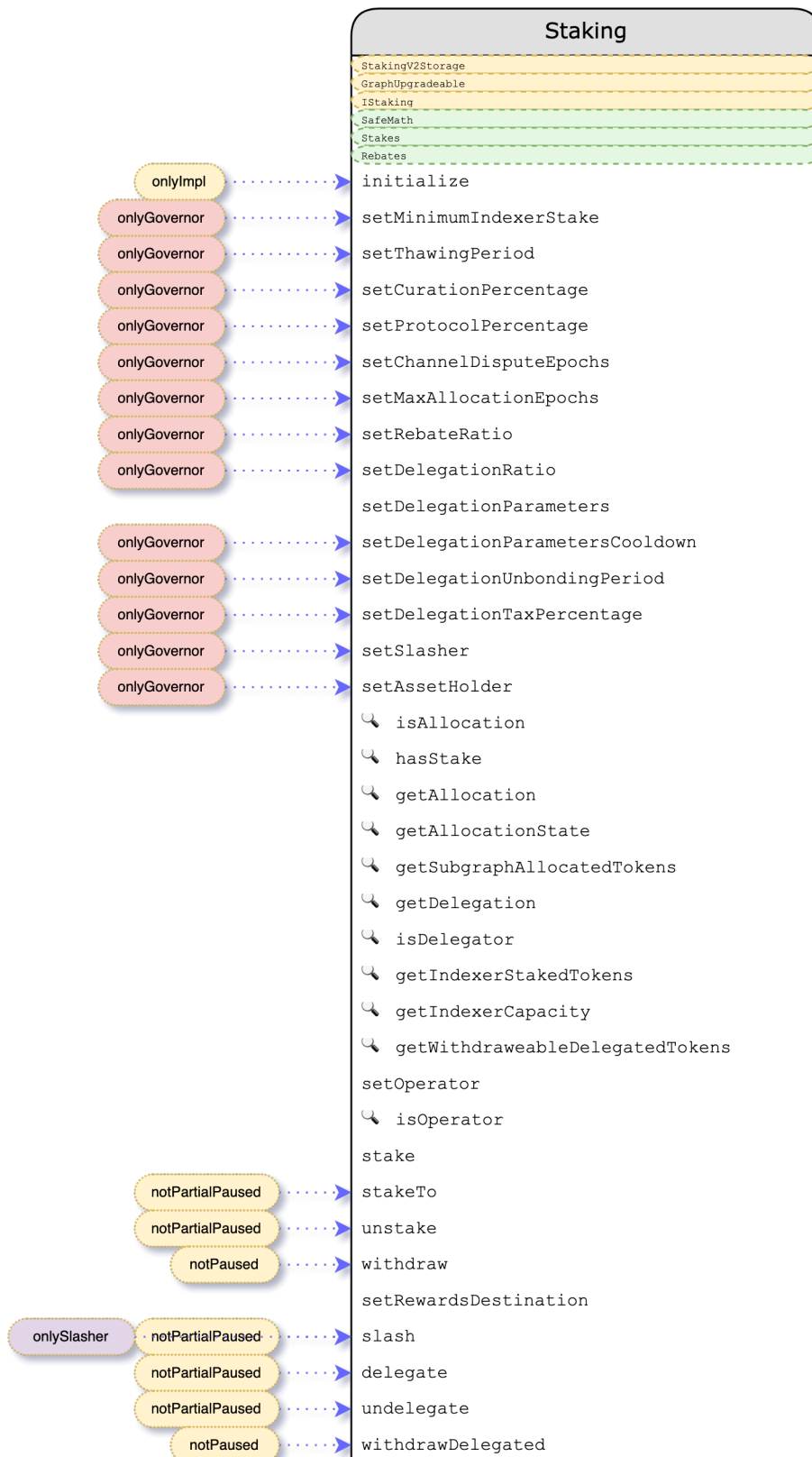
Together with the client, we identified the following priorities for our review:

