Sablier: v2core & v2periphery PR

Security Review

Cantina Managed review by:
Rustyrabbit, Security Researcher

July 3, 2024

Contents

1 Introduction **2**
- 1.1 About Cantina . . . 2
- 1.2 Disclaimer . . . 2
- 1.3 Risk assessment . . . 2
  - 1.3.1 Severity Classification . . . 2

2 Security Review Summary **3**

3 Findings **4**
- 3.1 Low Risk . . . 4
  - 3.1.1 Removal of try/catch could allow the recipient to make cancelable streams uncancelable . . . 4
  - 3.1.2 Integrators can wrongly interpret ´cliff´ as `end` of a Lineair stream when using getTimestamps . . . 4
- 3.2 Informational . . . 5
  - 3.2.1 Consider returning withdrawn and refunded amounts for improved integrations . . . 5

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

Sablier is a token streaming protocol available on Ethereum, Optimism, Arbitrum, Polygon, Ronin, Avalanche, and BSC. It's the first of its kind to have ever been built in crypto, tracing its origins back to 2019. Similar to how you can stream a movie on Netflix or a song on Spotify, so you can stream tokens by the second on Sablier.

From Jun 26th to Jun 28th the Cantina team conducted a review of v2core-v2periphery-pr on commit hash a8b2a1ff. The team identified a total of **3** issues in the following risk categories:

- Critical Risk: 0
- High Risk: 0
- Medium Risk: 0
- Low Risk: 2
- Gas Optimizations: 0
- Informational: 1

3 Findings

3.1 Low Risk

3.1.1 Removal of try/catch could allow the recipient to make cancelable streams uncancelable

Severity: Low Risk

Context: SablierV2Lockup.sol#L392-L398, SablierV2Lockup.sol#L582-L588

Description: The removal of the `try/catch` has given the recipient the ability to put their NFT in the allow listed Defi project and create a situation (e.g. lend against it beyond the current streamed amount or similar) where the Defi project would choose to revert or not return the correct selector in order to safeguard them from bad debt.

However this means that the sender of a cancelable stream is now unable to cancel their stream if the recipient has put their NFT in such a situation. This is a core setting of the stream where if so created the sender has the legitimate right to cancel their stream at any time, no matter the current use by the recipient. Furthermore the try/catch has also been removed from the `_withdraw` function.

If for some reason the recipient contract would also not allow the withdraw to happen (for example the cliff time must have passed), the stream would be locked in the contract until resolved. It is highly likely that in this case the contract would not allow the transfer of the NFT out (due to the same logic blocking the cancel and the withdrawal) and as such the removal of the `cancel` blocking would not be possible until either the withdrawal or cancel would be allowed by the contract. This again would impact the sender without their upfront approval or even knowledge.

Recommendation: As the impact of reduced cancelability of the stream (ie. certain contracts being able to block cancellations) primarily impacts the sender, consider letting the senders opt-in which contracts are able to block their streams on top of allow listing of contracts by Sablier to safeguard the technical implications to Sablier's contracts as a whole. This would allow co-operation between senders and Defi project where needed.

Other options would be to use a 2-step approach where either off-chain monitoring or regular on-chain polling of the cancellation status handle the needed state changes in the recipient contract. In the latter case an additional `statusOf` function that takes a list of tokens (i.e. currently owned by the contract) would be useful here to reduce the gas usage.

Sablier: Thank you for your feedback on the new hook design. We acknowledge this as a valid issue and we will mitigate the risk by managing the allowlist diligently and responsibly. Allowing senders to opt-in to the allowlisted contract is a good idea, but as we discussed it adds complexity and raises new questions. The recommendation of off-chain monitoring is also interesting but makes it asynchronous between the cause and effect, introducing new risk surfaces.

Therefore, we have decided to not take any action on this in the current version and will consider changes in future releases as we learn from the users.

3.1.2 Integrators can wrongly interpret ´cliff´ as `end` of a Lineair stream when using getTimestamps

Severity: Low Risk

Context: DataTypes.sol#L191-L194, DataTypes.sol#L275-L279, DataTypes.sol#L353-L356, SablierV2Lockup.sol#L211, SablierV2LockupDynamic.sol#L123-L128, SablierV2LockupLinear.sol#L108, SablierV2LockupTranched.sol#L107-L112

Description: Previously this function in `SablierV2LockupLinear` was called `getRange`. It is now renamed to `getTimestamps` but still returns the type `LockupLinear.Timestamps` which includes 3 timestamps (including `cliff`).

It now has the same function selector as `getTimestamps` in `SablierV2LockupDynamic` and `SablierV2LockupTranched` that return only 2 timestamps (in the form of `LockupDynamic.Timestamps` or `LockupTranched.Timestamps`, which only have `start` and `end`)

An integrator has no way to distinguish a Linear stream from a Dynamic or Tranched (as the ERC165 `supportsInterface` does not include Sablier's own interface Id). Therefor if and when they use this function on Linear stream while assuming it's a Dynamic or Tranched stream they will mistakenly interpret the

`cliff` time as the `end` time. Previously this wasn't an issue as the different function selectors would mean the call would revert due to the absence of a `fallback` function.

Recommendation: Either keep using different function selectors when returning different types of `Timestamps` structures or return the same type with the `cliff` set appropriately for dynamic and tranched streams.

Additionally if you decide to keep the different return types rename the Timestamps for Linear to indicate the inclusion of the `cliff`. This does not have any impact on this issue but aids developer to realize they are in fact very different types.

Also consider adding Interface Ids for `SablierV2LockupLinear`, `SablierV2LockupDynamic` and `SablierLockupTranched` in the ERC165 `supportsInterface` to enable integrators to distinguish between the different types of streams.

Sablier: Thank you so much for the suggestion. Our understanding is that most integrators will call the `getStream` function rather than using `getTimestamps` directly, so we would like to consider this as an informational rather than a low finding.

I agree that while `supportsInterface` can help integrators identify the contract they are interacting with, but each new upgrade requiring a new interface ID makes this approach less optimal as opposed to ERC165 in which the interface ID is expected to remain same. But since this is a valid point, we will do our best to communicate it through the docs.

3.2 Informational

3.2.1 Consider returning withdrawn and refunded amounts for improved integrations

Severity: Informational

Context: SablierV2Lockup.sol#L282, SablierV2Lockup.sol#L300, SablierV2Lockup.sol#L408-L410, SablierV2Lockup.sol#L413-L416

Description: `withdrawMax` and `withdrawMaxAndTransfer` do not return the withdrawn amount. In light of the need for improved integration with other Defi contracts it would be useful for them to have the withdrawn amount returned so as to not need an upfront call to `withdrawableAmountOf` in order for internal accounting to be correctly processed.

In case Defi projects would stream to recipients the same applies to `cancelMultiple` and `cancel` where the amount refunded has to be queried with `refundableAmountOf` upfront.

Recommendation: Consider returning the withdrawn amount for `withdrawMax` and `withdrawMaxAndTransfer`. The other withdraw don't need this as the amounts are specified as input. Also consider returning the refunded amount for `cancelMultiple` and `cancel`.

Sablier: Good suggestion. We have addressed it in Issue 955.