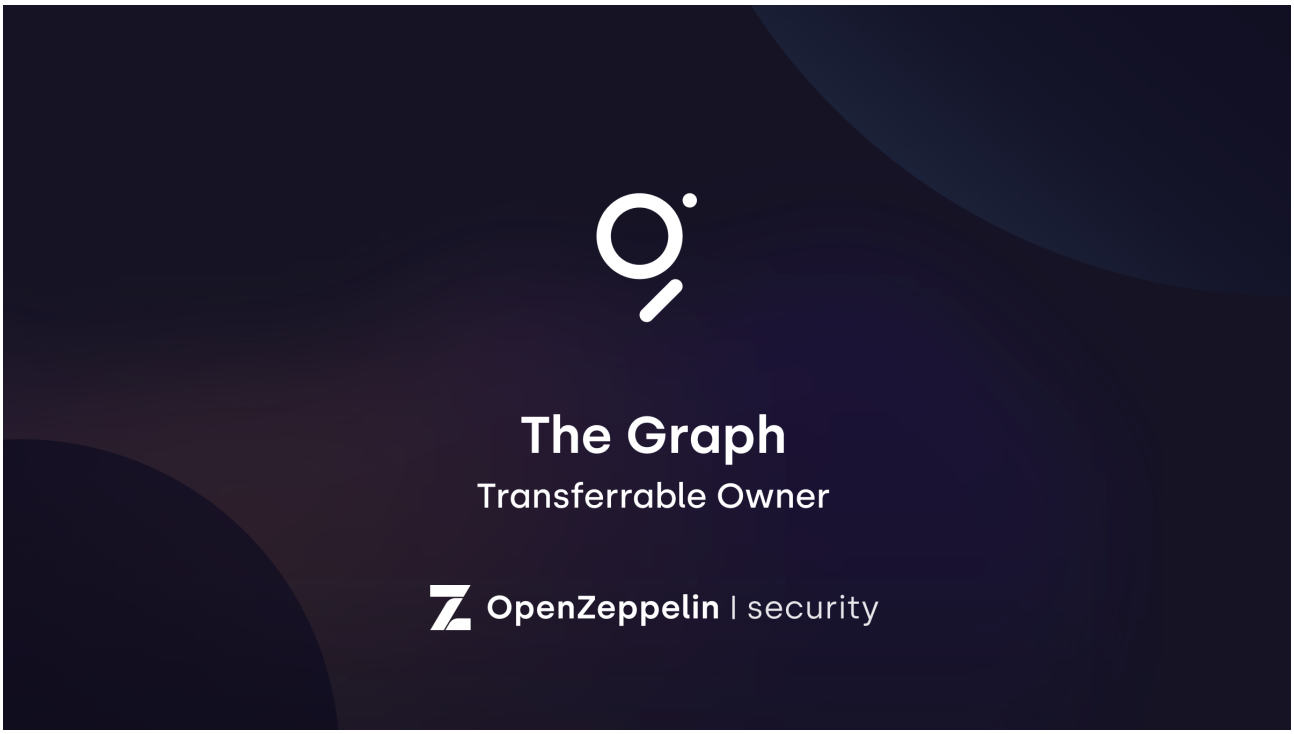

The Graph team asked us to review their new `GNS` access control implementation. We looked at the code and now publish our results.

Scope

We audited Pull Request 497 up to commit `396d06b120e7d7a528ce053fc28313eff617a128`.

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

The Graph Name Service (GNS) is an on-chain registry of subgraphs which decouples the subgraph identifiers from the actual subgraph deployment data. Previously, each subgraph was owned and controlled by a specific address stored in a separate EIP-1056 registry. The pull request we reviewed changes this access control implementation by transforming the `GNS` contract into a Non-Fungible Token (NFT), so that the control over a specific subgraph is granted to the owner of the corresponding NFT identifier.

Privileged roles and trust assumptions

The `GNS` contract inherits from the `GraphUpgradeable` contract, which enables upgrading the `GNS` implementation through the use of a proxy pattern. This upgradeability mechanism is fully controlled by the "proxy admin", which is assumed to behave non-maliciously.

Also, the ownership of a subgraph is now represented by the ownership of an NFT. The owner of subgraph's NFT and the addresses to which said NFT has been approved have the power to:

- Update the subgraph's metadata.
- Publish a new version of the subgraph.
- Deprecate a subgraph (and enable the withdrawal of GRT tokens)

Client-reported findings

During the audit, the client reported one vulnerability. Here we present the client-reported issue, followed by our findings.

Subgraph deployment not updated when there is no signal (client-reported)

The SubgraphDeploymentID for the Subgraph was not being updated in the case the Subgraph had zero signal. This fix updates the target deployment even if the Subgraph has never minted signal.

Update: *Fixed in commit `3482568292855cf1c54fe6b509d2289c36a0cc9b` of PR523.*

Critical Severity

None.

High Severity

None.

