Sablier v2-core re-audit

Executive Summary

Auditor	Iaroslav Mazur, Smart Contracts Engineer
Timeline	01.06.2023 – 13.06.2023
Language	Solidity
Audit Methods	Static Code Analysis, Automated Testing
Source Code	Sablier v2-core (5997ac0)

Overall Assessment

Since the original Sablier v2-core Audit, the Sablier team has worked on several important updates for the `v2-core` project, including the NFT Descriptor functionality (part of the ERC721 Metadata Extension), the reworked categorization for the Stream states (that better represents the Stream dynamics), the adapted and added automated tests and various other enhancements to the codebase.

The team has, notably, continued down the path of a quality-oriented project development focused on delivering the functionalities in a well-tested and properly-documented manner.

During the audit, 7 Gas Optimisation and 13 QA findings (according to Code4Arena Severity Categorization) have been identified, as well as several additional, general purpose things to consider.

Some of the findings mentioned in this report have been addressed and included in Pull Request #550 on GitHub.

Update: The findings from this report and the associated Pull Request have been addressed as of commit ceaf05f.

Scope

The reference point for the codebase analysed during this audit was the difference between the 5997ac0 and the 8bd57eb commits (the latter of which has already been covered by the original Sablier v2-core Audit) from the Sablier v2-core repository.

Assumptions

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

- Immutable variables have been set correctly in the constructors, i.e. `MAX_FEE` does not exceed 10^{18}, and `MAX_SEGMENT_COUNT` has a value that cannot lead to an overflow of the block gas limit.
- The total supply of any ERC-20 token remains below $2^{128} - 1$, i.e. `type(uint128).max`.
- The `transfer()` and `transferFrom()` functions of any ERC-20 token strictly reduce the sender's balance by the transfer amount and increase the recipient's balance by the same amount. In other words, tokens that charge fees on transfers are not supported.
- An address' ERC-20 balance can only change as a result of a `transfer()` call by the sender or a `transferFrom()` call by an approved address. This excludes rebase tokens and interest-bearing tokens.

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. **SablierV2NFTDescriptor.sol**: `vars.sablierAddress` isn't used after its definition, even though there are 2 places further in the function where this would be very suitable.
2. **SablierV2NFTDescriptor.sol**: `vars.` `asset` is converted from `address` to `IERC20Metadata` at its initialization, and back to `address` - each time it's accessed afterwards. The `asset` member could just be kept as an `address` to simplify things.
3. **SablierV2LockupLinear.sol:** `_streamedAmountOf()` could also avoid the costly calculations inside `_calculateStreamedAmount()` for the `PENDING` and `SETTLED` Stream states.

   Also applies to `SablierV2LockupDynamic`.
4. **SablierV2LockupLinear.sol:** the `_streams[streamId].isCancelable` verification can be moved up, so that the SLOAD of `_streams[streamId].amounts` doesn't happen unless it can really make a difference.

   Also applies to `SablierV2LockupDynamic`.
5. **SablierV2Lockup.sol:** `withdrawableAmountOf()` is just a proxy for `_withdrawableAmountOf()`. These 2 functions could be merged.
6. **SablierV2LockupLinear.sol:** `refundableAmountOf()` could also avoid the costly calculations inside `_calculateStreamedAmount()` for the `PENDING` and `SETTLED` Stream states.

   Also applies to `SablierV2LockupDynamic`.
7. **SablierV2Lockup.sol:** these 2 verifications can be moved up the function, in order to only do the more complex verifications after the less expensive ones have passed.

Quality Assurance

1. **SablierV2Lockup.sol:** `withdrawableAmountOf()` should be used here, instead of `_withdrawableAmountOf()`.
2. Linear **createWithRange.t.sol:** duplicate value bounds, with the exception of `1` / `1 seconds`.
3. **createWithRange.tree**: typo.
4. **FlashLoan.t.sol**: the `whenAssetFlashLoanable` modifier could be defined just once in here, instead of the 5 times it's currently defined in the contracts inheriting from

`FlashLoan_Integration_Shared_Test`.

5. **SablierV2NFTDescriptor.sol**: a cast from a parent (`IERC721Metadata`) to child (`ISablierV2Lockup`) is happening here, which is both incorrect (from the architecture perspective) and unnecessary.

   To address this, the `IERC721Metadata` parameter of `tokenURI()` could be declared as `ISablierV2Lockup` directly.

6. **SablierV2Lockup.sol**: the visibility of `_nftDescriptor` can be reduced to `private`.
7. **DataTypes.sol:** the wording is confusing/incorrect, as by the moment the `isStream` member is accessed, the `Stream` struct can't be inexistent. A better way to say this would, probably, be "whether the associated Stream has been created/initialized".
8. **setProtocolFee.t.sol:** the imported `IERC20` symbol is not used in the contract.
9. **SablierV2LockupLinear.sol** / **SablierV2LockupDynamic.sol:** the `isCold()` function definition could be moved up the hierarchy tree to `SablierV2Lockup`.
10. **Fuzzers.sol:** the `Math` symbol is imported, but never used.
11. **SablierV2LockupLinear.sol** / **SablierV2LockupDynamic.sol:** the `isWarm()` function definition could be moved up the hierarchy tree to `SablierV2Lockup`.
12. **SablierV2LockupLinear.sol:** the "if the stream is either depleted or canceled" bit is too verbose and is covered suggestively by "otherwise".

    Also applies to `SablierV2LockupDynamic`.

13. **SablierV2LockupLinear.sol** / **SablierV2LockupDynamic.sol:** the `refundableAmountOf()` function definition could be moved up the hierarchy tree to `SablierV2Lockup`.

Additional Feedback / Things to Consider

1. *Visually incorrect relative representation of the `STREAMING` and `DEPLETED` hourglasses* (GitHub Discussion #544).
2. Both when the Stream sender and recepient `cancel()` the Stream, only the sender receives their part of the Stream assets, while the recipient has to additionally `withdraw()` their part.

   While this can be justified for the sender cancelling the Stream, it's not very userfriendly for the use-case of the recepient triggering the cancel operation (as they'd have to perform 2 txs to receive their assets).

Additionally, it's not like it'd make any sense for the recepient to keep the assets inside the Stream, anyway, as the only thing they can do with a `CANCELED` Stream is `withdraw()` from it.