Sablier v2-core Audit

Executive Summary

Auditor	Iaroslav Mazur, Smart Contracts Engineer
Timeline	21.03.2023 – 04.04.2023
Language	Solidity
Audit Methods	Architecture Review, Static Code Analysis, Automated Testing (Unit, Fuzz, Fork, Invariant)
Source Code	Sablier v2-core (8bd57eb)

Overall Assessment

All in all, the Sablier v2-core smart contracts exhibit a very high degree of quality and robust functionality. The smart contracts of the project are thoroughly tested and documented, as well as consistent and efficient. The Sablier team has broadly adhered to the industry standards for coding practices, design patterns and coding style.

During the audit, 1 Medium, 9 Gas Optimisation and 41 QA findings (according to Code4Arena Severity Categorization) have been identified.

The Medium finding has to do with the fact that the `tokenURI` function doesn't behave the way that is specified in the ERC-3156 standard that the Sablier v2-core smart contracts implement. More specifically, the `tokenURI` implementation doesn't revert when the queried NFT doesn't exist.

Aside from the findings listed in this report, there have, also, been a number of small improvements/optimizations proposed for integration into the Sablier v2-core codebase directly via the Pull Requests **#426** and **#427** 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 **2e89615**.

Scope

The reference point for the codebase analysed during this audit was the **8bd57eb** commit from the **Sablier v2-core** repository. More specifically, the following has been reviewed:

- All the code under “src” and “test”, with the exception of the **SablierV2NFTDescriptor** contract (which is currently just a demo stub).
- The deterministic deployment scripts under “script”.

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.

Medium

1. SablierV2LockupLinear.sol and SablierV2LockupDynamic.sol: The `tokenURI()` function doesn't revert when the queried NFT doesn't exist (as is required by the ERC-721 standard).

Gas Optimisation

1. It's not possible to change `immutable MAX_FEE` after the contracts are deployed. Instead of being passed all the way through SablierV2LockupLinear.sol / SablierV2LockupDynamic.sol, SablierV2Lockup.sol and SablierV2Base.sol, it could be inlined as a constant.
2. `MAX_SEGMENT_COUNT` could also be inlined as a constant, instead of being passed around at construction time.
3. SablierV2Lockup.sol: `_isApprovedOrOwner()` is always called with `msg.sender` as the `spender`, and so, the `spender` argument is redundant.
   Applicable for all definitions of `_isApprovedOrOwner()` in the codebase.
4. SablierV2LockupDynamic.sol (line 216): The initialization of `segmentCount` should be moved close to its first use, to optimize the use of resources for the cases when the function is called with the `currentTime >= endTime` condition yielding `true`.
5. There are many places (eg. SablierV2Comptroller.sol) where the old value of a property is saved in a temporary `memory` variable before it's changed, so that it can later be used when emitting an event.
   Alternatively, the event could be emitted before changing the property. This wouldn't break the Checks-Effects-Interactions pattern nor damage the readability, but would help save some gas each time the event is emitted.
6. SablierV2Lockup.sol (line 133): `streamIds.length` could be used directly, as it's only used once in the function.
7. ISablierV2Lockup.sol: `ISablierV2Comptroller` is imported, but is never used.

8. SablierV2Lockup.sol: `IERC20` is imported from, but is never used.
9. SablierV2LockupDynamic.sol: `sd` is imported, but is never used.

Quality Assurance

1. SablierV2LockupLinear.sol (line 290): The visibility of `createWithRange()` can be reduced to `external`.
2. SablierV2Lockup.sol (line 104): The first check inside the function could be abstracted in a separate function.
3. SablierV2LockupDynamic.sol (line 314): The line incrementing `index` could be moved to the beginning of the loop body, removing the mental burden of having to think of the "current" element as `index - 1` and the "previous" one - as `index - 2`.
4. SablierV2Base.sol (line 62): The event emission should be a part of the “Effects” section.
5. DataTypes.sol (line 45): The `Lockup.Status` enum should be numbered.
6. flashLoan.t.sol: The flash loaned amount shouldn't be fuzzed.
7. Both streamedAmountOf (Linear) and streamedAmountOf (Dynamic) pass, but don't verify what they're supposed to verify. Not to the full extent, at least.
   When `streamedAmountOf()` is called on a previously-canceled stream, the withdrawn amount should be returned. While it's true that if you create a stream and, then, cancel it immediately, the withdrawn amount is, practically, 0, this is, also, the default value for the withdrawn amount.
   `vm.warp({ timestamp: DEFAULT_END_TIME });` could be added to the beginning of the test function, in order to have a non-zero withdrawn amount to compare against afterwards.
8. flashLoan.t.sol: The dealing of sufficient liquidity should happen inside the `whenSufficientAssetLiquidity` modifier, instead of being done repetitively inside the test functions.
9. createWithDeltas.t.sol (Dynamic): There's no test for "no delegate call", even though the associated `.tree` file specifies it.
10. createWithMilestones.tree (Dynamic): The "block gas limit overflow" logic isn't present in the test.
11. burn.tree: The tree is missing a part of the test structure.

