The Graph

GNS updates

OpenZeppelin | security

The Graph team asked us to review some incremental improvements to their protocol. We looked at the code and now publish our results.

Scope

We audited two pull requests in the core contracts repository:

- Pull Request 485 up to commit `94b84643`.
- Pull Request 486 up to commit `3ae5d6c9`.

The scope includes all modifications to Solidity files within these pull requests. We also reviewed related parts of the existing code base to the extent that additional context was necessary to understand the modifications under audit. All other code and contract dependencies were assumed to work as documented.

System Overview

The Graph Name System (GNS) provides a mechanism for translating subgraph names into subgraph versions, so the details of a subgraph can change while the human-readable name remains stable. It also incentivizes a curation mechanism where users can signal their belief in the importance of a subgraph by exchanging Graph Tokens for subgraph-specific credits.

The modifications we reviewed included adding *multicall* functionality to the GNS contract so that multiple operations can be executed atomically. A motivating example is to allow subgraph owners to signal their subgraph on deployment, without the possibility of their signal transaction being front-run. We also reviewed a refactoring operation that simplifies the internal accounting.

Privileged Roles and Trust Assumptions

The GNS contract has an administrator role that can upgrade the functionality arbitrarily. Therefore, users must rely on the governance structure to use this power fairly and wisely.

Here we present our findings.

Critical Severity

None.

High Severity

None.

Medium Severity

[M01] GNS update edge cases

Pull Request 486 is intended to replace complex bonding curve calculations with a simple expression for the particular case that is actually used. This should be a pure refactor, where the functionality remains the same. However, there are some discrepancies that should be noted:

- In the `vSignalToNSignal` function
  - the original code would revert if the supply is zero, while the new code returns zero.
- In the `nSignalToVSignal` function
  - the original code would revert with "invalid parameters" if the supply is zero, while the new code attempts to divide by zero, which would produce a generic `SafeMath` revert.
  - the original code would revert if the reserve balance is zero, while the new code returns zero.
  - the original code would revert if the amount to sell exceeds the supply, while the new code simply returns more than the vSignal.
  - the original code returns the entire reserve balance if the sell amount matches the supply. The new code should calculate the same value, unless the initial multiplication overflows.

Consider validating that the new behavior is acceptable. Additionally, consider expanding the coverage of unit tests to ensure they cover all edge cases.

Low Severity

None.

Notes & Additional Information

[N01] Bypassing zero-transfer events

Pull request 485 is intended to support multi-call transactions. It should be noted that it also introduces a side-effect in the `GNS` contract. Instead of executing direct ERC20 transfers, the contract now uses the functions in the `TokenUtils` library. This library bypasses zero-value transfers, and therefore does not emit the associated `Transfer` events. We simply note this because it's an unexpected side-effect and it introduces the possibility that some GNS events are no longer matched with a corresponding `Transfer` event.

[N02] Inconsistent Filenames

The `Multicall` contract and its interface are defined in the *MultiCall.sol* and *IMultiCall.sol* files respectively. Consider ensuring the file names match the contract and interface names, including matching capitalization.

Conclusions

No critical or high severity issues have been found. Recommendations and fixes have been proposed to increase code quality, improve readability, and minimize errors.