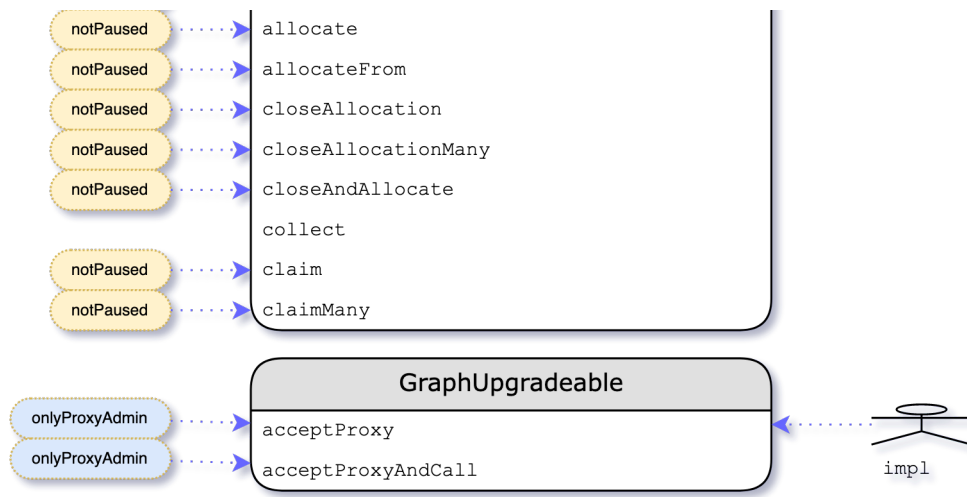
1. Ensure that the system is implemented consistently with the intended functionality, and without unintended edge cases.

2. Identify known vulnerabilities particular to smart contract systems, as outlined in our [Smart Contract Best Practices](#), and the [Smart Contract Weakness Classification Registry](#).

3 PR-Review

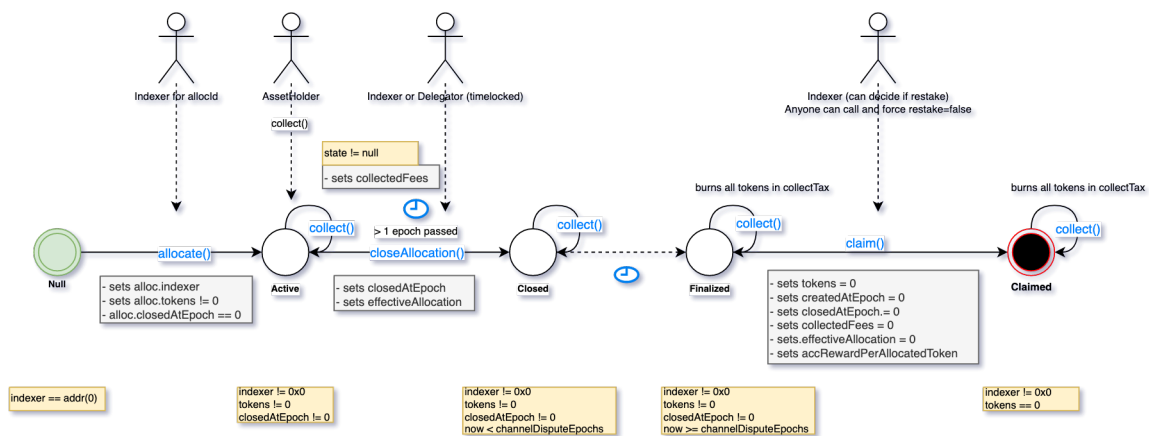
The pull-requests in scope are affecting the the-graph Staking contract. This section provides an outline of the public-reachable interface of the contract.





Staking - Contract Overview

The staking contract manages a data structure to track allocations and their state. The following image provides an overview of the allocation states and state transitions.



Staking - Allocation State Diagram

3.1 PR491 - Revert when not enough precision to assign a delegation share

This PR makes sure the `delegate` method reverts if no shares are minted for the provided amount of tokens.

- Informational** A call to `_delegate` might now fail if it would result in zero shares being issued due to lack of precision. This slightly changes the behavior of the function. This should be unproblematic since the function is expected to throw on error conditions. However, the fact that the behavior was altered should be made transparent to notify external callers. Internally, `_delegate` is pot. called by `_withdrawDelegated` (reachable

via `withdrawDelegated`) which might now throw for small token amounts that result in zero shares.

3.2 PR-475 - Add multicall to staking contract

The PR removes the `many*` prefixed helper-functions in favor of a more general approach using the sushi-swap `MultiCall` pattern.

- **Informational** `multicall()` will only break on `reverts` . If a function signals error condition via e.g. bool return values the loop will continue, skipping this error. However, this does not seem to be the case for `Staking` which is generally designed to throw on error conditions.
- **Informational** It should be noted that `multicall()` can also be used by `Governor` (batched proposals?) or `ProxyAdmin` (and theoretically `implementation()`).
- **Informational** Even though more generic this method might be less gas efficient (public calls, checks run multiple times).
- `MultiCall` is logically almost identical to [uniswap/Multicall.sol](https://github.com/Uniswap/multicall.sol) with the difference that `multicall()` is not `payable` .
- The `MultiCall` implementation is **not** payable (preventing pot. misuse of `msg.value` in the `MultiCall` loop which could mess up accounting).
- Adds `MultiCall` to inherited contracts of `Staking` . Does not change the storage layout.

