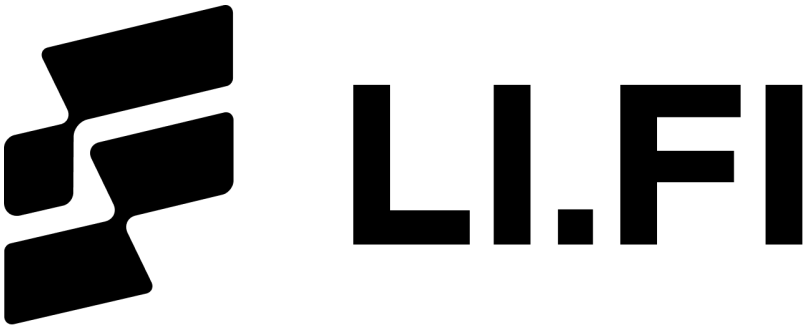

LI.FI Security Review

LidoWrapper.sol(v1.0.0), GenericSwapFacetV3.sol(v1.0.2)

Security Researcher

Sujith Somraaj (somraajsujith@gmail.com)

Report prepared by: Sujith Somraaj

June 10, 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, 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, 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/LidoWrapper.sol(v1.0.0)
- src/Facets/GenericSwapFacetV3.sol(v1.0.2)

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 5.5 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 5 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	57f8eb80ff, feab070994
Audit Timeline	June 09, 2025
Methods	Manual Review
Documentation	High
Test Coverage	High

Issues Found	
Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	0
Gas Optimizations	0
Informational	5
Total Issues	**5**

6 Findings

6.1 Informational

6.1.1 Typo in `GenericSwapFacetV3`

Context: GenericSwapFacetV3.sol#L555

Description: There's a small typo in the line below:

```
// sometimes produce rounding errors. In those cases there might we 1 wei leftover at the end of a swap
```

Recommendation: Consider fixing the typo:

```
- // sometimes produce rounding errors. In those cases there might we 1 wei leftover at the end of a
↪  swap
+ // sometimes produce rounding errors. In those cases there might be 1 wei leftover at the end of a
↪  swap
```

LI.FI: Fixed in 1ef2674ef51162e42ac5d4fb487f3e95d996cef5

Researcher: Verified fix

6.1.2 Missing events for wrapping and unwrapping operations

Context: LidoWrapper.sol#L59, LidoWrapper.sol#L78

Description: The `LidoWrapper.sol` contract fails to emit events for important state-changing operations, specifically when users wrap stETH to wstETH or unwrap wstETH to stETH. This absence of events makes it difficult to track these operations off-chain, limiting the contract's transparency and observability.

Recommendation: Consider adding appropriate events for both key functions mentioned above.

LI.FI: Acknowledged. The stETH contract will always emit events. We dont need to add another layer of events for this as it would just be duplicate information.

Researcher: Acknowledged.

6.1.3 Add slippage params to `wrapStETHToWstETH` and `unwrapWstETHToStETH`

Context: LidoWrapper.sol#L59, LidoWrapper.sol#L78

Description: The LidoWrapper contract lacks slippage protection in the `wrapStETHToWstETH()` and `unwrapWstETHToStETH()` functions.

This makes transactions vulnerable to MEV (Miner Extractable Value) attacks and could result in users receiving fewer tokens than expected when the exchange rate between stETH and wstETH fluctuates.

Recommendation: Add minimum amount parameters and verification checks to both functions:

```
function wrapStETHToWstETH(uint256 _amount, uint256 _minAmountOut) external {
    // Pull stETH from sender
    IERC20(address(ST_ETH)).transferFrom(
        msg.sender,
        address(this),
        _amount
    );

    // Call `unwrap` on stETH contract to get wstETH (naming is inverted)
    ST_ETH.unwrap(_amount);

    // Get resulting wstETH amount
    uint256 balance = IERC20(WST_ETH_ADDRESS).balanceOf(address(this));

    // Verify slippage tolerance is met
    require(balance >= _minAmountOut, "Insufficient output amount");

    // Transfer resulting wstETH to sender
    IERC20(WST_ETH_ADDRESS).transfer(msg.sender, balance);
}
```

LI.FI: Acknowledged but not fixed. Since this is not an AMM pool, we do not see how this could be exploited. If you believe there is a valid exploit factor, then we will be happy to add protection. If not, then we would prefer to save the gas.

Researcher: Don't see any direct attack vectors, but it's good to have such defensive checks to protect any similar kind of attacks.

6.1.4 Remove hardcoded values

Context: LidoWrapper.sol#L50

Description: The code contains hardcoded values, affecting readability.

```
// the wrap/unwrap functions are different on mainnet
if (block.chainid == 1) revert ContractNotYetReadyForMainnet();
```

Recommendation: Consider replacing the hardcoded values with constants as follows:

```
uint256 private constant ETH_CHAIN_ID = 1;

if (block.chainid == ETH_CHAIN_ID) revert ContractNotYetReadyForMainnet();
```

LI.FI: Fixed in 5f7f2b8734d6e08a0bc8c97816ada17fdc3fda95

Researcher: Verified fix

6.1.5 Remove unused code from `IStETH`

Context: LidoWrapper.sol#L18-L20

Description: The interface `IStETH` has function `unwrapShares()` which is not used anywhere in the code.

Recommendation: Consider removing the unused function in the interface declaration.

LI.FI: Fixed in d162f76ae77a8eae1c9f870e4441d1cbc5ef3085

Researcher: Verified fix