OpenZeppelin | security

The Graph Collect Allowlist Removal Incremental Audit

The Graph

November 8, 2023

Table of Contents

Summary

Type	DeFi
Timeline	From 2023-10-30 To 2023-11-03
Languages	Solidity

Total Issues	3 (0 resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	1 (0 resolved)
Medium Severity Issues	0 (0 resolved)
Low Severity Issues	1 (0 resolved)
Notes & Additional Information	1 (0 resolved)

Scope

We audited pull request #864 of the graphprotocol/contracts repository at commit 3882b35.

In scope were the following contracts:

```
graphprotocol/contracts
├── l2
│   └── staking
│         └── L2Staking.sol
└── staking
      ├── IStakingBase.sol
      ├── IStakingExtension.sol
      ├── StakingExtension.sol
      ├── Staking.sol
      └── StakingStorage.sol
```

Overview of Changes

Commit 3882b35 removes the query fee payers allowlist. Previously, only specific governance-approved accounts (gateways) were able to pay query fees to indexers on the network via permissioned calls to the `collect` function of the `Staking` contract. Specifically, the caller must belong to the `assetHolders` mapping. The removal of this restriction allows anyone to pay, making this a permissionless operation motivated by a push towards further decentralization of The Graph network as part of the new Timeline Aggregation Protocol change audited previously. In addition to simply removing the `assetHolders` mapping and associated permissions, a one epoch delay was added between allocation creation and fee collection to prevent manipulation of curation pools. As such, the implications of these changes on the various components of the protocol constituted the primary focus of the audit.

Security Model and Trust Assumptions

The removal of the query fee payers allowlist enables anyone to call `collect`. This creates a potential for rounding attacks which manipulate balances in the curation or delegation pools. A similar attack flow has been described in vulnerability M-09 of the Staking and Vesting L2 Migration audit report. In order to mitigate this vulnerability, a delay between allocation creation and collection is enforced to make the attack in the current context harder to execute. This way, the atomicity in the allocate/curate/collect flow is removed, putting the attacker at risk through potential front-running of the exploit transaction.

High Severity

H-01 Ineffective Mitigation of Shares Inflation Attack on Curation Bridging

Background

An attack scenario, similar to the EIP-4626 share inflation attack, involving query fees and curation bridging was identified in a previous audit:

1. During the migration of a subgraph from L1 to L2 with its curation signal, an indexer on L2 creates an allocation. This indexer may or may not also be the subsequent attacker of the curation pool.
2. An attacker curates 1 wei to the subgraph.
3. When query fees are collected for the allocation and added to the curation pool, they inflate the signal price in tokens.
4. When the subgraph and its curation signal migration transactions are completed on L2, the inflated price causes integer division truncation, adding the tokens to the pool but not minting any new signal. This steals the GRT from the bridging users to the attacker.

Previously, this attack was unlikely due to the difficulty of controlling the amount and timing of query fees. However, the current removal of the allowlist on `collect` makes the attack easier to execute in a predictable way with attacker-controlled amount and timing.

To mitigate this attack, a delay of 1 epoch was added between the creation of an allocation and collecting the fees for it. This delay is intended to break the atomicity of the inflation step and make the attack riskier for the attacker due to the opportunity for other curators to join the pool.

Attack Scenarios

However, this mitigation is insufficient and several scenarios may circumvent it:

- If the timing of the migration of a subgraph is known far in advance, and the attacker can guess the `_subgraphDeploymentID` that will be used, they may create an allocation for it well ahead of time. This would circumvent any reasonable delay that can be imposed.

- The attacker may target bridging transactions initiated just before the epoch boundary, allocate, and wait. They will then be able to spoof fees only after the epoch transition, and will do so only if the transfer has not landed on L2 by that time such that they are still the only curator in the pool.
- The attacker may target failed bridging transfers (that have failed due to insufficient gas parameters on L2). This will allow them to trigger a retryable ticket at the time of their choosing right after successfully spoofing the fees.
- The attacker may be the subgraph owner themselves and thus able to control the timing on both layers. In this scenario, the owner of a subgraph bridges and sets up the attack differently from the other scenarios. Instead of curating 1 wei, they burn their signal to 1 wei after bridging.

Likelihood Considerations

In all scenarios, the attacker risks losing the spoofed part of the fees to another curator. However, they may always choose the timing of spoofing in a way that can minimize this risk. The attacker can observe when the target bridging transaction is submitted on L1 and spoof the fees just before it is included on L2, reducing the time their funds are exposed to risk.

