CANTINA

Sablier Airdrops 1.3.0

Security Review

Cantina Managed review by:

Eric Wang, Lead Security Researcher
RustyRabbit, Security Researcher

January 12, 2025

Contents

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, 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 Dec 17th to Jan 6th the Cantina team conducted a review of airdrops on commit hash aa45dabb. The team identified a total of **7** issues:

Issues Found

Severity	**Count**	**Fixed**	**Acknowledged**
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	2	1	1
Gas Optimizations	0	0	0
Informational	5	2	3
Total	**7**	**3**	**4**

3 Findings

3.1 Low Risk

3.1.1 Lack of input validation in `collectFees()` may result in incorrect values being emitted in the `CollectFees` events

Severity: Low Risk

Context: SablierMerkleFactory.sol#L88-L94

Description: Anyone can emit a `CollectFees` event with any value of `feeAmount` since the caller controls the input parameter `merkleBase`. The results would be incorrect if the events are read off-chain without validating that the `merkleBase` contract is deployed by the trusted `SablierMerkleFactory`.

Recommendation: Consider validating the input parameter `merkleBase`. The validation can be done by keeping track of the deployed contracts either on-chain (with a mapping `address => bool`) or off-chain.

Sablier: We index campaign addresses, meaning that if a dummy address is provided in `collectFees` emitting a `CollectFees` event, our system will not be spammed. In this case, input validation is unnecessary and would only add complexity to the function. Acknowledged.

Cantina Managed: Acknowledged.

3.1.2 Integer overflow allows the creation of linear streams with an invalid cliff and end time

Severity: Low Risk

Context: SablierMerkleLL.sol#L110

Description: In the `_claim()` function of `SablierMerkleLL`, the `cliffTime` of a linear stream is calculated as the sum of `timestamps.start` and `durations.cliff`, which are both of type `uint40`. In an edge case where the sum overflows to 0 due to unchecked math, e.g.,

- `timestamps.start = 1734481466`.
- `durations.cliff = type(uint40).max - 1734481466 + 1`.
- `durations.total = 1000`.

The creation of the stream will still succeed. The stream will have a duration of 1000 seconds and without a cliff. However, strictly speaking, the inputs are invalid, and the stream needs to be rejected since the cliff time is greater than the end time.

Recommendation: Consider removing the `unchecked` box when calculating the cliff and end time of the stream.

Sablier: Fixed in PR 39.

Cantina Managed: Verified.

3.2 Informational

3.2.1 Updating fee values may result in different deployment addresses of `SablierMerkle` contracts in the event of chain reorgs

Severity: Informational

Context: SablierMerkleFactory.sol#L113, SablierMerkleFactory.sol#L142, SablierMerkleFactory.sol#L267-L269

Description: The `createMerkleInstant()`, `createMerkleLL()`, and `createMerkleLT()` functions create a contract using CREATE2, whose address depends on both the `baseParams` (which determines `salt`) and `fee`. Since the admin of `SablierMerkleFactory` can change the `fee` at any time, the execution order for changing the fee and users deploying new contracts may result in different deployment addresses in case of a chain reorg.

For example, consider three transactions with the following execution order:

1. User creates a new `MerkleInstant`, whose address is A.

2. User transfers tokens to address A.
3. Admin updates the default fee.

A reorg occurs, and the execution order of the three transactions becomes:

1. Admin updates the default fee.
2. User creates a new `MerkleInstant`, whose address is B.
3. User transfers tokens to address A.

As a result, tokens are transferred to address A rather than B. The tokens are temporarily locked but can still be recovered by (1) the admin updating the default fee to the original value, (2) the user deploying another `MerkleInstant`, and (3) clawing back the tokens.

Recommendation: It is generally best practice for CREATE2-generated addresses to be determined only by user-provided inputs and the `msg.sender`. Consider not providing `fee` as a constructor parameter but fetching it from `SablierMerkleFactory` in the constructor of `SablierMerkleInstant`, `SablierMerkleLL`, and `SablierMerkleLT`. By doing so, the deployment address remains the same regardless of the `fee` value.

Sablier: Based on your recommendation, we have removed the `fee` parameter from the constructor arguments for merkle lockups as can be seen in PR 39.

Cantina Managed: Verified.

3.2.2 Notes on canceling unclaimed airdrop streams

Severity: Informational

Context: SablierMerkleBase.sol#L107-L112

Description: Consider the following scenario:

- The airdrop campaign starts now and lasts for 1 year.
- The stream starts now and lasts for 2 years.
- Alice, eligible for an airdrop, predicts her stream will be canceled after 0.5 years. e.g., she intends to do something that will cause it.

Alice has two claiming strategies:

- Suppose Alice claims the stream now. After 0.5 years, the sender will be able to cancel the stream directly.
- Suppose Alice does not claim her stream now. After 0.5 years, the sender has to claim Alice's stream on her behalf, pay a protocol fee, and cancel her stream. Otherwise, as long as Alice's stream is not claimed, the streamed amount will still accumulate over time, allowing her to receive more than she should.

