LI.FI Security Review

EmergencyPauseFacet(v1.0.1)

Independent Review By:

Sujith Somraaj (somraajsujith@gmail.com)

November 5, 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/EmergencyPauseFacet.sol

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 0.5 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 2 issues were found.

Project Summary	
Project Name	LI.FI
Repository	lifinance/contracts
Commit Hashes	da61880ba.....45ec18ac
Type of Project	
Audit Timeline	November 4, 2024
Methods	Manual Review

Issues Found	
Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	1
Gas Optimizations	0
Informational	1
Total Issues	**2**

6 Findings

6.1 Low Risk

6.1.1 Unhandled revert while blacklisting non-existent facets during `unpause`

Context: EmergencyPauseFacet.sol#L151

Description: The `unpause` function is called by the `diamond owner` to unpause the diamond. This function accepts an array of addresses that must be blacklisted after the pause.

However, there are no checks made on this address to make sure that a valid facet address is passed in the blacklisted array leading to unexpected behavior.

The function reverts for a non-existent facet, but it reverts with a panic instead of a meaningful error message.

PoC: I tried calling `unpauseDiamond` with a non-existent facet address. The function reverts on L156 where it tries to access the first element of an empty array.

```
    [99488] EmergencyPauseFacet::unpauseDiamond([0xB9A555095D3d45211072aEf86D1622D1f6FDf316])
       [97051] EmergencyPauseFacet::unpauseDiamond([0xB9A555095D3d45211072aEf86D1622D1f6FDf316])
    ↪  [delegatecall]
          [0] console::log("blacklist length", 1) [staticcall]
             ← [Stop]
          [0] console::log(0) [staticcall]
             ← [Stop]
          ← [Revert] panic: array out-of-bounds access (0x32)
       ← [Revert] panic: array out-of-bounds access (0x32)
    ← [Revert] panic: array out-of-bounds access (0x32)
```

Recommendation: Consider either continuing the execution flow for non-existent facets in the blacklist array (or) adding an explicit revert.

```
function unpauseDiamond(address[] calldata _blacklist) external {
  ...
  bytes4[] memory currentSelectors;
  for (uint256 i; i < _blacklist.length; ) {
     currentSelectors = LibDiamondLoupe.facetFunctionSelectors(
      _blacklist[I]
      );
+    if(currentSelectors.length == 0) continue;
  }
  ...
}
```

LI.FI: This is an admin-only function; we know what call data we sent, so if the call fails, it's not a problem.

Researcher: Acknowledged. Add more checks to the emergency services off-chain to ensure the call is crafted properly.

6.2 Informational

6.2.1 Remove unused import `IDiamondCut`

Context: EmergencyPauseFacet.sol#L8

Description: The `EmergencyPauseFacet` contract imports `IDiamondCut`, but is not used anywhere in the code.

Recommendation: Consider removing the unused import.

LI.FI: Fixed in ecad14d13e8edb6928e52bf3435e059b20affe50

Researcher: Verified fixes.

7 Appendix

The `EmergencyPauseFacet` can be removed by the diamond owner while unpausing the contract. This allows the owner to remove the pause functionality from the contract. LI.FI team was made aware of this issue during this audit and acknowledged it. Furthermore, the team will handle this function with extra caution in the future.