LI.FI Security Review

CalldataVerificationFacet(v1.2.0)

Independent Review By:

Sujith Somraaj (somraajsujith@gmail.com)

September 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/CalldataVerificationFacet.sol(v1.2.0)

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 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	374d066a.....aa0bf2ff8
Type of Project	
Audit Timeline	August 28, 2024
Methods	Manual Review

Issues Found	
Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	1
Gas Optimizations	3
Informational	4
Total Issues	**8**

6 Findings

6.1 Low Risk

6.1.1 `extractGenericSwapParameters` should validate if data length is > 484

Context: CalldataVerificationFacet.sol#L167

Description: The `extractGenericSwapParameters` function in the `CalldataVerificationFacet` extracts the generic swap parameters from a calldata.

This function has a length-based sanity check to ensure the passed-in data is valid, which is not accurate. The notion here is that if the data length is less than 484, then the `callData` inside `SwapData` is empty and invalid.

But even if the length of the data passed is 484, then the `callData` is empty. So, the validation should expect data with a length > 484 instead of >= 484.

Recommendation: Consider adapting the validation to revert even if the data length is 484.

```
function extractGenericSwapParameters(
  bytes calldata data
) public pure
   returns (
     address sendingAssetId,
     uint256 amount,
     address receiver,
     address receivingAssetId,
     uint256 receivingAmount
) {
     ....
-    if (callData.length < 484) {
+    if (callData.length <= 484) {
     ....
```

LI.FI: Implemented as suggested: `a4c2574f2f143dd732de02eeecb79db6c4864806`

Researcher: Validated Fix. Looks all good.

6.2 Gas Optimization

6.2.1 `validateCalldata` function could be optimized

Context: CalldataVerificationFacet.sol#L246

Description: The function `validateCalldata` in `CalldataVerificationFacet` could be optimized by

1. Using the `_extractBridgeData` internal function to decode the `data` instead of using `extractMainParameters`.
2. Caching the bridge name hash.

The function can be optimized as follows:

```
    function validateCalldata(
        bytes calldata data,
        string calldata bridge,
        address sendingAssetId,
        address receiver,
        uint256 amount,
        uint256 destinationChainId,
        bool hasSourceSwaps,
        bool hasDestinationCall
    ) external pure returns (bool isValid) {
        ILiFi.BridgeData memory bridgeData = _extractBridgeData(data);

        bytes32 bridgeNameHash = keccak256(abi.encodePacked(bridge));
        return
        // Check bridge
        (
             bridgeNameHash == keccak256(abi.encodePacked(""))
                || keccak256(abi.encodePacked(bridgeData.bridge)) == bridgeNameHash
        )
        // Check sendingAssetId
        && (sendingAssetId == 0xFFfFfFffFFfffFFfFFfFFFFFffFFFffffFfFFFfF || bridgeData.sendingAssetId
        ↪  == sendingAssetId)
        // Check receiver
        && (receiver == 0xFFfFfFffFFfffFFfFFfFFFFFffFFFffffFfFFFfF || bridgeData.receiver == receiver)
        // Check amount
        && (amount == type(uint256).max || bridgeData.minAmount == amount)
        // Check destinationChainId
        && (destinationChainId == type(uint256).max || bridgeData.destinationChainId ==
        ↪  destinationChainId)
        // Check hasSourceSwaps
        && bridgeData.hasSourceSwaps == hasSourceSwaps
        // Check hasDestinationCall
        && bridgeData.hasDestinationCall == hasDestinationCall;
    }
```

Recommendation: Optimize the function as mentioned above to save gas fees for on-chain integrators. Also, if the change mentioned above is made, the `extractMainParameters` visibility could be changed to `external`

```
Before Optimization
[PASS] test_CanValidateCalldata() (gas: 64896)

After Optimization
[PASS] test_CanValidateCalldata() (gas: 61932)
```

LI.FI: Fixed in `51b585a2667308bdb89010bb5f8e4c92c09bb802`

Researcher: Validated fix. Looks all good.

