LI.FI Security Review

GnosisBridgeFacet.sol(v2.0.0), IGnosisBridgeRouter.sol(v1.0.0)

Security Researcher

Sujith Somraaj (somraajsujith@gmail.com)

Report prepared by: Sujith Somraaj

June 3, 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 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 CISO at Superform, a yield aggregator with over $170M in TVL.

Sujith has experience working with protocols including Berachain, Optimism, Fantom, 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, 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/GnosisBridgeFacet.sol(v2.0.0)
- src/Interfaces/IGnosisBridgeRouter.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 3 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 2 issues were found. This review focussed only on the changes made from the previous version, not the code on its entirety.

Project Summary	
Project Name	LI.FI
Repository	lifinance/contracts
Commit	7197c0a237f4....5ca156ed
Audit Timeline	June 02, 2025
Methods	Manual Review
Documentation	Medium
Test Coverage	Low-Medium

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

6 Findings

6.1 Informational

6.1.1 Unlimited token approval (Mitigated by Architecture)

Context: GnosisBridgeFacet.sol#L104

Description: The contract grants unlimited approval (maxApproveERC20) to the Gnosis Bridge Router during every bridge transaction.

Since this contract doesn't hold funds persistently (funds flow through transiently), unlimited approvals are primarily not a significant issue; however, it could be re-architected to approve the router on demand to avoid exposure.

Recommendation: Consider approving the router for the exact amount, rather than maximum approvals.

LI.FI: We acknowledge this finding. It's an intentional design choice across all facets. Since these are contracts where funds flow through in atomic transactions, and given the gas efficiency benefits, the current implementation with max approvals is ok.

Researcher: Acknowledged.

6.1.2 Validate if `_swapData.receivingAssetId` of final swap step equals `bridgeData.sendingAssetId`

Context: GnosisBridgeFacet.sol#L90

Description: The `swapAndStartBridgeTokensViaGnosisBridge()` function lacks validation to ensure that the final swap step produces the token expected by the bridge. Currently, there's no check to verify that `_swapData[_swapData.length - 1].receivingAssetId` matches `_bridgeData.sendingAssetId`.

The contract validates that `_bridgeData.sendingAssetId` is either DAI or USDS, but doesn't validate that the swap chain produces this token.

This creates a mismatch where:

- User provides swap data that outputs Token X
- Bridge data expects to send Token Y (DAI/USDS)
- No validation catches this inconsistency

Recommendation: Consider adding a check to ensure the function validates the swap data's final receiving against the following:

```
function swapAndStartBridgeTokensViaGnosisBridge(
   ILiFi.BridgeData memory _bridgeData,
   LibSwap.SwapData[] calldata _swapData
)  external
   payable
   nonReentrant
   refundExcessNative(payable(msg.sender))
   containsSourceSwaps(_bridgeData)
   doesNotContainDestinationCalls(_bridgeData)
   validateBridgeData(_bridgeData)
   onlyAllowDestinationChain(_bridgeData, GNOSIS_CHAIN_ID)
{
   ....
+ if(_bridgeData.sendingAssetId != _swapData[_swapData.length - 1].receivingAssetId) {
+    revert InvalidSendingToken();
+ }
   ....
```

LI.FI: Fixed in 05ff21c84a27e110a71960f26b44499e33ef5fee

Researcher: Verified fix.