CANTINA

Sablier Flow 1.2.0

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 flow on commit hash 56e5b05. The team identified a total of **4** 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	2	0
Gas Optimizations	0	0	0
Informational	2	2	0
Total	**4**	**4**	**0**

The Cantina Managed team reviewed Sablier's flow on commit hash 3413c23, concluding that all the issues were addressed and no new vulnerabilities were introduced.

3 Findings

3.1 Low Risk

3.1.1 Use SafeERC20 to handle ERC-20 token transfers

Severity: Low Risk

Context: SablierFlow.sol#L422

Description: Using `token.safeTransferFrom()` from the `SafeERC20` library is necessary at SablierFlow.sol#L422. It is because `transferFrom()` cannot handle ERC-20 tokens that do not return a boolean (e.g., USDT on Ethereum mainnet). In that case, the call to `transferFrom()` will revert. Since the function is to recover any accidentally transferred ERC-20 tokens, it would be better to make the function compatible with those ERC-20 tokens that do not follow the standard.

Recommendation: Consider using the `safeTransferFrom()` function from `SafeERC20`.

Sablier: Fixed in PR 405. There was indeed an issue with tokens like USDT. We're now using the `safe` version, and we've renamed it to `transferTokens` to avoid confusion with `ERC721.transferFrom`.

Cantina Managed: Verified.

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

Severity: Low Risk

Context: SablierFlowBase.sol#L196-L210

Description: In the current design, the Flow contract only supports ERC-20-compliant tokens instead of native tokens. Native tokens, as protocol fees, are charged on the UI for specific operations. Such a design could cause issues if the native token of the chain has an ERC-20 representation and is used as the token of a Flow. 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.

As a result, anyone calling `collectFees()` on the Flow contract would accidentally transfer all the CELO tokens in the contract to the admin. This is because in `collectFees()`, the entire `address(this).balance` is transferred. As those CELO tokens are deposited for some Flows, this would affect the withdrawals of the Flow recipients.

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 Flows.

Sablier: Fixed in PR 405. 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.

3.2 Informational

3.2.1 NatSpec and various improvements

Severity: Informational

Context: ISablierFlow.sol#L13, ISablierFlow.sol#L114-L115, ISablierFlow.sol#L229, ISablierFlow.sol#L405, Errors.sol#L19-L20, Errors.sol#L67-L68, Helpers.sol#L8, Helpers.sol#L20

Description:

1. ISablierFlow.sol: All the non-view functions defined in the `ISablierFlow` interface, except `transferFrom()`, emits a `MetadataUpdate()` event. Consider update the NatSpec of the functions accordingly.
2. ISablierFlow.sol#L114-L145: Consider updating the second sentence to.

   > If the total debt exceeds the stream balance, it returns 0.

   to match the actual behavior of the `depletionTimeOf()` function.
3. ISablierFlow.sol#L229: To be consistent with the NatSpec of the `create()` function, consider adding a comment.

   > A value of zero means the stream will be created with the snapshot time as `block.timestamp`.
4. ISablierFlow.sol#L405: Since an approved third party of the Flow NFT is allowed to set the recipient of an withdrawal to any address, consider updating the comment to.

   > /// - `to` must be the recipient if `msg.sender` is neither the stream's recipient nor an approved third party.
5. Errors.sol#L19-L20: Consider updating the NatSpec of the `SablierFlow_InvalidTokenDecimals()` error to.

   > Thrown when trying to create a stream with a token with decimals greater than 18.
6. Errors.sol#L67-L68: For clarity, consider updating the NatSpec of the `SablierFlow_Unauthorized()` error to.

   > /// @notice Thrown when `caller` lacks authorization to perform an action.
7. Helpers.sol#L8: Consider adding the following comment to the NatSpec of `descaleAmount()` the function:

   > `decimals` should be less than or equal to 18.
8. Helpers.sol#L20: Consider adding the following comment to the NatSpec of `scaleAmount()` the function:

   > `decimals` should be less than or equal to 18. Note: The scaled result may overflow `uint256`. If `amount` fits into `uint128`, the result is guaranteed not to overflow.

   The following issue is identified from at commit db1d213:
9. SablierFlowBase.sol#L67-L82: The NatSepc for the `notVoided()` modifier is incorrect. "Not voided" does not imply "not paused", while "not paused" implies "not voided." Consider removing the 2nd sentence for `notVoided()`. Also, add the following comment to `notPaused()`.

   > Note that this implicitly checks that the stream is not voided either.

Recommendation: Consider implementing the above suggestions.

Sablier: We've incorporated all of them except for the 6th one, as the current version is consistent with how we refer to the caller as `msg.sender` throughout `ISablierFlow`. The changes can be seen in PR 405, commit c28b33fc, and PR 429.

Cantina Managed: Verified.

3.2.2 Security considerations for the recovery function

Severity: Informational

Context: SablierFlowBase.sol#L212-L225

Description: A `recover()` function is introduced to recover ERC-20 tokens when necessary, e.g., accidental transfers. The Flow contract keeps track of the balance of each ERC-20 token by an `aggregateBalance` mapping, which is updated whenever ERC-20 tokens are transferred from or to the contract.

It should be noted that the `recover()` function does not support double-entry tokens, i.e., tokens that have two entry points (contracts) where users can interact with any of them to transfer tokens and call any other ERC-20 functions. In the past, SNX, sBTC, and TUSD were double-entry tokens that caused integration issues with DeFi protocols (check the balancer forum entry on truesd and truesd compound vulnerability post by chainsecurity). In this case, if a Flow is deposited with a double-entry token from an entry point A, it would be possible to recover all the token balance using the second entry point B, which would affect the ongoing Flow.

Recommendation: Consider updating the documentation (also the Assumptions section in `SECURITY.md`) that a double-entry token is not supported. Generally, when recovering tokens, the protocol admin should be cautious of the side effects the token transfer would cause.

Sablier: Fixed in PR 405. We've included in the `Assumptions` section that we only support single entry point tokens.

Cantina Managed: Verified.