6.2.2 `extractGenericSwapParameters` function could be optimized

Context: CalldataVerificationFacet.sol#L142

Description: The function `extractGenericSwapParameters` can be optimized in two ways to save significant gas.

1. Remove unnecessary writing into memory. (two places)
2. Remove the expensive `_isGenericV3SingleSwap` internal call. Also, this function is not re-used, so it is probably a code quality improvement, too.

The function could be re-arranged like the following:

```
function extractGenericSwapParameters(bytes calldata data)
    public
    pure
    returns (
        address sendingAssetId,
        uint256 amount,
        address receiver,
        address receivingAssetId,
        uint256 receivingAmount
    )
{
    // valid callData for a genericSwap call should have at least 484 bytes:
    // Function selector: 4 bytes
    // _transactionId: 32 bytes
    // _integrator: 64 bytes
    // _referrer: 64 bytes
    // _receiver: 32 bytes
    // _minAmountOut: 32 bytes
    // _swapData: 256 bytes
    if (data.length < 484) {
        revert InvalidCallData();
    }

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

    // check if this is a call via StandardizedCallFacet
    if (functionSelector == StandardizedCallFacet.standardizedCall.selector) {
        // extract nested function selector and calldata
        // will always start at position 68
        functionSelector = bytes4(data[68:72]);
        callData = data[68:];
        // callData = abi.decode(data[4:], (bytes)); // this one is also valid, even though the
        ↪  calldata differs slightly (add. padding)
    } else {
        callData = data;
    }

    if (
        functionSelector == GenericSwapFacetV3.swapTokensSingleV3ERC20ToERC20.selector
            || functionSelector == GenericSwapFacetV3.swapTokensSingleV3ERC20ToNative.selector
            || functionSelector == GenericSwapFacetV3.swapTokensSingleV3NativeToERC20.selector
    ) {
        // single swap
        swapData = new LibSwap.SwapData[](1);

        // extract parameters from calldata
        (,,, receiver, receivingAmount, swapData[0]) = abi.decode(
            callData.slice(4, callData.length - 4), (bytes32, string, string, address, uint256,
            ↪  LibSwap.SwapData)
        );
    } else {
        // multi swap or GenericSwap V1 call
        (,,, receiver, receivingAmount, swapData) = abi.decode(
            callData.slice(4, callData.length - 4), (bytes32, string, string, address, uint256,
            ↪  LibSwap.SwapData[])
        );
    }

    // extract missing return parameters from swapData
```

```
        sendingAssetId = swapData[0].sendingAssetId;
        amount = swapData[0].fromAmount;
        receivingAssetId = swapData[swapData.length - 1].receivingAssetId;
    }
```

Recommendation: Consider optimizing the function for any on-chain integrators to benefit from the gas savings.

```
Before Optimization

[PASS] test_CanExtractGenericSwapMinCallData() (gas: 35936)
[PASS] test_CanExtractGenericSwapV3MultipleParameters() (gas: 47362)
[PASS] test_CanExtractGenericSwapV3SingleParameters() (gas: 46461)

After Optimization

[PASS] test_CanExtractGenericSwapMinCallData() (gas: 35727)
[PASS] test_CanExtractGenericSwapV3MultipleParameters() (gas: 46982)
[PASS] test_CanExtractGenericSwapV3SingleParameters() (gas: 45757)
```

LI.FI: Implemented as suggested: `65aeb11253170fc480e72b6c9b3ad7abee9d6f0c`

Researcher: Validated fix.

6.2.3 `extractNonEVMAddress` function could be optimized by reducing MLOADS

Context: CalldataVerificationFacet.sol#L108

Description: The `extractNonEVMAddress` function could be optimized to save approximately 300 GAS for a non-swap case and 450 GAS for a swap case using the following optimizations,

1. Avoid excessive memory writing of bridge data. The function could avoid writing additional bridge data by adding a new function that returns whether the bridge data has source swaps. By doing so, around 25 GAS can be saved.
2. Declare empty `callData` The function writes the data to a memory variable called `callData`, which will be overwritten in a swap case. Hence, it can be restructured as well.

```
before optimization
[PASS] test_CanExtractNonEVMAddress() (gas: 42868)
[PASS] test_CanExtractNonEVMAddressWithSwaps() (gas: 67292)

after optimization
[PASS] test_CanExtractNonEVMAddress() (gas: 42535)
[PASS] test_CanExtractNonEVMAddressWithSwaps() (gas: 66875)
```

