The Graph — PR Review Dec 2021 [PR-527 only]

Date	December 2021
Auditors	Heiko Fisch, Martin Ortner

1 Executive Summary

This report presents the results specific to **PR-527** of our engagement with **The Graph** to review

- PR-526 (**under review by the client**) and
- **PR-527** (**this document**)

in their contracts repository.

The results were initially presented to the client in December 2021 in the form of a report that combines our findings for PR-526 and PR-527. This report is still under review by the client and not yet publicly available. In January 2022, the client requested an individual report to be created for findings related to PR-527 only. **This document** is the result of splitting off the issues related to PR-527 from the initially delivered combined report. In order to preserve the audit consistency, timeline and provide full transparency, references to PR-526 in sections other than “Findings” and “Recommendations” have not been altered.

We conducted this assessment over two weeks, from **December 6–17, 2021**, and allocated 4 person-weeks.

A previous round of reviews on different PRs was carried out in October 2021. The review notes can be found here. Please note that some important findings mentioned in these notes have not been addressed yet by the client:

- Selling subgraph NFTs on an exchange requires special care as the owner does not have to lock the NFT before putting it up for auction. That means the sale might happen under conditions the buyer didn’t agree on. (This is a general issue with NFT auctions.)
- A potential griefing Denial-of-Service (DoS) by front-running `GNS.publishNewVersion(..., _subgraphDeploymentID, ...)` on `curation`.
- Risk of being sandwiched when actions are performed with slippage control effectively disabled (as in PR-526).

In the first week of the current engagement, we focused on PR-526, which introduces a change to the `GNS` contract that splits the signal for a subgraph in half between two deployments to allow a smoother transition to a new version. We identified several issues of various severities that can be fixed easily; some of them have already been addressed in later commits on the PR while our assessment was still ongoing. Notably, though, we also observed that the mathematical assumption underlying this PR is flawed and that an attacker could exploit this to steal funds from the contract. This is a more fundamental issue and might make it necessary to reconsider the feature/PR in its entirety.

The second week’s efforts were split approximately in half between the two PRs, with one auditor focusing on the aforementioned issue and discussing with the client a proof-of-concept exploit and the second reviewing PR-527, which refactors the `GNS` contract’s NFT functionality to be implemented via composition instead of inheritance to avoid problems with the latter approach that surfaced on testnet deployments. We found two issues in this PR (3.1 and 3.2).

2 Scope

Our review focused on the following Pull-Requests:

- (PR-526 @ 0acb399cbb696ac5acb86e9fa277e13abc403a29) (not part of this report)
- PR-527 @ 760a7eaae745432053911c915cc2ca5e7a5d5717

2.1 Document Change Log

Version	Date	Description
1.0	2021-12-14	Delivered initial report
1.1	2022-02-03	Updated Issues: Preliminary Review of 3.1 / 3.2

3 Findings

Each issue has an assigned severity:

- Minor issues are subjective in nature. They are typically suggestions around best practices or readability. Code maintainers should use their own judgment as to whether to address such issues.
- Medium issues are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.
- Major issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- Critical issues are directly exploitable security vulnerabilities that need to be fixed.

3.1 [PR-527] `tokenURI` – produces an invalid URI if subgraph metadata is set Major ✓ Preliminary Review OK

Resolution

Preliminary Review requested by the client: It is recommended to thoroughly review this changeset with the next auditing round.

The issue was addressed with graphprotocol/contracts@ `0a3888d` . The client provided the following statement regarding the changes:

- You can see that I store the metadata as bytes32 instead of string and then use a base58 encoder to show the proper IPFS format.
- Also added a few missing natspec.

Preliminary review (Diligence):

The code was rewritten and is now storing a `bytes32 metadatahash` . Events emit `bytes32` instead of `string` addressing the potential exception in off-chain library components. If a `tokenDescriptor` contract is set the `bytes32` metadata will be converted to `base58` (we haven't checked if this may revert in some edge cases and we suggest investigating this; we treated the `base58` lib and `toString` methods as black-boxes for this quick-review). If no `tokenDescriptor` contract is set the `metadatahash` will be converted to `uint` and subsequently to a `hexString` representation. It is suggested to keep the original name for `toString` -> `toHexString` as taken from the original source code. It should also be noted that string conversions can be cumbersome and gas intensive when performed on-chain (solidity) and it is generally recommended to perform them off-chain (unfortunately a `string tokenURI` is part of the ERC-721 metadata extension).