12. transferAdmin.t.sol (line 15) The second condition should be removed, because the `Adminable` implementation doesn't require the `newAdmin` to be different from the current one.
13. Linear.t.sol (line 303) and Dynamic.t.sol (line 312): The “NFT burned or not” verification should be moved outside the if-else block, such that this is checked for all cases, not just when the stream has been cancelled.
14. LockupLinearCreateHandler.t.sol (line 45), LockupDynamicCreateHandler.t.sol (line 55) The `useNewSender` modifier could be defined just once for the 2 contracts, in their base.
15. cancel.t.sol (line 167): The bodies of `test_Cancel_Sender_RecipientDoesNotImplementHook()`, `test_Cancel_Sender_RecipientReverts()` and `test_Cancel_Sender_RecipientReentrancy()` are the same, with the exception of the empty/reentrant contract address. They could be abstracted away in a separate, reusable function.
16. In the test contracts, there are many functions called “…_RevertWhen_…”, even though they are, actually, expected to 1) do nothing or 2) do something else besides reverting.
17. The `return` NatSpec tag is missing in many function documentations throughout the codebase.
18. Given the similarity of the functions in Fuzzers.t.sol, some more NatSpec documentation could be added to better describe and differentiate them.
19. SablierV2LockupLinear.sol (line 199) Cliff Time isn't guaranteed to always be greater than Start Time (it may be equal to the latter).
20. SablierV2LockupLinear.sol (line 200): The latter hypothesis is not necessarily true, as the current `block.timestamp` might be situated between Start Time and Cliff Time.
21. In the documentation for any function that triggers a transfer of assets from the user, it should be specified that a sufficient allowance must have been signed previously.
22. DataTypes.sol (line 153): The documentation for `LockupLinear.CreateWithDurations` says that the `Broker` member is "optional". The wording is incorrect because the `Broker` must be a part of `LockupLinear.CreateWithDurations`. The correct wording would be that the `Broker` member “may be default initialized”.
23. Helpers.sol (line 38): several comments say that the `maxFee` is "always less than 1e18", while in other places, 1e18 is specified as "100%".
24. Helpers.sol (line 39): `protocolFee` and `amounts.protocolFee` are used alongside each other (sometimes, on the very same line), while having different semantics (one

represents the amount, while the other - the percentage of the total amount).
`Lockup.CreateAmounts.protocolFee` could become `Lockup.CreateAmounts.protocolFeeAmount`, in order to differentiate it from the `protocolFee` passed as parameter.

25. SablierV2FlashLoan.sol (line 72): The `flashFee()` from the comptroller is used as such, while the documentation of this property says it is a percentage (which suggests the need of dividing the number by 100 before using it in a multiplication).
    Also applicable for the other functions in SablierV2FlashLoan.sol.

26. ISablierV2Base.sol: The `ClaimProtocolRevenues` and `SetComptroller` events could be renamed as `ProtocolRevenuesClaimed` and `ComptrollerSet` respectively, to make it more suggestive that the corresponding action has been performed (and isn't about to be performed, as it is with functions).
    Also applicable to the event naming in the other contracts.

27. ISablierV2Comptroller.sol (line 50): `flashFee()` could be renamed as `flashFeePercentage()` in order to remove the burden of the possible confusion with the `flashFee()` from ERC3156, along with the necessity of commenting the warning about this confusion everywhere the fee appears.
    `protocolFees()` could also be renamed accordingly.

28. The "as an UD60x18 number" / "as an SD59x18 number" mentions in functions/events NatSpec are usually redundant.

29. SablierV2Comptroller.sol (line 40): `flashAssets()` could be renamed as `isAssetFlashLoanable()` for better code readability.

30. SablierV2Comptroller.sol (line 46): `protocolFees()` could be renamed as `protocolFeeForAsset()` for better code readability.

31. ISablierV2Lockup.sol: For consistency, functions like `returnableAmountOf()` could be renamed as `getReturnableAmount()` - or the `get[Smth]()` functions could be renamed as `smthOf()`.

32. SablierV2Lockup.sol (line 104): A better name for the `SablierV2Lockup_StreamNotCanceledOrDepleted` error would, probably, be `SablierV2Lockup_StreamNotBurnable`.

33. Helpers.sol (line 103): A better name for the `SablierV2LockupLinear_CliffTimeNotLessThanEndTime` error would, probably, be `SablierV2LockupLinear_CliffTimeGtEEndTime`

34. Helpers.sol (line 160): A better name for the `SablierV2LockupDynamic_StartTimeNotLessThanFirstSegmentMilestone` error would, probably, be `SablierV2LockupDynamic_StartTimeGtEFirstSegmentMilestone`.
35. SablierV2LockupLinear.sol (line 487): A better name for the `SablierV2Lockup_WithdrawAmountGreaterThanWithdrawableAmount` error would, probably, be `SablierV2Lockup_WithdrawAmountTooBig`. The fact that it's the "withdrawable amount" that's being referred to is easily deducible and logical.
36. SablierV2Lockup.sol (line 230): A better name for the `SablierV2Lockup_WithdrawArrayCountsNotEqual` error would, probably, be `SablierV2Lockup_WithdrawArraysDifferentSizes`.
37. ISablierV2Lockup.sol (line 106): `returnableAmountOf()` could be renamed as `refundableAmountOf()`.
38. DataTypes.sol (line 34): `Lockup.CreateAmounts` would, probably, be better named `Lockup.CreatedAmounts` or `Lockup.FeesAndDepositAmounts`.
39. Helpers.sol (line 18): `checkAndCalculateFees()` could be renamed as `checkFeesAndCalculateAmounts()`.
40. DataTypes.sol: For consistency, the `cancelable` member of several structs would be better named `isCancelable`.
41. DataTypes.sol: `SegmentWithDelta` could be renamed as `DeltaSegment`, `segmentsWithMilestones` - as `MilestoneSegment`.

Additional Feedback / Things to Consider

1. ISablierV2Lockup.sol (line 201): When the `withdraw()` function is called by an approved operator, shouldn't both the recipient and the sender be notified via hooks?
2. The possibility of losing access to the single `admin` address for the protocol should be considered and addressed via some sort of a flexible governance solution.
3. The Sablier contracts may receive (either by mistake or not) some ETH or tokens via a direct transfer (and not in the form of protocol fees), and right now, there's no way to withdraw that value from the contracts.