LI.FI Security Review

CalldataVerificationFacet (v1.3.0)

Independent Review By:

Sujith Somraaj (somraajsujith@gmail.com)

February 26, 2025

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 security researcher at Spearbit and a former founding-engineer at Superform, an yield aggregator with over $100M in TVL.

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

- Facets/CalldataVerificationFacet.sol(v1.3.0)

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 2.5 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 2 issues were found.

Project Summary	
Project Name	LI.FI
Repository	lifi/contracts
Commit	48427d2116.....6bd42c4c03
Audit Timeline	February 20, 2025
Methods	Manual Review

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

6 Findings

6.1 Gas Optimization

6.1.1 Optimise `extractGenericSwapParameters` by avoiding copying `data` parameter to memory

Context: CalldataVerificationFacet.sol#L159

Description: The `extractGenericSwapParameters` copies the `data` param to a new local memory variable and operate on it, which turns out to be gas expensive and is unnecessary.

```
function extractGenericSwapParameters(
  bytes calldata data
)
public
pure
returns (
    address sendingAssetId,
    uint256 amount,
    address receiver,
    address receivingAssetId,
    uint256 receivingAmount
) {
     /// ...

     LibSwap.SwapData[] memory swapData;
     bytes memory callData = data; /// [1] - unwanted copy to memory
     bytes4 functionSelector = bytes4(data[:4]);

     /// ...
    (, , , receiver, receivingAmount, swapData[0]) = abi.decode(
        callData.slice(4, callData.length - 4), /// [2] - unwanted memory slice
        (bytes32, string, string, address, uint256, LibSwap.SwapData)
    );
    /// ...
   (, , , receiver, receivingAmount, swapData) = abi.decode(
          callData.slice(4, callData.length - 4), /// [3] - unwanted memory slice
          (bytes32, string, string, address, uint256, LibSwap.SwapData[])
   );

    /// ...
}
```

Recommendation: Consider operating on the original parameter without copying it to memory as follows:

```
function extractGenericSwapParameters(
  bytes calldata data
)
public
pure
returns (
    address sendingAssetId,
    uint256 amount,
    address receiver,
    address receivingAssetId,
    uint256 receivingAmount
) {
     /// ...

     LibSwap.SwapData[] memory swapData;
-     bytes memory callData = data;
     bytes4 functionSelector = bytes4(data[:4]);

     /// ...
    (, , , receiver, receivingAmount, swapData[0]) = abi.decode(
-       callData.slice(4, callData.length - 4),
+      data[4:],
        (bytes32, string, string, address, uint256, LibSwap.SwapData)
    );
    /// ...
   (, , , receiver, receivingAmount, swapData) = abi.decode(
-        callData.slice(4, callData.length - 4),
+       data[4:],
          (bytes32, string, string, address, uint256, LibSwap.SwapData[])
   );

    /// ...
}
```

LI.FI: Fixed in d8582587bdbfe6f54490ca12f582182551ae34bc

Researcher: Verified fix

6.1.2 Optimise `validateDestinationCalldata` without copying `data` to memory

Context: CalldataVerificationFacet.sol#L250

Description: The `validateDestinationCalldata` function copies the input parameter `data` to a local memory variable `callData` and operates on it, which is gas expensive.

Rough benchmarks indicate that copying to memory costs nearly 10% more gas.

```
Ran 1 test for test/solidity/Facets/CalldataVerificationFacet.t.sol:CalldataVerificationFacetTest
[PASS] test_CanValidateStargateV2DestinationCalldata() (gas: 128556) /// copying to memory
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 1.78s (881.92µs CPU time)

Ran 1 test for test/solidity/Facets/CalldataVerificationFacet.t.sol:CalldataVerificationFacetTest
[PASS] test_CanValidateStargateV2DestinationCalldata() (gas: 115210) /// operating at calldata level
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 1.93s (1.52ms CPU time)
```

Recommendation: Consider updating the function as follows:

```
function validateDestinationCalldata(
  bytes calldata data,
  bytes calldata callTo,
 bytes calldata dstCalldata
) external pure returns (bool isValid) {
- bytes memory callData = data;
- bytes4 selector = abi.decode(callData, (bytes4));
+ bytes4 selector = bytes4(data[:4]);

   /// ....

  (, StargateFacetV2.StargateData memory stargateDataV2) = abi
     .decode(
-   callData.slice(4, callData.length - 4)
+   data[4:],
    (ILiFi.BridgeData, StargateFacetV2.StargateData)
   );

  /// ... similarly changing slice to calldata slicing in all branches
}
```

LI.FI: Fixed in d8582587bdbfe6f54490ca12f582182551ae34bc

Researcher: Verified fix