All in all, this looks like a viable solution to go forward with. The main risks from the changeset are `tokenURI` construction failures (i.e. forced `metadatahash` conversion issues; we have not investigated for such cases). However, these cases are prime candidates usually be covered with an extensive unit-test suite (highly recommended).

Description

Assuming that `_subgraphMetadata` is the raw bytes32 format of an IPFS hash, it will certainly contain byte sequences that belong to the non-printable class of ASCII characters. Concatenating this byte sequence to a common URI string will render the resulting `tokenURI` invalid unless the hash is properly converted to its integer- or hex-string representation. Casting a byte sequence to `string` in Solidity will not automatically convert it to an ASCII-compatible hex-string that can be appended to an URI.

```
 »  bytes32 ipfsHash = 0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470
 »  string(abi.encodePacked(ipfsHash))

Exception: JS-Decoding-Error: invalid codepoint at offset 1; missing continuation byte (argument="bytes", value=Uint8Array(0xc5d

reason: 'invalid codepoint at offset 1; missing continuation byte',
  code: 'INVALID_ARGUMENT',
  argument: 'bytes',
  value: Uint8Array(32) [
    197, 210,  70,   1, 134, 247,  35,  60,
    146, 126, 125, 178, 220, 199,   3, 192,
    229,   0, 182,  83, 202, 130,  39,  59,
    123, 250, 216,   4,  93, 133, 164, 112
  ],
  baseType: 'string',
  type: 'string'
```

As illustrated in this example, depending on the server/client-side implementation of off-chain components, a library might choose to throw an exception when trying to decode an incompatible string argument or output in transactions or events (certain JavaScript libraries, Remix, Ganache CLI). This can be unexpected and, therefore, potentially impact business processes tied to the affected component.

Examples

- `updateSubgraphMetadata` – takes a raw `bytes32` IPFS hash (likely non-printable).

code/pr527_760a7ea/contracts/discovery/GNS.sol:L229-L240

```
/**
 * @dev Allows a subgraph owner to update the metadata of a subgraph they have published
 * @param _subgraphID Subgraph ID
 * @param _subgraphMetadata IPFS hash for the subgraph metadata
 */
function updateSubgraphMetadata(uint256 _subgraphID, bytes32 _subgraphMetadata)
    public
    override
    onlySubgraphAuth(_subgraphID)
{
    _updateSubgraphMetadata(_subgraphID, _subgraphMetadata);
}
```

- `_updateSubgraphMetadata` – takes a raw `bytes32` IPFS hash (likely non-printable) and casts it to `string` before passing it to `_setSubgraphURI` . The string variable contains raw bytes that are likely non-printable.

code/pr527_760a7ea/contracts/discovery/GNS.sol:L242-L250

```
/**
 * @dev Internal: Allows a subgraph owner to update the metadata of a subgraph they have published
 * @param _subgraphID Subgraph ID
 * @param _subgraphMetadata IPFS hash for the subgraph metadata
 */
function _updateSubgraphMetadata(uint256 _subgraphID, bytes32 _subgraphMetadata) internal {
    _setSubgraphURI(_subgraphID, string(abi.encodePacked(_subgraphMetadata)));
    emit SubgraphMetadataUpdated(_subgraphID, _subgraphMetadata);
}
```

- `string memory _subgraphURI` – is `bytes32` converted to `bytes[]` interpreted as `string` (likely non-printable).

code/pr527_760a7ea/contracts/discovery/GNS.sol:L804-L806

```
function _setSubgraphURI(uint256 _tokenID, string memory _subgraphURI) internal {
    subgraphNFT.setSubgraphURI(_tokenID, _subgraphURI);
}
```

- `setSubgraphURI` – stores the (likely non-printable) `_subgraphURI` in a mapping. It is still raw bytes.