A factor that may be seen as reducing the likelihood is that the indexer must stake at least 100K GRT, which may be slashed by governance during the 28-day "thawing period" before the stake can be withdrawn. However, since the only action performed by the indexer is the creation of the allocation, if the inflation attack is carried out from a different address, the indexer will have sufficient "plausible deniability". Consequently, they may avoid slashing by claiming that they set up an allocation in legitimate anticipation of the migration and that a separate party carried out the attack on the curation pool.

As a result, given the attacker's ability to minimize risks and the multiple available scenarios, the likelihood is deemed medium. This results in a "high" severity due to the high impact arising from the theft of other users' funds.

Recommended Mitigation

Consider replacing the delay mitigation with a mechanism enforcing an invariant of a minimum curation amount. This would prevent inflation attacks similar to the already introduced invariant of a minimum delegation amount. Additionally, in the case of a bridged curation that is too small, consider transferring the tokens to the owner of the tokens.

Low Severity

L-01 Query Fee Volume Spoofing May Avoid Taxes and Fees for Small Amounts

During a call to `collect`, multiple taxes and fees are calculated and subtracted, or burned:

- `protocolTax`
- `curationFees`
- `delegationRewards`
- Excess non-rebated fees are burned

However, for small amounts of fees, an indexer may spoof the fees to their deployment without incurring any of the taxes or fees.

First, the protocol tax and the curation fee can get truncated to 0 because they are calculated through a multiplication with a precision of 6 decimal points (example for protocol tax):

- The tax is set at 1% (10000 PPM), leading to an amount of 99 wei in fees to be rounded down to 0.
- The curation fee cut is 10% (100000 PPM), leading to an amount of 9 wei to be rounded down to 0.

Secondly, the delegation query fee cut can be changed by the indexer at will due to the default cooldown being 0, and can be set in such a way that the query fees are not distributed to the delegators during spoofing. This can be adjusted and reset atomically before and after the spoofing, ensuring that genuine delegation fees are not diverted from the delegators so as to avoid causing them to undelegate.

Finally, no burning will occur for the initial part of the exponential rebate function because the entire fee will be rebated. While the exponential rebate function eventually tapers off and may cause spoofed fees to be burned, initially, for the accumulated fees up to a ratio of 1:25 relative to the stake, it is exactly linear, thereby rebating the entire fee. In this range, there is no burn at all due to the limit on the maximum exponent during the rebate calculation. As a result, the full amount originally pulled will be transferred to the indexer, without any taxes or fees being applied.

Since the contracts are deployed on Arbitrum L2 which charges for gas mostly based on calldata costs, it becomes feasible to call `collect` multiple times within a loop from a contract. When measured with L1 gas settings, a loop of 100 calls to `collect` (each collecting 9 wei) cost 3.2M gas, averaging only 32K gas per call. Furthermore, after the planned Ethereum Denali hard fork to enable EIP-4844, L2 fees are expected to decrease dramatically. This, in turn, will likely make this type of manipulation even cheaper to execute.

Since the previous allowlist of `assetHolders` approved by governance is now removed, deploying such a contract and paying fees through it becomes possible for any party paying fees. This enables a single cheap transaction (due to minimal L1 calldata cost) to spoof a small amount of fees directly to the indexer, without paying the expected amount of fees and protocol taxes, thereby undermining the incentive structure of the protocol.

According to the docs, the query fee volume may be used as an off-chain indication of the profitability of a subgraph for indexers. Thus, a subgraph owner may continuously spoof fees in this tax-free manner by also being an indexer using the above method to attract indexing capacity to their subgraph.

Consider rounding up in favor of the protocol and curators in their fee calculations. This can be done by calculating the remainder and subtracting it, similar to how the delegators' cut is calculated. Alternatively, 1 wei can be added to the fee calculations as long as the remainder is not 0.

Notes & Additional Information

N-01 Unnecessarily Complex and Inefficient `require` Statement

The `require` statement in `_undelegate` can be refactored to be more readable. This can be done by extracting the calculated token amount to an intermediate variable. Additionally, `delegation.shares`, `pool.shares`, and `pool.tokens` may be cached in intermediate variables to save gas on redundant storage reads.

Conclusion

The Graph is a decentralized protocol for indexing and querying data from blockchain networks, making it easier for developers to access blockchain data without the need to run their own infrastructure. It allows developers to create and publish APIs, called subgraphs, that provide efficient and scalable access to blockchain data.

The audit yielded only one high-severity issue and one medium-severity issue. Some code improvements have been suggested to improve the clarity of the codebase.

The Graph team were highly responsive and forthcoming throughout the audit period. They helped the audit team understand the intended changes and analyze any potential issues.