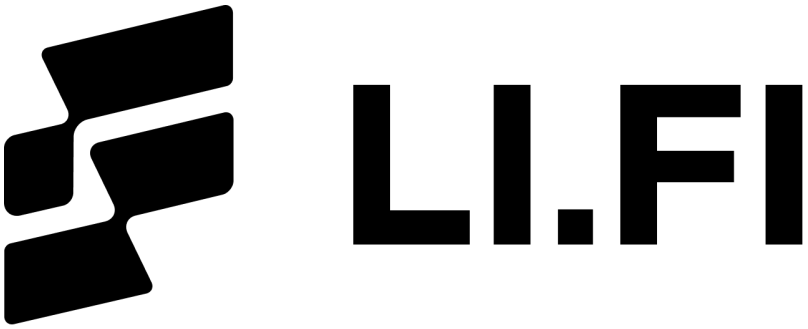

# LI.FI Security Review

ReceiverOIF.sol(v1.0.0)

**Security Researcher**

Sujith Somraaj (somraajsujith@gmail.com)

**Report prepared by:** Sujith Somraaj

December 15, 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, 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/Periphery/ReceiverOIF.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 5 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 | 2f49105451 |
| Audit Timeline | December 12, 2025 |
| Methods | Manual Review |
| Documentation | High |
| Test Coverage | Medium |

| Issues Found | |
|---|---|
| Critical Risk | 0 |
| High Risk | 0 |
| Medium Risk | 0 |
| Low Risk | 1 |
| Gas Optimizations | 0 |
| Informational | 5 |
| **Total Issues** | **6** |

# 6 Findings

## 6.1 Low Risk

### 6.1.1 Missing zero-address validation for decoded receiver in `ReceiverOIF.outputFilled()`

**Context:** ReceiverOIF.sol#L75

**Description:** The `outputFilled()` function in `ReceiverOIF.sol` decodes a receiver address from the **executionData** parameter without validating that it is not the zero address:

```
function outputFilled(
  bytes32 token,
  uint256 amount,
  bytes calldata executionData
) external onlyTrustedOutputSettler {
      // decode payload
      (
          bytes32 transactionId,
          LibSwap.SwapData[] memory swapData,
          address receiver  // @audit no zero-address check
      ) = abi.decode(executionData, (bytes32, LibSwap.SwapData[], address));

      // execute swap(s)
      _swapAndCompleteBridgeTokens(
          transactionId,
          swapData,
          address(uint160(uint256(token))),
          payable(receiver),  // @audit receiver could be address(0)
          amount
      );
  }
```

The decoded receiver is subsequently passed to the `EXECUTOR.swapAndCompleteBridgeTokens()` function. If the downstream Executor contract does not validate that the receiver is not address(0), tokens will be permanently sent to the zero address (effectively burned).

**Recommendation:** Add explicit validation in the `outputFilled()` function to ensure the decoded receiver address is not the zero address. This will refund the tokens to the user's source chain address, resulting in a more safer approach.

**LI.FI:** Fixed in 1ecae5a

**Researcher:** Verified fix.

## 6.2 Informational

### 6.2.1 Slippage protection must be handled at `SwapData` level

**Context:** Global

**Description:** The `ReceiverOIF` contract does not enforce any minimum output amount or slippage protection.

Users must ensure slippage protection is embedded within the swap calldata they provide (depending on the swapTo address). Failure to do so can result in receiving zero tokens or suffering from extreme slippage with no transaction revert.

**Recommendation:** Add warning to contract natspec in the `outputFilled()` function on `ReceiverOIF.sol` contract.

**LI.FI:** Documentation added in c6ead50

**Researcher:** Added documentation looks good.

#### 6.2.2 Approval to executor not reset to zero after swap

**Context:** ReceiverOIF.sol#L115

**Description:** The `_swapAndCompleteBridgeTokens()` function in `ReceiverOIF.sol` approves tokens to the `EXECUTOR` contract, but never resets the approval to zero after the swap execution completes. This leaves residual approvals that could be exploited if the Executor contract is compromised or has vulnerabilities.

**Recommendation:** Consider resetting approvals after the executor call.

**LI.FI:** The `ReceiverOIF.sol` cannot hold tokens long-term since you can easily craft an external call that withdraws the tokens sent to this contract. As a result, whether the Executor can withdraw these tokens is insignificant.

Resetting the approval would be a waste of gas. Additionally, the approve call uses safeApproveWithRetry, which can handle tokens where approvals are set to 0.

**Researcher:** Acknowledged.

#### 6.2.3 TransactionId pollution can affect off-chain indexers

**Context:** ReceiverOIF.sol#L72

**Description:** The `ReceiverOIF.outputFilled()` function allows user-controlled `transactionId` values to be emitted in `LiFiTransferCompleted` events, which can pollute off-chain indexers and analytics systems that rely on transaction ID uniqueness for tracking cross-chain transfers.

This could result in:

- Indexers relying on `transactionId` uniqueness will have data consistency issues
- Analytics dashboards may show incorrect transaction counts or duplicate entries - Bridge monitoring systems could misattribute transactions - Historical data queries may return ambiguous or incorrect results

**Recommendation:** Consider tracking used transaction IDs and rejecting duplicates.

**LI.FI:** Acknowledged. When we are integrating the facet, we will take into consideration that other people are able to issue events with fraudulent transaction IDs.

**Researcher:** Acknowledged.

#### 6.2.4 Incomplete test suite

**Context:** Global

**Description:** The following test cases are not found in the `ReceiverOIF.t.sol` file:

1. Native token swap success case 1.1 `receive()` function test (explicit functionality)
2. Empty swapData array (common edge case)
3. Multiple swaps in swapData
4. Edge cases (amount=0, receiver=0, invalid encoding)

A comprehensive test suite is essential for ensuring the code works as expected under different execution paths.

**Recommendation:** Add to test suite, the above mentioned cases and ensure the code has full coverage.

**LI.FI:** Partially fixed in e01422c. Multiple swaps in the swapData test case are not covered. The execution of swaps within the executor (including multiple swaps in swapData) is not in the scope of this contract. There could indeed be an encoding edge, but I would not expect this to differ from existing receivers.

**Researcher:** Fix verified.

### 6.2.5 Typos

**Context:** ReceiverOIF.sol#L29

**Description:**

```
- /// Anyone can call this contract with fradulently filled intents, incoming outputFilled calls cannot
↪  be trusted.
+ /// Anyone can call this contract with fraudulently filled intents, incoming outputFilled calls cannot
↪  be trusted.
```

**Recommendation:** Consider fixing the above mentioned typo.

**LI.FI:** Fixed in c6ead50

**Researcher:** Verified fix.