LI.FI Security Review

LiFiIntentEscrowFacet.sol(v1.1.0)

Security Researcher

Sujith Somraaj (somraajsujith@gmail.com)

Report prepared by: Sujith Somraaj

January 30, 2026

Contents

1 About Researcher

Sujith Somraaj is a distinguished security researcher and protocol engineer with over eight years of comprehensive experience in the Web3 ecosystem.

In addition to working as a Lead Security Researcher at Spearbit, Sujith is also the security researcher and advisor for leading bridge protocol LI.FI and also is a former founding engineer and current security advisor at Superform, a yield aggregator with over $170M in TVL.

Sujith has experience working with protocols / funds including Coinbase, Uniswap, Layerzero, Edge Capital, Berachain, Optimism, Ondo, Sonic, Monad, Blast, ZkSync, Decent, Drips, SuperSushi Samurai, DistrictOne, Omni-X, Centrifuge, Superform-V2, Tea.xyz, Paintswap, Bitcorn, Sweep n' Flip, Byzantine Finance, Variational Finance, Satsbridge, Rova, Horizen, Earthfast and Angles

Learn more about Sujith on sujithsomraaj.xyz or on cantina.xyz

2 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release, and does not give any warranties on finding all possible security issues of that given smart contract(s) or blockchain software. i.e., the evaluation result does not guarantee against a hack (or) the non existence of any further findings of security issues. As one audit-based assessment cannot be considered comprehensive, I always recommend proceeding with several audits and a public bug bounty program to ensure the security of smart contract(s). Lastly, the security audit is not an investment advice.

This review is done independently by the reviewer and is not entitled to any of the security agencies the researcher worked / may work with.

3 Scope

- src/Facets/LiFiIntentEscrowFacet.sol(v1.1.0)

4 Risk classification

Severity level	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

4.1 Impact

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.

4.2 Likelihood

High	almost certain to happen, easy to perform, or not easy but highly incentivized
Medium	only conditionally possible or incentivized, but still relatively likely
Low	requires stars to align, or little-to-no incentive

4.3 Action required for severity levels

Critical	Must fix as soon as possible (if already deployed)
High	Must fix (before deployment if not already deployed)
Medium	Should fix
Low	Could fix

5 Executive Summary

Over the course of 3 hours in total, LI.FI engaged with the researcher to audit the contracts described in section 3 of this document ("scope"). This is a differential review focussed on the changes from previous version.

In this period of time a total of 2 issues were found.

Project Summary	
Project Name	LI.FI
Repository	lifinance/contracts
Commit	151a2a
Audit Timeline	January 20, 2025
Methods	Manual Review
Documentation	High
Test Coverage	High

Issues Found	
Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	1
Gas Optimizations	0
Informational	1
Total Issues	**2**

6 Findings

6.1 Low Risk

6.1.1 Inconsistent refund recipient

Context: LiFiIntentEscrowFacet.sol#L112, LiFiIntentEscrowFacet.sol#L120, LiFiIntentEscrowFacet.sol#L127

Description: In `swapAndStartBridgeTokensViaLiFiIntentEscrow()`, there is inconsistent handling of refund recipients. Positive slippage is correctly sent to depositAndRefundAddress, but excess native ETH and swap leftovers are sent to `msg.sender`, which in some cases could be a relayer / forwarder.

Refund Type	Current Recipient	Expected Recipient
Positive slippage	depositAndRefundAddress	depositAndRefundAddress
Excess native ETH	msg.sender	depositAndRefundAddress
Swap leftovers	msg.sender	depositAndRefundAddress

Recommendation: Consider enforcing uniform refund recipient across all code paths.

LI.FI: Resolved in df0c3c2

Recommendation implemented except for refundExcessNative(payable(msg.sender)). Currently, every single facet uses this refund flow for excessNative. "Fixing" the flow on a single facet would cause implementation fragmentation and make it hard to track which facets have been upgraded. Fixing this across all facets would not be a minor change.

Researcher: Verified fix.

6.2 Informational

6.2.1 Misleading error message for zero `depositAndRefundAddress()` validation

Context: LiFiIntentEscrowFacet.sol#L149-L150

Description: In LiFiIntentEscrowFacet._startBridge(), the validation for depositAndRefundAddress reverts with `InvalidReceiver()` when the address is zero:

```
  // src/Facets/LiFiIntentEscrowFacet.sol:149-150
  if (_lifiIntentData.depositAndRefundAddress == address(0))
      revert InvalidReceiver();
```

The depositAndRefundAddress field serves as:

1. The user in StandardOrder (the depositor)
2. The recipient of refunds if the intent expires/fails
3. The recipient of positive slippage from swaps

This is semantically different from the receiver/recipient, which is the destination address for the bridged assets. Using InvalidReceiver() for this validation is misleading and makes debugging harder for integrators who might look for issues with the wrong field.

Recommendation: Use a more specific error or add a dedicated error for this validation:

```
error InvalidDepositAndRefundAddress();

  // In _startBridge:
  if (_lifiIntentData.depositAndRefundAddress == address(0))
      revert InvalidDepositAndRefundAddress();
```

LI.FI: Fixed in df0c3c

Researcher: Verified fix.