CANTINA

Sablier Airdrops 1.4.0

Security Review

Cantina Managed review by:

Eric Wang, Lead Security Researcher
Akshay Srivastav, Security Researcher

April 5, 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 Mar 4th to Mar 15th the Cantina team conducted a review of airdrops on commit hash b9dba97. The team identified a total of **5** issues:

Issues Found

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

The Cantina Managed team reviewed Sablier's airdrops on commit hash ad350f2, concluding that all the issues were addressed and no new vulnerabilities were introduced.

3 Findings

3.1 Medium Risk

3.1.1 MerkleVCA campaign admins could abuse the clawback functionality

Severity: Medium Risk

Context: SablierMerkleBase.sol#L163-L171

Description: The campaign admin is allowed to claw back the funds if the first claim hasn't happened yet. In Merkle VCA campaigns, the first claim is more likely to occur near or after the end time since users are incentivized to do so. There may be a potential rug-pull scenario if the admin is malicious:

1. The admin funds the VCA campaign with sufficient funds.
2. All users wait until the end time as they believe they will be able to receive the full amount.
3. The admin claws back all the funds right before the end time (or before the first claim).
4. The users end up receiving nothing in return.

The above scenario is more of a concern for VCA campaigns due to its design of encouraging users to claim late, which increases the attack's likelihood and success rate.

Recommendation: A possible mitigation is to disable the grace period of VCA campaigns, thus preventing the campaign admin from clawing back the funds during the campaign period but only when the campaign has expired.

Sablier: While we recognize that `clawback` could potentially be misused in the case of Merkle VCA, we also believe that a grace period is necessary for addressing issues a misconfigured campaign, such as an incorrect Merkle root. That said, your concern is completely valid. To address this, we plan to introduce a dummy eligible recipient, perhaps with a value of just 1 wei. This recipient can be used to test the claim process immediately after the campaign is created. Not only will this ensure the claim is functioning correctly, but it will also trigger the grace period.

Cantina Managed: Acknowledged.

3.2 Low Risk

3.2.1 Timestamps for MerkleLL and MerkleLT campaigns should be validated before the first claim

Severity: Low Risk

Context: SablierMerkleLL.sol#L103-L109, SablierMerkleLT.sol#L123-L129

Description: In `SablierMerkleLT`, the validation of all parameters is deferred until the first Stream is created. For example, the timestamps of the `tranches` are only validated in the `Lockup` contract. Therefore, at SablierMerkleLT.sol#L123-L129, `endTime` is not yet validated and can be incorrect. A creator may accidentally provide some incorrect parameters that result in an incorrect `endTime` and less than the current time, therefore unlocking all the airdrops at once unexpectedly. Before the latest changes, such incorrect parameters would be rejected when creating the first Stream, so the creator would have a chance to claw back the funds and create a new campaign.

A similar issue also applies to `SablierMerkleLL`, where a creator may accidentally provide a `cliffDuration` less than the `totalDuration`, resulting in the end time being earlier than the cliff time.

Recommendation: The design choice of deferring the parameter checks until the first user claim is to allow the campaign creator to claw back the pre-funded tokens in case the parameters are invalid. However, with the latest changes, validating the timestamps becomes necessary even before the first claim.

The issue can be fixed by checking the timestamps in the constructor (when the airdrop campaigns are created). This would be a reasonable tradeoff given that there haven't been any pre-funded campaigns seen in practice at the time of writing, confirmed by the protocol team. In addition to the code changes, consider adding a warning in the docs that campaign creators should not pre-fund the campaigns before deploying them.

Sablier: We have decided to take no action on this. The grace period and the sender's ability to clawback funds are sufficient to mitigate this problem. Even if we implement all the checks in the constructor, it will not be a complete solution since start time can be set as `block.timestamp` in case of non-ranged streams.

See the full discussion in the discussion 94.

Cantina Managed: It is correct that adding checks in the constructor is not sufficient for non-ranged streams, since the start time can be set to `block.timestamp`, and therefore the end time needs to be calculated and checked at claim time (e.g., by removing the `unchecked` block, if a mitigation is implemented). Acknowledged.

3.2.2 Stale prices from Chainlink oracles could affect the minimum claim fee

Severity: Low Risk

Context: SablierMerkleBase.sol#L238-L244

Description: The `SablierMerkleBase` contract reads the native token price from a Chainlink price feed and calculates the minimum fee required for each airdrop claim. Although the code handles the case of invalid prices (which are less than or equal to 0), the prices can still be stale since the `updatedAt` value returned from the oracle is not checked against the current timestamp.

As a result, there could be a scenario where a user is paying enough fees from the UI (calculated based on the market price), but because of the stale price from the oracle, the transaction fails due to the minimum fee check. The user would then have to pay more than the minimum fee, which, however, could discourage small claims as they may become less profitable.

For example, assuming a minimum fee of 1 USD is set at the contract level, and for a claim of 5 USD, the fee charged on the UI is 1.1 USD. With such assumptions, the market price can only be at most 10% higher than the oracle price. If it exceeds 10%, then the user will need to pay more than 1.1 USD when claiming.

Recommendation: Consider checking if the price from the oracle is stale, and if so, return 0 from `_minimumFeeInWei()` to disable the fees temporarily.

A stale price can be defined as `block.timestamp - updatedAt > heartbeat + delta`. For example, for the ETH/USD price feed on Ethereum mainnet, the heartbeat is 3600 seconds. The `delta` parameter allows additional time for the price-updating transaction to be included on-chain (in case network congestion happens, etc.) For reference, here's Chainlink's example of reading a price feed and handling stale prices with a fallback price.

