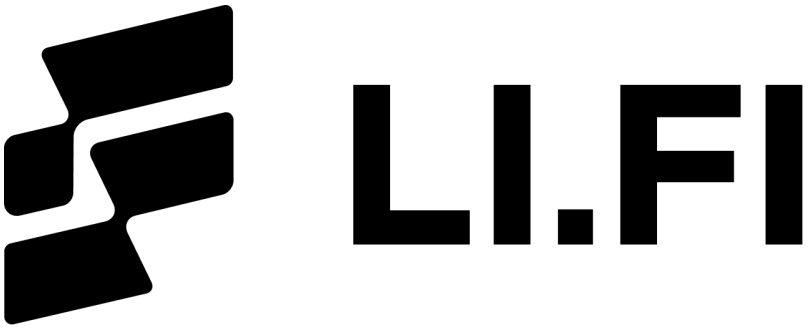

LI.FI Security Review

LiFiIntentEscrowFacet.sol(v1.0.0)

Security Researcher

Sujith Somraaj (somraajsujith@gmail.com)

Report prepared by: Sujith Somraaj

November 19, 2025

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, 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(v1.0.0)
- src/Interfaces/IOpenIntentFramework.sol(v1.0.0)
- src/Interfaces/IOriginSettler.sol(v1.0.0)

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 11 hours in total, LI.FI engaged with the researcher to audit the contracts described in section 3 of this document ("scope").

In this period of time a total of 6 issues were found.

Project Summary	
Project Name	LI.FI
Repository	lifinance/contracts
Commit	4e71699
Audit Timeline	November 17, 2025 - November 19, 2025
Methods	Manual Review
Documentation	High
Test Coverage	High

Issues Found	
Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	3
Gas Optimizations	0
Informational	3
Total Issues	**6**

6 Findings

6.1 Low Risk

6.1.1 Misleading event emission for destination calls via OIF callbackData

Context: LiFiIntentEscrowFacet.sol#L158

Description: The **LiFiIntentEscrowFacet** contract enforces the `doesNotContainDestinationCalls(_bridgeData)` modifier on both bridge functions, which requires **_bridgeData.hasDestinationCall** to be false. However, the contract simultaneously allows users to specify arbitrary calldata in **_lifiIntentData.outputCall** that will be executed on the destination chain after token delivery.

Per Open Intent Framework (OIF) documentation and the contract's own comments, outputCall is "calldata to be executed after the token has been delivered" on the receiverAddress.

This creates a discrepancy where:

1. Event claims: No destination call (_bridgeData.hasDestinationCall = false)
2. Actual behavior: Destination call executes if outputCall.length > 0

Recommendation: Synchronize Flag with Actual Behavior. Remove the doesNotContainDestinationCalls modifier and update the hasDestinationCall flag based on actual behavior.

LI.FI: Fixed in 67d51637

Researcher: Verified fix.

6.1.2 No validation of `outputAmount`

Context: LiFiIntentEscrowFacet.sol#L156

Description: The **outputAmount** is not validated and could be zero. If it's zero, the solver could steal all deposited funds by sending zero tokens to the user.

Recommendation: Add validation:

```
if (_lifiIntentData.outputAmount == 0) revert InvalidAmount();
```

LI.FI: Fixed in e769f33

Researcher: Verified fix.

6.1.3 No validation of `depositAndRefundAddress`

Context: LiFiIntentEscrowFacet.sol#L165

Description: The **depositAndRefundAddress** is not validated and could be `address(0)`. If it's zero, refunds would fail or be lost.

Recommendation: Add validation:

```
if (_lifiIntentData.depositAndRefundAddress == address(0)) revert InvalidReceiver();
```

LI.FI: Fixed in 7346115.

Researcher: Verified fix.

6.2 Informational

6.2.1 Incompatability with non-evm chains

Context: LiFiIntentEscrowFacet.sol#L132

Description: The LiFi intents protocol supports non-evm bridging to chains like Solana, Bitcoin, and others. But in the facet, it is not handled as expected and will either result in loss of funds if bridged to non-evm chains (or) will revert.

For bridging to non-evm chains, the `bridgeData.receiver` is set to a constant, and the logic is handled differently in other facets.

Recommendation: Consider fixing the `_startBridge()` function to accomodate for non-evm chain bridging.

LI.FI: We're going to launch only on EVM chains in this version and add an update later.

Researcher: Acknowledged.

6.2.2 No validation of `MandateOutput` parameter

Context: LiFiIntentEscrowFacet.sol#L150-L159

Description: The **outputOracle** and **outputSettler** bytes32 values are not validated. They could be zero or invalid. In most cases, it'll trigger a refund to the user. But validating them could save users from unexpected outcomes.

Recommendation: Validate if the above-mentioned values in the MandateOutput parameter are not zero.

LI.FI: Researcher:

6.2.3 Unlimited token approval

Context: LiFiIntentEscrowFacet.sol#L140-L144

Description: The **LiFiIntentEscrowFacet** uses `maxApproveERC20()` which sets `type(uint256).max` approval if the current allowance is insufficient. This means the settler contract has unlimited approval to spend all tokens, not just the intended amount.

If the settler contract is compromised or has a vulnerability, all approved tokens could be drained.

Recommendation: Use exact approval amounts instead:

```
LibAsset.approveERC20(
      IERC20(sendingAsset),
      address(LIFI_INTENT_ESCROW_SETTLER),
      amount,
      amount  // Set exact allowance, not max
  );
```

LI.FI: Acknowledged.

Researcher: Acknowledged.