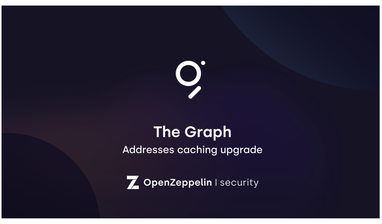
The Graph – Addresses Caching Upgrade Audit

APRIL 27, 2021 | IN SECURITY AUDITS | BY OPENZEPPELIN SECURITY



Introduction

The Graph team asked us to audit the new functionality of the `Managed` contract that saves on gas consumption by caching contract addresses retrieved from the controller.

The pull request that we have audited is the PR#430 at commit `cab50f4975580d71cb43b5731a62b866d07d951bc` and the audited file is the following:

```
contracts/governance/Managed.sol
```

Overview

Two new variables have been added to the `Managed` state: the `addressCache` mapping and the `__gap` variable.

Values are set to the `addressCache` mapping in the `_syncContract` function, internally called by the `syncAllContracts` external function and triggered whenever contract addresses change.

The `__gap` variable is used to reserve slot storages for eventual future implementations. In this way, the protocol is able to save a cached version of the addresses of the protocol and avoid retrieving them from the controller when needed, improving gas consumption depending on the transaction that triggers such retrievals.

Summary

We are happy to see clear and modular code being proposed to enhance the protocol functionalities. We must note that the PR in question is still not merged; we assume that The Graph team will merge it as it is and that no other bugs will be introduced in eventual changes. Two auditors have audited the code over one day, with the findings presented below.

Update: *All of the following issues have been either fixed or acknowledged by the Graph team. Our analysis of the mitigations is limited to the specific changes made to cover the issues, and disregards all other unrelated changes in the codebase.*

Critical Severity

None.

High Severity

None.

Medium Severity

[M01] Old contract versions can be mistakenly used

The `syncAllContracts` function is in charge of updating the cached addresses with new values whenever needed. However, if this function is not called as soon as a new contract is deployed, users may happen to use the protocol with an old cached version of the contracts. The consequences of this could be various and of different severities since many contracts in the protocol extends from the `Managed` contract.

Additionally, there's no easy way for users to know exactly when the last sync of the cache was performed, even if tech-savvy users can retrieve that information from transaction timestamps.

Consider documenting how the `syncAllContracts` function will be called and when, especially when addresses change, and ensure users have access to this information. Moreover, consider saving the timestamp of the last performed cache update so that users can easily retrieve the information directly from the contract and compare it with the deploy time of a new contract whose address must be synced.

Update: *Fixed in commit `824bb25f17b70cf27c76fe72265f2fdb294a121f` where documentation was updated and a new event `ContractSynced` is called when an address changes.*

[M02] `_resolveContract` return value can cause failure

The `_resolveContract` function of the `Managed` contract attempts to retrieve the address of a contract given the `keccak256` representation of its name passed in as the `_nameHash` parameter. The `_resolveContract` function returns the cached value if it exists but otherwise returns the result of calling the `controller`'s `getContractProxy` method. The `getContractProxy` method can return the zero-address either in the case the contract is never set or if it is unset.

The return value of `_resolveContract` is expected to be a contract satisfying one of many interfaces so that other functions can call its methods. But, when the returned value is the zero-address, these attempts to call its methods will fail because the zero-address does not have a fallback function.

Following the "fail early and loudly" principle, consider including specific and informative error-handling structures to avoid unexpected failures.

Update: *Acknowledged but not fixed. The Graph did not include that check because it is a "frequently used function, with low probability to be zero" and they were "concerned about the gas cost of doing that check".*

Low Severity

[L01] Duplicated getters

The `Managed` contract has been refactored to add getter functions that retrieve the addresses of the five contracts through the call to the `_resolveContract` internal function.

Contracts are set in the `_syncContract` function to the `addressCache` mapping that uses the `keccak256` representation of the contract names as key values.

At the same time, the `addressCache` mapping is defined as public, and it exposes an automatically generated getter function that can be used to retrieve the same addresses by the direct use of the `keccak256` name representation.

To be more gas efficient, consider either defining the `addressCache` mapping as private or removing the explicit getters to avoid duplicate getter functions.

Update: *Fixed in commit `01dea49469c3c7bee0c1989ff80baa267c14fb0f` where the `addressCache` mapping is defined to be private.*

Notes & Additional Information

[N01] Event lacks detail

The `SetController` event of the `Managed` contract is emitted whenever the controller is set by `_setController` function and has the new `controller` as its only parameter.

It may benefit off-chain services to receive both the old value and the new value of this updated variable.

Consider including both the old and new values of the `controller` as parameters to the `SetController` event.

Update: *Acknowledged. In the words of The Graph team:*

> The reason for not adding the old value is to maintain consistency and avoid changing the signature that is currently being used by third party services and the network subgraph.

[N02] Lack of indexed parameters in events

None of the parameters in the events defined in the `Managed` contract are indexed. Consider indexing event parameters to avoid hindering the task of off-chain services searching and filtering for specific events.

Update: *Akcnowledged. In the words of The Graph team:*

> The ParameterUpdated events are not currently carrying values, just a variable name that then a subgraph can read from the contract. For that reason we don't see a huge gain to change the interface at this moment.

[N03] Inconsistent style

The `onlyController` modifier of the `Managed` contract deviates from the style of the other modifiers of this contract in that it does not wrap an internal function call.

Considering how much value a consistent coding style adds to the project's readability, we recommend enforcing a standard coding style.

Consider refactoring the `onlyController` modifier to wrap internal function calls to have style consistent with the other modifiers of this contract.

Update: *Fixed in commit `b1d4fa52b479f64c16942845b5d82b5d5eae37237`.*

[N04] Missing functionality

The `syncAllContracts` function is externally callable, and it updates five featured contracts in just one single execution.

To be more gas efficient, if just one or a few of the five contracts change, consider implementing a function to sync just one contract at a time.

Update: *Acknowledged. In the words of The Graph team:*

> We preferred to avoid adding extra bytecode to all the contracts that will inherit Managed, and just use the `syncAllContracts()`. Even if could take more gas it will seldomly be used.

[N05] Unnecessary import

In the `Managed.sol` file, consider removing the import statement for `IManaged.sol`, as it is never used in the `Managed` contract.

Update: *Fixed in commit `11826bc0a6088aa294b7c25dbea7fa2442a56db4b`.*

Conclusions

2 Medium and other lower severity issues were reported with recommended changes to improve the codebase.