To be flexible and account for different heartbeat values across chains and price feeds, the total `heartbeat + delta` can be set as a configurable `delay` variable in the factory contract. The protocol admin can then adjust the `delay` settings if needed. Alternatively, a custom `Oracle` contract can be deployed on each chain, which includes stale price checks. Similarly, the protocol admin can set its `delay` value or make `delay` an immutable variable.

Sablier: Agree on the finding. Fixed in PR 93.

Cantina Managed: Verified.

3.2.3 Potential conflict between the deposited tokens and protocol fees on specific chains

Severity: Low Risk

Context: SablierMerkleBase.sol#L174, SablierMerkleBase.sol#L180-L196

Description: In the current design, the airdrop campaigns only support ERC-20-compliant tokens instead of native tokens. Native tokens are charged as protocol fees in each claim. Such a design could cause issues if the native token of the chain has an ERC-20 representation and is used as the campaign token. If so, the protocol would fail to distinguish between the deposited ERC-20 tokens and the received protocol fees.

For example, CELO, the native token of the Celo blockchain, has an ERC-20 representation at address 0x471EcE3750Da237f93B8E339c536989b8978a438. By design, calling `CELO.transfer()` has the same effect as transferring CELO with `msg.value`, and vice versa. In case an airdrop campaign has a `TOKEN` of CELO, there would be two unexpected outcomes:

1. If the campaign admin can claw back the funds, they can also withdraw the protocol fees kept in the contract. Note that the admin can also withdraw the protocol fees during the claim period by submitting a valid Merkle proof.
2. Anyone calling `factory.collectFees()` would accidentally transfer all the held CELO tokens from the campaign to the factory admin. It is because in `collectFees()`, the entire `address(this).balance` is transferred.

Below are some other blockchains whose native token also has an ERC-20 representation:

- Polygon: 0x0000000000000000000000000000000000001010.
- Metis: 0xDeadDeAddeAddEAddeadDEaDDEAdDeaDDeAD0000.
- Moonbeam: 0x0000000000000000000000000000000000000802.
- Tangle: 0x0000000000000000000000000000000000000802.

It is worth noting that not all ERC-20 representations listed above fully comply with the ERC-20 standard. For example, the POL token on Polygon does not implement a `transferFrom()` or `approve()` function.

Recommendation: A possible solution could be keeping track of the deposited ERC-20 tokens and the charged protocol fees in separate state variables and enforcing the parties to withdraw the token up to the tracked balances. This would avoid confusion between the deposited tokens and the protocol fees.

Alternatively, consider adding a warning in the docs that users on the affected chains should avoid using the ERC-20 representation of the native token for airdrop campaigns.

Sablier: Fixed in PR 93. Instead of tracking fees charged in a separate variable, we have decided to enable an option to block native tokens from being used with the protocol, if they implement an interface similar to ERC-20.

The rationale is that because this is only applicable on very few chains, adding new variables to track fees would impact the experience of non-affected chains, which are large in number.

Cantina Managed: Verified. The admin should be responsible for setting the correct ERC-20 representations on applicable chains.

3.3 Informational

3.3.1 NatSpec and various improvements

Severity: Informational

Context: SablierMerkleFactoryBase.sol#L83-L86, ISablierMerkleBase.sol#L87, SablierMerkleVCA.sol#L113, DataTypes.sol#L130

Description:

1. SablierMerkleFactoryBase.sol#L83-L86: Consider updating the comment to:

   // Effect: enable custom fee for the user.

   to match the actual behavior. Also, consider moving this `if` block after the next `if` block (which checks the `newFee` variable) to follow the Checks-Effects-Interactions (CEI) pattern.
2. ISablierMerkleBase.sol#L87: Consider updating the comment to.

   The `msg.value` must not be less than `minimumFeeInWei`.
3. SablierMerkleVCA.sol#L113: Consider updating the comment to.

   // Check: unlock start time is not in the future or now.

   Since a >= is used instead of >, users cannot claim at the start time.
4. DataTypes.sol#L130: Consider removing the second sentence, "A value of zero means the campaign does not expire", since Merkle VCA campaigns should expire eventually by having a non-zero expiration time.

   The following issues are identified from at commit dc6bbd6:

5. ISablierMerkleVCA.sol#L45: Consider adding the following to the `calculateForgoneAmount()` function's NatSpec:

   > Returns zero if the claim time is less than the vesting start time. Since the claim cannot be made, no amount can be forgone.

6. SablierMerkleVCA.sol#L154-L156: In `_calculateClaimAmount()`, the block `if (claimTime == VESTING_START_TIME) { ... }` is not necessary. Considering most of the claims happen after the start time, removing this `if` block could save gas for most users. However, it can be kept to be explicit.

7. airdrops/SECURITY.md: Consider adding the following warning for airdrop campaigns:

   > Do not pre-fund an airdrop campaign (i.e., sending tokens to the to-be-deployed address) before creating it through the factory. Otherwise, if the provided parameters are invalid, the campaign will fail to be created, and the funds will be lost.

8. Consider following the same event emission syntax across all the protocol contracts. As an example, in `SablierFactoryMerkleBase.sol#L65-L79`:

   - In `collectFees` function an event is emitted as `emit CollectFees(admin, campaign, feeAmount);`.
   - In `disableCustomFeeUSD` function an event is emitted as `emit DisableCustomFeeUSD({ admin: msg.sender, campaignCreator: campaignCreator });`.

Recommendation: Consider implementing the above suggestions.

Sablier: Fixed in PR 93 and PR 140.

Cantina Managed: Verified.