OpenZeppelin | security

OpenZeppelin Checkpoints Library Audit

November 15th, 2022

This security assessment was prepared by OpenZeppelin.

Table of Contents

Table of Contents 2

Summary 3

Scope 4

Introduction 5

Findings 5

Low Severity 6

L-01 Gas inefficiency 6

L-02 Unused private function 6

Notes & Additional Information 7

N-01 Inconsistent formatting 7

N-02 Incorrect require statement 7

N-03 Missing docstrings 7

N-04 Mixed nomenclature 8

Conclusions 9

Summary

Type	Library
Timeline	From 2022-10-17 To 2022-10-28
Languages	Solidity

Total Issues	6 (3 resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	0 (0 resolved)
Low Severity Issues	2 (1 resolved)
Notes & Additional Information	4 (2 resolved)

Scope

We audited the `OpenZeppelin/openzeppelin-contracts` repository at the `14f98dbb581a5365ce3f0c50bd850e499c554f72` commit.

In scope were the following contracts:

```
contracts
└── utils
    ├── math
    │   ├── SafeCast.sol
    │   └── Math.sol
    └── Checkpoints.sol
```

Introduction

The scope for this audit is the `Checkpoints` library. The `Checkpoints` library is designed to allow developers to chronologically track specific events happening within their application. This is achieved through the use of a `History` or `Trace` struct which themselves are composed of checkpoints. A checkpoint is a struct containing the `block.number` for when the checkpoint is created, along with a specified `value`.

The library has several functions repeated that are meant to deal with different versions of the struct used. The struct versions differ in their data type sizes, allowing developers to choose the one that best fits their custom codebase.

The commit used for the audit is 14f98db and the code has been audited during the course of five (5) days by two (2) auditors.

Findings

Here we present our findings.

Low Severity

L-01 Gas inefficiency

The getAtProbablyRecentBlock function is intended to provide an optimized search for recent checkpoints where recent is defined as the checkpoint being among the last `sqrt(n)` checkpoints.

However, in an internal test conducted by our team, the optimization was tested to create slightly more expensive lookups for smaller arrays such as those of size `5`, `8`, `16`, and `25`. Because of this, consider increasing the minimum length check to a larger size to ensure optimization is actually achieved.

Update: *Acknowledged, resolved. After discussions with OpenZeppelin's Libraries & Tooling team, both of our teams came up with specific tests giving contrastring results. Given the subjectivity of those and the little improvement in gas consumption that might derive from it, we leave the issue as it is for the reader to have knowledge of it.*

L-02 Unused private function

This specific `_lowerBinaryLookup` function is marked as private but is not used by any other function.

Given the different repetitions of the same functions, in order to improve readability, clarity and code size, consider removing the unused function.

Update: *Acknowledged, not resolved. The OpenZeppelin stated:*

> *This will be tackled in a later PR.*

Notes & Additional Information

N-01 Inconsistent formatting

The `_unsafeAccess` functions are formatted in a different way across definitions. Consider reviewing the entire codebase and unifying the way code lines are formatted.

This will improve readability and overall quality of the codebase.

Update: *Acknowledged, not resolved. The OpenZeppelin team stated:*

> This is due to the use of automatic formatting and some of the definitions going over the line length limit. We believe there is an upcoming version of prettier-solidity that may result in a more consistent output.

N-02 Incorrect require statement

Within all `_insert` functions, there is a comment stating that all keys must be increasing. However, the require statement only requires that a key be *non-decreasing*.

In order to improve correctness and clarity, consider aligning the comment and the require statement. Also, consider reflecting this *non-decreasing* requirement in the require statement error message.

Update: *Resolved in commit 0943b4d.*

N-03 Missing docstrings

In the codebase there are some places that might benefit from improved docstrings. Some examples are:

- The private `_unsafeAccess` functions are using inline assembly and have no docstrings at all. While it is not common practice to include docstrings in private functions, functions using inline assembly warrant additional documentation.

- The `getAtBlock` function is designed to disallow querying a checkpoint pushed within the same block as `block.number`. While there might be several reasons for that, it's not clear why the library limits this.

Consider improving the docstrings in the examples mentioned to improve readability and understandability as well as to improve the developer experience when making use of this library.

Updated: *Resolved in commits 6b5de2c and 1ee1154.*

N-04 Mixed nomenclature

The structs `Checkpoint160` and `Checkpoint224` both have their initial named value as `_key`. The near identical `Checkpoint` struct has its initial named value as `_blockNumber`. Functions that interact with these initial values all have named parameters using the `key` nomenclature.

We understand that part of the reason of the current codebase state is because of backward compatibility with previous versions. However, consider using `blockNumber` for parameter names for functions that interact with `Checkpoint` like the `_insert` function. Moreover, consider whether it is also worth changing the `_lowerBinaryLookup` and `_upperBinaryLookup` functions to have `blockNumber` instead of `key` as input parameter.

Update: *Acknowledged, not resolved. The OpenZeppelin team stated:*

> *It will be addressed in a future PR along with the unused private function issue.*

Conclusions

No high or critical severity issues have been identified. Minor recommendations have been made to improve documentation and readability within the codebase.