LI.FI Security Review

RelayFacet(v1.0.0), LibAsset(v1.0.1)

Independent Review By:

Sujith Somraaj (somraajsujith@gmail.com)

December 2, 2024

Contents

1 About Researcher

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

In addition to working as an external auditor/security researcher with LI.FI, Sujith is a protocol engineer and security researcher at Superform and Spearbit.

Learn more about Sujith on sujithsomraaj.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/RelayFacet.sol
- src/Libraries/LibAsset.sol

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 1 days 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 8 issues were found.

Project Summary	
Project Name	LI.FI
Repository	lifinance/contracts
Commit Hashes	291d0a78bc.....6b5d3e8ec8
Type of Project	
Audit Timeline	November 19, 2024
Methods	Manual Review

Issues Found	
Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	4
Gas Optimizations	0
Informational	4
Total Issues	**8**

6 Findings

6.1 Low Risk

6.1.1 Validate if `_bridgeData.sendingAssetId` equals swap output token

Context: RelayFacet.sol#L121

Description: The **swapAndStartBridgeTokensViaRelay** function swaps the sendingAssetId to an intermediary asset before calling the relay bridge.

However, the final output of the swaps could be a token different from the **_bridgeData.sendingAssetId**, leading to unexpected behavior. In most cases, the call should revert, and the user funds will be safe, but it's ideal to add a check for sanity.

Recommendation: Consider validating if the `_swapData[_swapData.length -1].receivingAssetId` equals the `_bridgeData.sendingAssetId` in the **swapAndStartBridgeTokensViaRelay** function.

LI.FI: This is fine, as we are optimizing for gas, and this should revert anyway.

Researcher: Agree on the gas optimization argument. Still, this will revert down the stack without a proper error message, which might result in a degraded experience.

6.1.2 Use `safeTransferETH` to transfer native asset

Context: LibAsset.sol#L48

Description: The **transferNativeAsset** function in **LibAsset** is used to transfer native tokens using the `call` method. However, switching to **safeTransferETH** from the **SafeTransferLib** is considered generally safe and gas efficient.

Recommendation: Consider using **safeTransferETH** from solady library to send native tokens.

LI.FI: We are already planning to make this update later.

Researcher: Acknowledged.

6.1.3 Refunds are sent to the `diamond` contract instead of the user

Context: RelayFacet.sol#L18, demoRelay.ts#L36

** Description: ** The RelayBridge has a complex refunding mechanism (`https://docs.relay.link/how-it-works/refunds`) for multiple reasons, including invalid quotes, expired quotes, wallet addresses different from the depositing address, deposits being too little (or) the destination chain becoming unavailable, and many more.

During these times, refunds occur on either the source chain or the destination chain and are sent to either the user address (on the source chain) or the recipient address (on the destination chain).

If the refunds happen on the source chain, the diamond contract always receives the refund, which can be risky. Though this is not an intelligent contract risk, as an integrator, it is essential to refund the user for failed bridging actions.

Recommendation: To avoid refund confusion, Consider using the **refundTo** and **refundOnOrigin** parameters of the `/quote` route to generate the transaction request on relay.

Refunding on source chain is ideal as the user might not have gas on destination chain / would lead to sub-optimal user experience.

LI.FI: I think this should be a low as we can't really force this behavior on the contract side. We will just need to ensure that we set that correctly in the API call.

Researcher: Agreed. The issue has to be fixed on the API level. No safety measures exist / could be implemented on smart contract against this issue.

6.1.4 Redundant `receiver` address checks while bridging to Solana & Bitcoin

Context: RelayFacet.sol#L106, RelayFacet.sol#L133

Description: The functions **startBridgeTokensViaRelay** and **swapAndStartBridgeTokensViaRelay** bridge tokens using the relay facet. These two functions depend on the **validateBridgeData** modifier to ensure the correctness of user input.

It validates if the **_bridgeData.receiver** is not a zero address. However, in the case of bridging to Solana (or) Bitcoin, the receiver address is passed in the **_relayData.nonEVMReceiver** and is allowed to be zero, making the **validateBridgeData** check redundant.

Recommendation: Consider implementing additional checks to ensure the passed-in **nonEVMReceiver** of relay data is not empty.

LI.FI: Fixed in f285130f63f4ef8b1d8c94605ee5eb069c7eca19

Researcher: Verified fix.

6.2 Informational

6.2.1 Emit mapped destination chain id in `BridgeToNonEVMChain` event

Context: RelayFacet.sol#L189

Description: The **BridgeToNonEVMChain** is emitted whenever the relay facet is used to bridge tokens to either Solana (or) Bitcoin. The event includes a destination chain parameter, which is the same as the destination chain emitted by the **LiFiTransferStarted** event.

Recommendation: It is more logical to emit the mapped destination chain ID in this event (**BridgeToNonEVMChain**).

LI.FI: Fixed in 2efd99ef72c9b7c752d5dd3c16d7089828baaae3

Researcher: Verified fix.

6.2.2 Prevent `requestId` replay

Context: RelayFacet.sol#L97, RelayFacet.sol#L121

Description: The **swapAndStartBridgeTokensViaRelay** and **startBridgeTokensViaRelay** functions are used to bridge tokens using the relay bridge. The two functions accept a **RelayData** parameter that includes the **requestId**.

The **requestId** is a unique identifier associated with individual quotes generated using the relay API. Since the **requestId** is usually passed to the relayReceiver (or) relaySolver in the `msg.data`, there is possibly no validation if it's unique on the relay side.

If the funds are already used, they are refunded to the user. This can be avoided if the RelayFacet prevents replay on its own.

Recommendation: Add a mapping to store the request IDs already consumed by the Facet to avoid replay.

LI.FI: Fixed in 4b3535fbbd6a3345f9f82900668a8c575048d562

Researcher: Verified fix.

6.2.3 `relaySolver` can be updated by the bridge leading to re-deployment of the facet

Context: RelayFacet.sol#L22

Description: The `RelayFacet.sol` contract currently initializes the **relaySolver** address in the constructors and makes it immutable. This address primarily receives the ERC20 tokens to initiate the bridging process.

However, this address is configured on the relay bridge's API and could be upgraded in the future for any reason. If that happens, then the current facet has to be redeployed.

Recommendation: Make the `relaySolver` configurable by the diamond admin.

LI.FI: This is acceptable, and we would be fine with redeploying the contract to an updated address.

Researcher: Acknowledged.

6.2.4 Remove unused `LibDiamond` import

Context: RelayFacet.sol#L5

Description: The **RelayFacet** imports **LibDiamond**, but is not used anywhere in the context. Unused imports could impact code quality.

Recommendation: Consider removing the unused import **LibDiamond**

LI.FI: Fixed in a1fcd9a1fd41f5b9e5ff4d886c6230b2af3cd86c

Researcher: Verified fix.