Medium Severity

[M01] Semantic overloading of approvals for access control

The `onlySubgraphAuth` modifier is used to restrict access to certain functions from the `GNS` contract. These functions can only be called by the owner of a subgraph NFT or by the accounts to which the NFT has been approved through the `approve` and the `setApprovalForAll` functions defined by EIP-721. Some of the functions that use the `onlySubgraphAuth` modifier include: * updateSubgraphMetadata * publishNewVersion * deprecateSubgraph

It is common within the NFT ecosystem to hand out token approvals to other accounts, such as NFT marketplace contracts. This could lead to unexpected security issues because there is no easy way for users to verify that approved accounts will not call `GNS` functions. Thus, a malicious actor with partial control over an approved account might be able to manipulate a subgraph's `GNS` data by indirectly calling one of the `onlySubgraphAuth` functions.

The `onlySubgraphAuth` modifier semantically overloads the ERC-721 approval mechanism, as token approvals are understood to only allow a third party to transfer a specific token with no additional token functionality delegated. Granting additional control over certain aspects of a protocol to approved accounts might lead users to encounter unexpected behavior, as they will likely be accustomed to stricter approval semantics that only deal with token transfers.

To decrease the chance of unexpected security incidents and avoid semantic overload, consider only allowing the owner of a subgraph NFT to have access to call `GNS` functions for that subgraph.

Update: *Fixed in commit `c66614ade263c1e947ad7d0bd986b0627660dfb9` of PR519.*

Low Severity

[L01] Misleading comments

The following misleading comments were identified:

- A comment for the `deprecateSubgraph` function states that this can "only be done by the subgraph owner", however the function uses the `onlySubgraphAuth` modifier, and is thus callable by the subgraph owner and any approved address.
- A comment describing the `MAX_PPM` variable, used for tax calculations, states it is the denominated in "parts per million", however the comments for the `_setOwnerTaxPercentage` and `setOwnerTaxPercentages` functions claim the values are denominated in "parts per hundred".

Consider revising the above comments to improve consistency and clarify implemented access controls.

Update: *Fixed in commit `c152f484b2005fe1cd2d4a37a889bd0d405fd6af` of PR518.*

Notes & Additional Information

[N01] Lack of input validation

The `nSignalToTokens` function from the `GNS` contract calculates the amount of tokens returned for signaling a particular subgraph. The subgraph data is first stored in the `subgraphData` variable using the `_getSubgraphData` function. Then, the `_nSignalIn` parameter is converted to `vSignal` using the `nSignalToVSignal` function. Finally, that `vSignal` amount is used in the call to the curation function `signalToTokens`. This `signalToTokens` function requires that the subgraph must be deployed and curated in order to perform calculations.

In the case that a subgraph is not yet deployed, consider failing early and loudly by using the `_getSubgraphOrRevert` function instead of using `_getSubgraphData`.

Update: *Fixed in commit `321c29b98ce1f1b6495da8e6ca4832a25cd6c154` of PR522.*

[N02] TODOs in codebase

The `tokenURI` function contains "TODO" comments that should be tracked in the project's issues backlog.

During development, having well described "TODO" comments will make the process of tracking and solving them easier. Without that information, these comments might tend to rot and important information for the security of the system might be forgotten by the time it is released to production.

These TODO comments should at least have a brief description of the task pending to do, and a link to the corresponding issue in the project repository.

Consider updating the TODO comments to add this information. For completeness and traceability, a signature and a timestamp can also be added.

Update: *Acknowledged. The team said that the NFTDescriptor functionality is still under design.*

[N03] Typographical and grammatical errors

The following typographical and grammatical errors were identified:

- "Only can set your own name" should be "Only you can set your own name".
- "asume" should be "assume".

Consider correcting the above typographical and grammatical errors.

Update: *Fixed in commit `6ec4065216c90b7710dffce7efa4da62760fbabb` of PR520.*

[N04] Unnecessary type conversion

The `_buildSubgraphID` function unnecessarily converts the `bytes32` result of a `keccak256` hashing operation into a `uint256`, whereas most other IDs in the GNS contract are of type `bytes32` (e.g., the `nameIdentifier` and `subgraphDeploymentID`). This conversion of a `bytes32` type to `uint256` to create the `subgraphID` results in slight increase in gas costs.

Consider using the `bytes32` type for the `subgraphID` for consistency and minor gas savings.

Update: *Acknowledged. TheGraph team has chosen to use `uint256` for the `subgraphID` to make it consistent with the tokenID of the NFT.*

[N05] Unnecessary use of `msg.sender`

The `mintSignal` function defines a local `curator` variable to store the `msg.sender` value, but still uses `msg.sender` in the call to `pullTokens`.

Consider using the `curator` value for consistency within the `mintSignal` function.

Update: *Fixed in commit `66d6e5ee82769e2b32d94d2cd8a4a03072fcf25a` of PR521.*

Conclusions

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