3.3 PR-492 - Average delegation unbonding on multiple requests

Before this change, calling `undelegate` while tokens are already locked for undelegation would reset the unlock time to the total locking time specified for the contract. This PR changes this behavior to use a token-time-average locktime instead.

- **Medium** Tokens can be undelegated “for free” due to low precision. Consider the following example:
 - A users `undelegates` a large amount of tokens and waits until there is one day remaining for the tokens to be withdrawable: e.g.


```
tokensLocked=80000; timeRemaining=1
```
 - The user now `undelegates` another batch of shares/tokens, but only barely as much such as the token-time-weighted average truncated to a precision of 1 integer decimal would return the same locking

time as before (equivalent to `timeRemaining=1`). With `tokensLocked=80000` previously and `timeRemaining=1` averaged with `_tokens=297` and `newLockTime=28` (full lock time is 28 epochs) this would mean with this call another 297 tokens could be undelegated without having an impact on the overall lock time. This can of course be performed multiple times, both increasing the total `tokensLocked` which would also allow to undelegate more tokens with every call, w/o increasing the locking time.

- **Recommendation:** Consider increasing the precision for the math operation from 1 to 4 decimals. This would mitigate the effect by allowing less “error” and therefore less tokens to be undelegate w/o increasing the locking time and effectively render this method economically infeasible. It should be noted that GRT token is already 18 decimals and lifting it could theoretically although unlikely cause the method to throw in SafeMath for very large token amounts. Another option is to programatically ensure that with every call to `undelegate` at least one epoch is added to the locktime (further ensuring that the total locktime cannot be greater as the default lock time).
- **Minor** “Whales” benefit from the token weight averaging effect.
 - The averaging effect creates an imbalance between large cap and small cap token holders as every next weight-averaged undelegation is averaged to the initial undelegation. This means that a “whale” that initially undelegates a large amount of tokens, then waits until `timeRemaining=1`, can get a higher token/day undelegation ration when averaging than a user that initially only undelegated as small amount.
 - Note that a “whale” can strategically optimize to keep some tokens delegated (earning) and only undelegate them close to `timeRemaining` going to zero.
- **Minor** Consider using `private const PRECISION = 10` instead of a hardcoded integer literal 10 (will be replaced by the compiler/preprocessor; double-check that this does not change the storage layout)
- **Informational** The CI-Unittests are failing for this PR ([Unittest](#))

Appendix 1 - Disclosure

ConsenSys Diligence (“CD”) typically receives compensation from one or more clients (the “Clients”) for performing the analysis contained in these reports (the “Reports”). The Reports may be distributed through other means, including via ConsenSys publications and other distributions.

The Reports are not an endorsement or indictment of any particular project or team, and the Reports do not guarantee the security of any particular project. This Report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. No Report provides any warranty or representation to any Third-Party in any respect, including regarding the bugfree nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on the Reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this Report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. CD owes no duty to any Third-Party by virtue of publishing these Reports.

PURPOSE OF REPORTS The Reports and the analysis described therein are created solely for Clients and published with their consent. The scope of our review is limited to a review of code and only the code we note as being within the scope of our review within this report. Any Solidity code itself presents unique and unquantifiable risks as the Solidity language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond specified code that could present security risks. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. In some instances, we may perform penetration testing or infrastructure assessments depending on the scope of the particular engagement.

CD makes the Reports available to parties other than the Clients (i.e., “third parties”) – on its website. CD hopes that by making these analyses publicly available, it can help the blockchain ecosystem develop technical best practices in this rapidly evolving area of innovation.

LINKS TO OTHER WEB SITES FROM THIS WEB SITE You may, through hypertext or other computer links, gain access to web sites operated by persons other than ConsenSys and CD. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that ConsenSys and CD are not responsible for the content or operation of such Web sites, and that ConsenSys and CD shall have no liability to you or any other person or entity for the use of third party Web sites. Except as described below, a hyperlink from this web Site to another web site does not imply or mean that ConsenSys and CD endorses the content on that Web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the Reports. ConsenSys and CD assumes no responsibility for the use of third party software on the Web Site and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

TIMELINESS OF CONTENT The content contained in the Reports is current as of the date appearing on the Report and is subject to change without notice. Unless indicated otherwise, by ConsenSys and CD.



Request a Security Review Today

Get in touch with our team to request a quote for a smart contract audit.

[CONTACT US](#)

[AUDITS](#)

[FUZZING](#)

[SCRIBBLE](#)

[BLOG](#)

**Subscribe to Our
Newsletter**

STAY UP-TO-DATE

Stay up-to-date on our latest offerings, tools, and the world of blockchain security.

TOOLS

RESEARCH


ABOUT

CONTACT

CAREERS

PRIVACY POLICY

POWERED BY



CONSENSYS