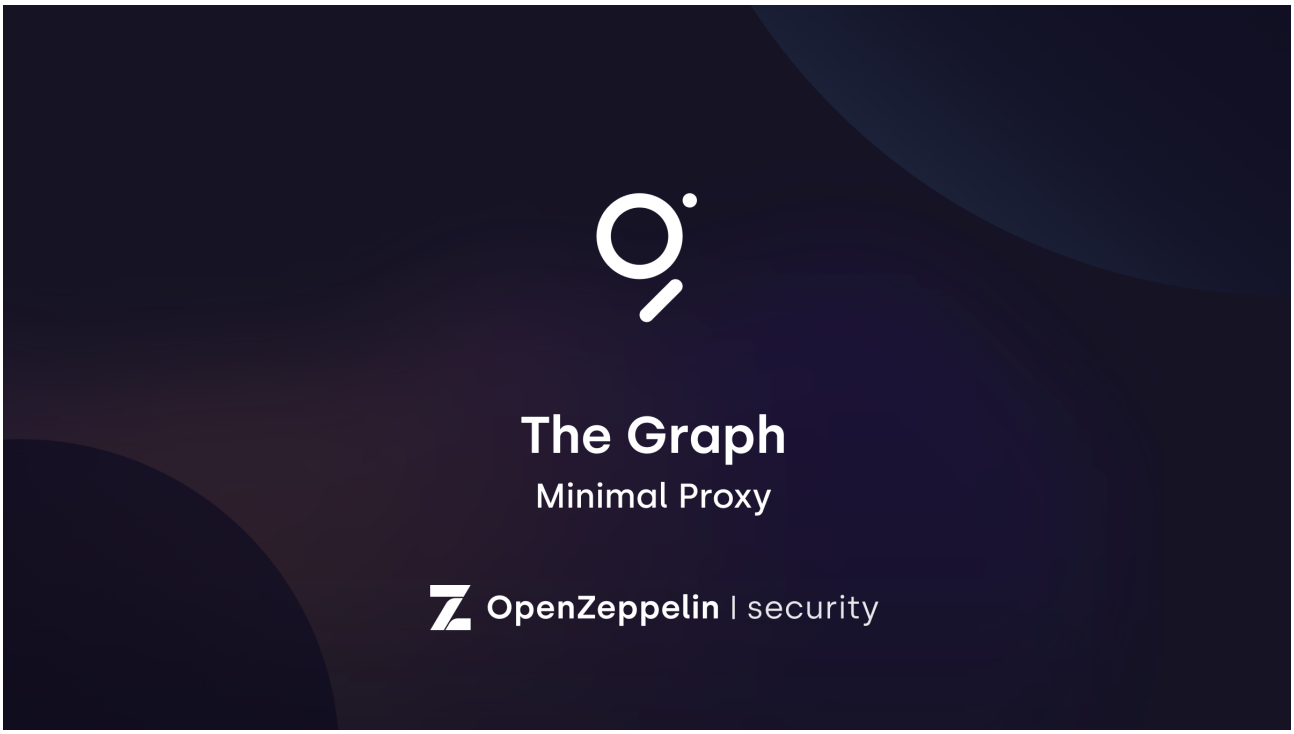

The Graph team asked us to review their gas optimizations for deploying ERC-20 tokens. We looked at the code and now publish our results.

Scope

We audited Pull Request 505 up to commit `1a2a89b56ad04658655a25f1c0361b4bfd97c91f`.

The scope includes all modifications to Solidity files within these commits. 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

When signaling on a subgraph deployment, a subgraph-specific ERC-20 token is minted to represent the curation "shares" owned by a user. The first time one of these tokens is minted, it is necessary to deploy a new token contract. Previously, the complete bytecode that contained the ERC-20 implementation was deployed. The pull request we reviewed attempts to make this process more efficient by using a "Minimal Proxy Contract", provided by OpenZeppelin's Clones library.

Trust Assumptions

The base implementation contract used by token proxies is set upon initialization of the `Curation` contract. It is assumed that this base implementation behaves correctly and non-maliciously.

Here we present our findings.

Critical Severity

None.

High Severity

None.

Medium Severity

None.

Low Severity

None.

Notes & Additional Information

[N01] Ambiguous revert reasons

The `mint` function now uses `pullTokens` from the `TokenUtils` library, which replaces a `require` statement in the same function on the current dev branch (via commit 6962037). In each instance, the `transferFrom` function is checked for success. However, the revert message from the `TokenUtils` library is ambiguous, simply stating "!transfer", whereas the revert message currently on the dev branch states "Cannot transfer tokens to deposit".

Similarly, the `burn` function now uses `pushTokens` from the `TokenUtils` library, which returns the same ambiguous revert message, "!transfer", as opposed to the revert message from the dev branch that states "Error sending curator tokens".

Consider revising the revert reasons for `pushTokens` and `pullTokens` to differentiate between them and add detail to facilitate debugging failed transactions.

Update: *Acknowledged. The team prefers to continue using the `TokenUtils` library as it is also used by other contracts and keeps the revert reason short.*

[N02] Not emitting an event when burning `0` tokens

The `burn` function of the `Curation` contract now uses `TokenUtils.pushTokens` instead of a regular ERC20 `transfer`. The `pushTokens` function only performs a transfer if the amount being transferred is not `0`. This means that when the `tokensOut` parameter is `0`, the old implementation would call the token's `transfer` function, potentially emitting a `Transfer` event, while the new implementation will never emit a `Transfer` event.

An external client that relies on event emissions could present unexpected behavior as a result of the change. In such case, consider updating external clients accordingly.

Update: *Acknowledged. The team said that the it is ok to not emit a `Transfer` event when burning `0` tokens, as no client is relying on this condition.*

Conclusions

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