In either scenario, Alice receives 0.5 years of streamed tokens. However, in the latter scenario, the sender is forced to spend a protocol fee to claim.

Recommendation: This issue can be solved on the protocol side by adding a function to allow the sender to cancel unclaimed airdrops if the streams are cancelable. Alternatively, the protocol admin can also temporarily adjust the custom fees to exempt airdrop campaigns if a large number of unclaimed airdrops need to be canceled.

Also, clarify this possible scenario in the code or documentation so that airdrop senders can be aware of it when launching airdrop campaigns with cancelable streams.

Sablier: Acknowledged. We will add a warning about this circumstance in the docs.

Cantina Managed: Acknowledged.

3.2.3 Unchecked calculation of a variable used in an invariant check

Severity: Informational

Context: SablierMerkleLT.sol#L203-L209

Description: In the `_calculateStartTimeAndTranches()` function of `SablierMerkleLT`, the `calculatedAmountsSum` is calculated as the sum of all the calculated amounts of the individual tranches.

The sum of all percentages of those tranches is already checked to be `1` in the `claim()` function so that in the current code base it is impossible the `calculatedAmountsSum` will overflow but an `assert` is used as extra invariant check to guarantee from the perspective of this function that the total sum does not exceed the `claimAmount` w.r.t. any undiscovered bugs or future code changes.

As such it does not make sense to calculate `calculatedAmountsSum` in an `unchecked` box as it leaves the door open to overflows (from the perspective of this function)

Recommendation: Consider removing the `unchecked` box when calculating `calculatedAmountsSum` which serves as an important invariant check.

Sablier: Thank you for the feedback, we decided to remove the assert statement here, and it is addressed in PR 38. Acknowledged.

Cantina Managed: Acknowledged. Not the outcome we preferred, but the removal of the assert is considered safe as:

1. The sum of the percentages is checked to be `1e18` in `_claim()`.
2. The tranche amounts are calculated based on the `claimAmount`.
3. The `claimAmount` is the `totalAmount` parameter given as input to `SablierLockup`'s `createWithTimestampsLT()`.
4. `createWithTimestampsLT()` in the `SablierLockup` in this codebase reverts when the total sum of the tranche amounts is not equal to the `totalAmount`.

Note that the `SablierLockup` contract to be used is given as input by the user creating the airdrop campaign and max approved to withdraw funds. The validation of funds distribution between recipients is thus partially delegated to the specified contract.

3.2.4 Handling empty arrays by reverting with an explicit custom error

Severity: Informational

Context: SablierMerkleLT.sol#L179

Description: An array out-of-bounds access error may be triggered in the `SablierMerkleLT._calculateStartTimeAndTranches()` function. If the state variable `_tranchesWithPercentages` is an empty array, accessing `tranchesWithPercentages[0]` will cause an out-of-bounds access error.

Recommendation: Consider reverting with a custom error in the empty array cases for better error handling and debugging purposes.

Sablier: Thank you for your submission. We agree that the lack of array length validation can result in an out-of-bounds access error. However, since the transaction will ultimately revert and the user will be unable to create a stream, we have opted not to include array length validation and keep the gas cost as it is. Acknowledged.

Cantina Managed: Acknowledged.

3.2.5 NatSpec and various improvements

Severity: Informational

Context: ISablierMerkleBase.sol#L28-L29, ISablierMerkleBase.sol#L100, ISablierMerkleFactory.sol#L86, ISablierMerkleFactory.sol#L92, ISablierMerkleFactory.sol#L103, ISablierMerkleLL.sol#L33, SablierMerkleFactory.sol#L65, DataTypes.sol#L10

Description:

1. SablierMerkleFactory.sol#L65: `isPercentagesSum100()` isn't used anymore as it is split between the constructor and `claim()` function. This function can now be removed. ISablierMerkleFactory.sol#L86 can be removed as well.
2. ISablierMerkleLL.sol#L33: Update the comments `start unlock amount` and `cliff unlock amount` from `amount` to `percentage`.

3. ISablierMerkleBase.sol#L28-L29: Change `ETH` to `native tokens` to avoid confusion as the contracts may be deployed on some other EVM chains. Same for:

    - ISablierMerkleBase.sol#L100.
    - ISablierMerkleFactory.sol#L92.
    - ISablierMerkleFactory.sol#L103.

4. DataTypes.sol#L10: Since the `expiration` variable can be set to 0, update the comment to:

```
/// @param expiration The expiration of the campaign, as a Unix timestamp. A value of zero means the
↪  campaign does not expire.
```

Sablier: Fixed in PR 38. The only one missing is the removal of `isPercentagesSum100`, as it is useful for the application. We have added a note about this in the NatSpec.

Cantina Managed: Verified.