Sablier v2-periphery Audit

Executive Summary

Auditor	Iaroslav Mazur, Smart Contracts Engineer
Timeline	03.07.2023 – 09.07.2023
Language	Solidity
Audit Methods	Architecture Review, Static Code Analysis, Automated Testing (Fork, Fuzz, Integration)
Source Code	[Sablier v2-periphery (005df5f)]()

Overall Assessment

Sablier v2-periphery builds on top of **Sablier v2-core** and **PRBProxy**, abstracting the lower-level protocol logic away from the users and offering them more complex/aggregate functionalities. All that makes the whole experience of interacting with the combined Sablier V2 solution more user-friendly.

The **Sablier v2-periphery** repository has been developed with thoroughness and attention to detail. Its codebase manifests significant efforts invested in keeping the code clean, readable and secure. The testing of the project is extensive and comprehensive, covering a multitude of aspects in terms of interacting with the peripherial smart contracts, and boasting a 99% code coverage.

During the audit, 1 Gas Optimisation and 13 QA findings (according to **Code4Arena Severity Categorization**) have been identified.

Aside from the findings listed in this report, there have, also, been a number of smaller improvements/optimizations proposed for integration into the Sablier v2-periphery codebase directly via the Pull Requests **#138** and **#139** on GitHub. The fixes for a part of the findings from this report have also been included in the Pull Requests.

Update: The findings from this report and the associated Pull Requests have been addressed as of commit **561f49f**.

Scope

The reference point for the codebase analysed during this audit was the **005df5f** commit from the **Sablier v2-periphery** repository. More specifically, the following has been reviewed:

- All the code under “src” and “test”.
- The deployment scripts under “script”(e.g. `DeployPeriphery`).

Assumptions

Here are the assumptions that were kept in mind while analysing the project codebase:

- Every stream sender has a PRBProxy on which a SablierV2ProxyPlugin is installed.
- All relevant Sablier V2 contracts are listed in SablierV2Archive.
- All assumptions in Sablier V2 Core.
- All assumptions in PRBProxy.

Findings

Outlined below for your review are the findings identified during the audit, categorized by their severity and impact. Where appropriate, a short guidance on a solution for the finding is also provided. For clarity, each finding is prepended (where applicable) with a link to the file(-s) which it applies to.

Gas Optimisation

1. SablierV2ProxyTarget.sol (line 78): `_getBalances()` and `_postMultipleCancellations()` should only be called when `assets.length > 0`

Quality Assurance

1. Errors.sol (line 12): the `CallNotDelegateCall` error would be better named `NotADelegateCall`.
2. SablierV2ProxyTarget.sol (line 658): `_postMultipleCancellations()` would be better named `_transferBalanceDifferencesToOwner()` (or something similar), to be more suggestive about what's, actually, happening inside the function.
3. SablierV2ProxyTarget.sol (line 667): settled Streams cannot be canceled, but this doesn't mean `deltaBalance` cannot be 0, because the `assets` array passed to `cancelMultiple()` and/or `batchCancelMultiple()` may contain `IERC20`'s that are completely unrelated to any of the Streams that are being cancelled, resulting in no change of balance (i.e. delta == 0) for the respective tokens. Furthermore, spam txs can be created this way.
4. ISablierV2ProxyTarget.sol (line 40): not really a "mirror", given that `cancel()` does more than just that.
   Also applies for `cancelMultiple()`
5. Base.t.sol: (line 352): downcasting `batchSize` that is also passed to the function during Fuzz tests from `uint256` to `uint128`.

   Also applies for `batchCreateWithRange()`
6. onStreamCanceled.t.sol (line 49): the call of the `onStreamCanceled` hook of the plugin isn't, actually, tested.
7. list.t.sol (line 32): the emission of the `List` event isn't tested.
8. unlist.t.sol (line 20): the `archive.unlist()` function that's being tested isn't, actually, called in the test.
9. unlist.t.sol (line 31): he emission of the `Unlist` event isn't tested.
10. getMethods.t.sol: the `.tree` file is missing for this test.
11. onStreamCanceled.t.sol (line 67): the plugin being "triggered" (via the hook) isn't, actually, tested .
12. batchCreateWithRange.t.sol (line 14): wrong function called (`batchCreateWithDurations()` instead of `batchCreateWithRange()`).

13. cancelAndCreate.tree: the `.tree` file feels incomplete with regards to everything actually happening in the `.t.sol` file.

    Also applies to `wrapAndCreate.t.sol`.