Mitigation Review For
Sablier Core (abf7154d5371ab957b86fce9a8a4801499573d63)
&
Sablier Periphery (f9defaeb185360d09abba3f7e2f748d993063296)

Version 1.0

28.06.2024

Conducted by: nmirchev8 , SR
deth , SR

Table of Contents

1 About Egis Security

We are a team of experienced smart contract researchers, who strive to provide the best smart contract security services possible to DeFi protocols.

Both members of Egis Security have a proven track record on public auditing platforms such as Code4rena, Sherlock & Codehawks, uncovering more than 100 High/Medium severity vulnerabilities, with >$70,000 in winnings and multiple solo/team audits.

2 Disclaimer

Audits are a time, resource, and expertise bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to identify as many vulnerabilities as possible. Audits can show the presence of vulnerabilities **but not their absence**.

3 Risk classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

3.1 Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - only a small amount of funds can be lost or a functionality of the protocol is affected.
- **Low** - any kind of unexpected behaviour that's not so critical.

3.2 Likelihood

- **High** - direct attack vector; the cost is relatively low to the amount of funds that can be lost.
- **Medium** - only conditionally incentivized attack vector, but still relatively likely.
- **Low** - too many or too unlikely assumptions; provides little or no incentive.

3.3 Actions required by severity level

- **Critical** - client **must** fix the issue.
- **High** - client **must** fix the issue.
- **Medium** - client **should** fix the issue.
- **Low** - client **could** fix the issue.

4 Executive summary

Overview

Project Name	Sablier Core
Repository	https://github.com/sablier-labs/v2-core/tree/staging
Commit hash	from 60a8e60d328b5987121e69b3ed1adaf01d3806ac to abf7154d5371ab957b86fce9a8a4801499573d63
Resolution	-
Documentation	https://docs.sablier.com/
Methods	Manual review

Project Name	Sablier Periphery
Repository	https://github.com/sablier-labs/v2-periphery/tree/staging
Commit hash	from 46ca3bc1253057d9fefa0646e5d48d01d0319375 to f9defaeb185360d09abba3f7e2f748d993063296
Resolution	9eaac3403cad8a5a8f5f4f8a35daeccf3519a160
Documentation	https://docs.sablier.com/
Methods	Manual review

Scope

v2-core/src/abstract/SablierV2Lockup.sol
v2-core/src/SablierV2NFTDescriptor.sol
v2core/src/SablierV2LockupLinear.sol
v2core/src/interfaces/ISablierLockupRecipient.sol
v2-periphery/src/SablierV2MerkleLockupFactory.sol
v2-periphery/src/SablierV2MerkleLT.sol

Issues Found

Critical risk	0
High risk	0
Medium risk	1
Low risk	2
Informational	1

5 Findings

5.1 Medium risk

5.1.1 One-way recipient whitelisting holds a risk, which may influence different users

Severity: *Medium risk*

Context: SablierV2Lockup.sol#L582-L594

Description: It is a combination of the one-way action of allowing recipients and and the fact that anyone can transfer his/her nft to one of the recipients (depending on hook call), which may have high impact lookup cancellation for that specific stream, or mess the accouning for the recipient. We understand that recipients are assumed to implement hooks correct, but there could be a problem with an honest whitelisted recipient and malicious actor, who has noticed their strict implementation.

- Imagine a recipient, who has implemented withdraw/cancel hooks and the code check whether streamId matches an id that the store in their storage. If it doesn't, it reverts.
- Only one such address in allowToHooks is enough to brick entire cancellation logic of the protocol
- Users would be able to front-run sender's cancel transactions by transfering their nft to that recipient (they lose nothing, because otherwise sender will withdraw the remaining funds, making the NFT useless at this point)
- The Cancel transaction will revert due to the recipient's strict hook check
- Sender is not able to cancel his stream and withdraw the funds

Recommendation: To mitigate following concern, we thought of: - implement additional function in `ISablierLockupRecipient`, which return gas required for executing corresponding hook. Then you can check it's value against `gasleft()` in function execution. This way you could safely wrap the hook call in `try/catch` and emit event in the `catch` block - do not allow transfers to whitelisted recipient (this solutions limits functionality) - If you really want to hold current design, implement example hooks with great attention to ensuring hooks do NOT revert under ANY circumstances (checks, underflow, divisions by 0) and should always return the correct selector. You should strictly audit each new recipient contract and pay attention to potential low-level reverts.

Resolution: Acknowledged

5.2 Low risk

5.2.1 Consider making arguments in `AllowToHook` event `indexed`

Severity: *Low risk*

Context: ISablierV2Lockup.sol#L27

Description: This way if the admin is changed, it would be easy to query all addresses that were whitelisted by given admin address, etc.

Recommendation: Make arguments of the events `indexed`

Resolution: Fixed in 9eaac3403cad8a5a8f5f4f8a35daeccf3519a160

5.2.2 `SablierV2LockupLinear.sol#_calculateStreamedAmount()` - cliffTime and startTime checks can be more strict in order to skip unnecessary calculations.

Severity: *Low risk*

Context: SablierV2LockupLinear.sol#L195-L197

Description: If `cliffTime == blockTimestamp` or `startTime == blockTimestamp`, the calculation for `streamedAmount` will be executed, which in the end will return 0. This is unnecessary as the function will just waste more gas to return the same value in the end.

Recommendation: Make the two checks >= instead of > like they are done in `SablierV2LockupDynamic`

Resolution: Fixed in cf2c46a206dc67a2280c9e98b53591e6de0ee504

5.3 Informational

5.3.1 Add – as supported char in `isAlphanumericWithSpaces`

Severity: *Info risk*

Context: SablierV2NFTDescriptor.sol#L325

Description: Consider adding – (2D) as supported char in for token sybmol, because it is not a threat for JSON injection and there are tokens such as LP, which may use this char. Example token **Recommendation:** Add `bool isDash = char == 2D` check

Resolution: Fixed in 9eaac3403cad8a5a8f5f4f8a35daeccf3519a160