code/pr527_760a7ea/contracts/discovery/SubgraphNFT.sol:L118-L126

```
function setSubgraphURI(uint256 _tokenId, string memory _subgraphURI)
    external
    override
    onlyMinter
{
    require(_exists(_tokenId), "ERC721Metadata: URI set of nonexistent token");
    _subgraphURIs[_tokenId] = _subgraphURI;
    emit SubgraphURIUpdated(_tokenId, _subgraphURI);
}
```

- If `tokenURI()` is called, and `_subgraphURI` is set, it will return the URI (must be all printables or else no one can actually query that URL) by concatenating `baseURI` and `subgraphURI`. Since `_subgraphURI` is a raw byte sequence interpreted as `string`, it may very well contain non-printable characters and, therefore, produce an invalid URI.

code/pr527_760a7ea/contracts/discovery/SubgraphNFT.sol:L146-L156

```
string memory _subgraphURI = _subgraphURIs[_tokenId];
string memory base = baseURI();

// If there is no base URI, return the token URI.
if (bytes(base).length == 0) {
    return _subgraphURI;
}
// If both are set, concatenate the baseURI and tokenURI (via abi.encodePacked).
if (bytes(_subgraphURI).length > 0) {
    return string(abi.encodePacked(base, _subgraphURI));
}
```

Recommendation

Consider converting the `bytes32` raw IPFS hash to `uint256` (`_subgraphURI`) and convert it to `subgraphURI.toString()` (or `toHexString`) before concatenating it with the `baseURI`. Note that storing a proper string representation in `_subgraphURIs` may make it less likely that this issue re-surfaces with an improper implementation in `SubgraphNFTDescriptor`. Consider adding unit-test cases to avoid regressions.

3.2 [PR-527] `setSubgraphNFT` – changing subgraphNFT after first NFT was minted may break functionality for users Medium Acknowledged

Resolution

Acknowledged by the client and addressed with graphprotocol/contracts@`ceab729` by adding a warning to the natspec documentation of the affected function.

Description

`setSubgraphNFT` allows the Governor to change the linked subgraph ERC721/NFT address. The method can be called safely the first time the contract is deployed with the added NFT functionality. However, changing the contract address after users are already using it (NFTs were issued) may break a lot of functionality. For example, without a proper migration path that migrates all minted tokens from the previous to the new contract (which is not an easy task), users previously owning a subgraph NFT will lose subgraph ownership in GNS (`onlyOwner` now refers to the new contract), `deprecateSubgraph` will fail due to `onlySubgraphAuth` and `_burnNFT`.

The Governor can easily recover from this by changing the address back to the original NFT contract. This requires a majority vote, though.

Examples

code/pr527_760a7ea/contracts/discovery/GNS.sol:L189-L195

```
/**
 * @dev Set the NFT registry contract
 * @param _subgraphNFT Address of the ERC721 contract
 */
function setSubgraphNFT(address _subgraphNFT) public onlyGovernor {
    _setSubgraphNFT(_subgraphNFT);
}
```

Recommendation

It is unlikely that this will happen as a majority vote through Governor is required. Nevertheless, the method should either be removed because it will mess up ownership, or it should be explicitly clear that this method may impair how users interact with the GNS.

Consider removing the possibility to change `setSubgraphNFT` after initialization/migration to signal that the token contract is immutably linked to the GNS contract (assurance towards users that they keep ownership of their subgraphs), or else plan for a clean migration path when upgrading the `subgraphNFT` address. Consider adding development security notes/comments to `setSubgraphNFT`, or prefix the method to signal that calling this method requires special attention.

4 Recommendations

4.1 [Out of scope] Avoid unnecessary double cast

Description

`__SubgraphNFT_init` explicitly casts `address _tokenDescriptor` to `address` again.

Examples

code/0acb399/contracts/base/SubgraphNFT.sol:L21-L24

```
function __SubgraphNFT_init(address _tokenDescriptor) internal initializer {
    __ERC721_init("Subgraph", "SG");
    _setTokenDescriptor(address(_tokenDescriptor));
}
```

Appendix 1 - Disclosure