Recommendation: Consider implementing the above fixes to optimize the function. The updated function is also provided below for your reference,

```
function extractNonEVMAddress(
    bytes calldata data
) external pure returns (bytes32 nonEVMAddress) {
    bytes memory callData;

    if (
        bytes4(data[:4]) == StandardizedCallFacet.standardizedCall.selector
    ) {
        // standardizedCall
        callData = abi.decode(data[4:], (bytes));
    } else {
        callData = data;
    }

    // Non-EVM address is always the first parameter of bridge specific data
    if (_extractBridgeData(data).hasSourceSwaps) {
        assembly {
            let offset := mload(add(callData, 0x64)) // Get the offset of the bridge specific data
            nonEVMAddress := mload(add(callData, add(offset, 0x24))) // Get the non-EVM address
        }
    } else {
        assembly {
            let offset := mload(add(callData, 0x44)) // Get the offset of the bridge specific data
            nonEVMAddress := mload(add(callData, add(offset, 0x24))) // Get the non-EVM address
        }
    }
}
```

LI.FI: Implemented as suggested: `2475ff1af105667cc2b866240bdbe735bd038cd1`

Researcher: Validated fix. Looks all good.

6.3 Informational

6.3.1 Add support for `AcrossFacetV3`

Context: CalldataVerificationFacet.sol#L18

Description: The CalldataVerificationFacet decoded/uses outdated bridge facets, including earlier versions of Across. It would be good to include the latest version (v3) of the facet so that redeployment can be avoided.

Recommendation: Add support for the latest version (v3) of Across Facet.

LI.FI: Fixed in `6a7b6b1acb8cb9463d9d2790a9469ffea3d54c1b`

Researcher: Validated fix. Across V3 facet decoding added is all good.

6.3.2 Add support for `StargateV2` facet

Context: CalldataVerificationFacet.sol#L18

Description: The `CalldataVerificationFacet` decoded/uses outdated bridge facets, including earlier versions of Stargate. It would be good to update them to the latest version so that redeployment can be avoided.

Recommendation: Add support for the latest version of Stargate Facet.

LI.FI: The StargateV1 facet is still in use; hence, we keep handling both for now. StargateV2 handling was added: `afa7daba446baae397ead5d6c14f77b7d3eb3865`and further optimizations to the fixes were made in`028bcc2b7f0d92ff0d195ba78dffe7f0c7c7a105`

Researcher: Validated fix. Stargate V2 facet added is all good.

6.3.3 Remove redundant return in `extractGenericSwapParameters` function

Context: CalldataVerificationFacet.sol#L205-L210

Description: The function `extractGenericSwapParameters` assigns all the return variables inside the function execution and also has an unnecessary explicit return statement.

Recommendation: Consider removing the explicit return statement.

LI.FI: Fixed in `65aeb11253170fc480e72b6c9b3ad7abee9d6f0c`

Researcher: Validated fix. Looks all good.

6.3.4 typos

Context: CalldataVerificationFacet.sol#L105-L107, CalldataVerificationFacet.sol#L226

Description:

1. The CalldataVerification Facet has inconsistency in the inline documentation of the function, `extractNonEVMAddress`, where the documentation uses two slashes (`//`) instead of (`///`), which is inconsistent with the natspec format.
2. In the `validateCalldata` and `validateDestinationCalldata` functions, there's a typo in one of the comments. The word "validate" should be "valid" in two instances:

```
- /// @return isValid Whether the calldata is validate
+ /// @return isValid Whether the calldata is valid

- /// @return isValid Whether the destination calldata is validate
+ /// @return isValid Whether the calldata is valid
```

Recommendation: Consider fixing the typos above to improve code quality and consistency.

LI.FI: Fixed in `ecb03c81074c59ed3bfa7d401627a1ae3fefad88`

Researcher: Fix validated. Looks good.