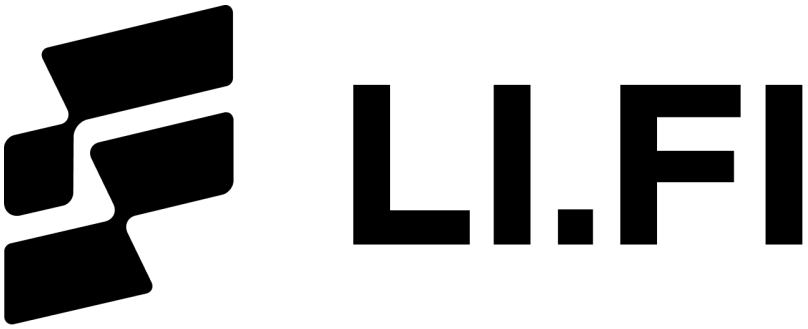

LI.FI Security Review

LibAsset.sol(v2.1.0)

Security Researcher

Sujith Somraaj (somraajsujith@gmail.com)

Report prepared by: Sujith Somraaj

June 20, 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/Libraries/LibAsset.sol(v2.1.0)

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 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 1 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	8a3e150513c
Audit Timeline	June 20, 2025
Methods	Manual Review
Documentation	High
Test Coverage	Medium

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

6 Findings

6.1 Informational

6.1.1 Function `isContract()` will fail to identify contracts sizes less than 23

Context: LibAsset.sol#L203

Description: The function `isContract()` in `LibAsset` is used to identify if a specified address is an EOA (or) smart contract based on the code size. To support EIP-7702, contracts of sizes below 23 are considered EOAs.

While this works good under most scenarios, there is a possibility to deploy contract with sizes < 23 by bypassing the solidity compiler. All contracts deployed through the compiler will be at-least have a size of 62, the ones created bypassing can have sizes < 23, which can be identified as EOA, wrongfully by this function.

PoC:

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.13;

import "forge-std/Test.sol";
import "forge-std/console.sol";

contract SetEOATest is Test {
    // EOAImplementation public implementation;
    // DestructedContract public sdImpl;

    uint256 constant ALICE_PK = 1020321312302131;
    address payable ALICE_ADDRESS = payable(vm.addr(ALICE_PK));
    bytes3 internal constant DELEGATION_DESIGNATOR = 0xef0100;

    function setUp() external {
        vm.createSelectFork("https://mainnet.era.zksync.io");
    }

    function test_isContract() public {
        Vm.SignedDelegation memory signedDelegation = vm.signDelegation(address(520), ALICE_PK);

        vm.startBroadcast(ALICE_PK);
        vm.attachDelegation(signedDelegation);

        console.log(isContract(ALICE_ADDRESS));
        console.log(isContract(address(deployMinimal())));
    }

    function isContract(address account) internal view returns (bool) {
        uint256 size;
        assembly {
            size := extcodesize(account)
        }

        console.log("size:", size);
        // Return true only for regular contracts (size > 23)
        // EIP-7702 delegated accounts (size == 23) are still EOAs, not contracts
        return size > 23;
    }

    function deployMinimal() internal returns (address) {
        bytes memory runtimeCode = hex"00"; // STOP opcode (1 byte)

        // Creation code that copies runtime code and returns it
        bytes memory creationCode = abi.encodePacked(
            hex"60", uint8(runtimeCode.length), // PUSH1 <runtime_size>
```

```
            hex"60", uint8(10 + runtimeCode.length), // PUSH1 <code_offset>
            hex"6000", // PUSH1 0x00 (dest_offset)
            hex"39", // CODECOPY
            hex"60", uint8(runtimeCode.length), // PUSH1 <runtime_size>
            hex"6000", // PUSH1 0x00 (offset)
            hex"f3", // RETURN
            runtimeCode // The actual runtime code
        );

        address deployed;
        assembly {
            deployed := create(0, add(creationCode, 0x20), mload(creationCode))
        }

        return deployed;
    }
}
```

Recommendation: Though the contracts of smaller size pose no security threats to the protocol at this point, would suggest consider alternative ways to identify EOAs.

LI.FI: Acknowledged.

Researcher: Acknowledged.