LI.FI Security Review

EmergencyPauseFacet(v1.0.0)

Independent Review By:

Sujith Somraaj (somraajsujith@gmail.com)

September 13, 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(v1.0.0)

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 14 issues were found.

Project Summary	
Project Name	LI.FI
Repository	lifinance/contracts
Commit	77441a088e0.....f6c5c0988ee42
Type of Project	
Audit Timeline	September 01, 2024
Methods	Manual Review

Issues Found	
Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	3
Gas Optimizations	2
Informational	9
Total Issues	**14**

6 Findings

6.1 Low Risk

6.1.1 Potential accidental removal of `DiamondCutFacet` during unpause

Context: EmergencyPauseFacet.sol#L140

Description: The `unpauseDiamond` function allows for a blacklist of facets that should not be reactivated. However, there's no safeguard to prevent the `DiamondCutFacet` from being included in this blacklist, which could lead to the accidental permanent disabling of upgrade functionality.

Recommendation: Implement a check to ensure that the DiamondCutFacet is never included in the blacklist.

```
function unpauseDiamond(address[] calldata _blacklist) external {
+  bytes4[] memory selectors;
    for (uint256 i; i < _blacklist.length; ) {
+  selectors = LibDiamondLoupe.facetFunctionSelectors(_blacklist[i]);
+  if(selectors[0] == DiamondCutFacet.diamondCut.selector) continue;
      // re-add facet and its selectors to diamond
      LibDiamond.removeFunctions(
         address(0),
+        selectors
       );
}
```

LI.FI: Fixed in 7709442ae76b0209a93c732c412fcb444216c618

Researcher: Confirmed.

6.1.2 `pauserWallet` can lead to irrecoverable states

Context: EmergencyPauseFacet.sol#L83, EmergencyPauseFacet.sol#L57

Description: The `pauserWallet` can take emergency actions on the LI.FI diamond contract by removing a specific facet (or) all the aspects (pausing) and later adding them.

However, the `pauserWallet` can push LI.FI diamond contract by doing the following:

- pause the diamond contract by calling the `pauseDiamond()` function
- remove the `EmergencyPauseFacet` by calling the `removeFacet()` function

By doing so, the pauser wallet can force LI.FI to deploy a new diamond contract as the owner can no longer add/remove facets through the `EmergencyPauseFacet` or the `DiamondCut` facet.

Recommendation:

- As the contract owner has a superior role to the pauser wallet, prevent the `pauserWallet` from operating on the `EmergencyPauseFacet`. Similar to the `unpauseDiamond()` function, the `removeFacet()` function can be operated only by the contract owner if the Diamond contract is paused.
- (or) Introduce a new function controlled by diamond admin to remove the emergency pause facet and add a check to prevent it from being removed in the `removeFacet()` function.

LI.FI: Fixed in d70d09b47dca3f36068311659510cc1764019f7a

Researcher: Confirmed.

6.1.3 Fix the `TODO` in `pauseDiamond` function

Context: EmergencyPauseFacet.sol#L84

Description: The `pauseDiamond` function is used by the pauser wallet/diamond admin to pause the diamond contract by removing all the function selectors. To do so, the function queries all the facets of the diamond using the `_getAllFacetFunctionSelectorsToBeRemoved` internal functions.

However, this function risks running out of gas if the size of the facets grows over time, and the same is added as a `TODO` in the function, which is not completed.

```
//TODO: add handling for cases with too many facets, and TX will run out of gas (>> pagination).
```

Recommendation: Consider adding a new function that allows the admin to pass an array of facets to pause, which can be used to pause the diamond without running out of block gas.

LI.FI: Acknowledged.

Researcher: Acknowledged. It is unlikely there will be as many facets to run into the out-of-gas issue; if it happens, a new emergency pause facet has to be added. Adding a watcher to the deploy pipeline to make sure whenever you add a new facet, you can pause the entire diamond.

6.2 Gas Optimization

6.2.1 Optimize `_isEmergencyPauseFacet` function

Context: EmergencyPauseFacet.sol#L159

Description: The `_isEmergencyPauseFacet` internal function is used to compare if the facet address is equal to the emergency pause facet. The function is used only in a single instance, so it can be optimized to avoid extra memory.

```
function _getAllFacetFunctionSelectorsToBeRemoved()
  internal
  view
  returns (IDiamondLoupe.Facet[] memory toBeRemoved)
{
...
if (allFacets[i].facetAddress != _emergencyPauseFacetAddress) {
...
}
```

```
[PASS] test_DiamondOwnerCanPauseDiamond() (gas: 490820) --> BEFORE OPTIMIZATION
[PASS] test_DiamondOwnerCanPauseDiamond() (gas: 490536) --> AFTER OPTIMIZATION
```

Recommendation: Optimize code by removing the `_isEmergencyPauseFacet`.

LI.FI: Fixed in 7ea4c75b2049fbe173b145eed91a7b0b048eb786

Researcher: Confirmed.

6.2.2 `OnlyPauserWalletOrOwner` modifier could be optimized

Context: EmergencyPauseFacet.sol#L37

Description: The `OnlyPauserWalletOrOwner` modifier validates the `msg.sender`. This modifier accepts the `msg.sender` as an argument; however, `msg.sender` is directly available to the modifier by default.

Hence, optimizing the function could result in a more clear and slightly optimized code.

Recommendation: Consider updating the `OnlyPauserWalletOrOwner` as below:

```
...
- modifier OnlyPauserWalletOrOwner(address msgSender) {
-     if (
-        msgSender != pauserWallet &&
-        msgSender != LibDiamond.contractOwner()
-    ) revert UnAuthorized();
-    _;
- }
+ modifier OnlyPauserWalletOrOwner {
+    if (
+        msg.sender != pauserWallet &&
+        msg.sender != LibDiamond.contractOwner()
+    ) revert UnAuthorized();
+    _;
+ }
...
function removeFacet(
        address _facetAddress
-    ) external OnlyPauserWalletOrOwner(msg.sender) {
+    ) external OnlyPauserWalletOrOwner {
...
```

LI.FI: Fixed in e6cbae6ef39e1fb13937aaaea384d0f93397c591

Researcher: Confirmed.

6.3 Informational

6.3.1 Increase test coverage

Context: EmergencyPauseFacet.sol#L17

Description: The test coverage of smart contracts under review must be completed. Adequate test coverage and regular reporting are essential to ensure the codebase works as intended.

Key test cases that were missing are,

- Passing in a non-existent facet in `removeFacet()` function
- Passing in the Diamond Cut Facet in `removeFacet()` function

Recommendation: Consider covering all the lines of code using integration and unit tests.

LI.FI: Fixed in 124fd68c7cac77dece45c7d7fc65e30060663341

Researcher: Confirmed.

6.3.2 Incorrect inline documentation in `unpauseDiamond` function

Context: EmergencyPauseFacet.sol#L139

Description: In the `unpauseDiamond` function of the `EmergencyPauseFacet` contract, an incorrect comment misrepresents the functionality of the code it describes. The comment suggests that the code is re-adding a facet and its selectors to the diamond, while in reality, the code is removing functions of blacklisted facets.

```
function unpauseDiamond(address[] calldata _blacklist) external {
    // ... (earlier part of the function)

    for (uint256 i; i < _blacklist.length; ) {
        // re-add facet and its selectors to diamond  // <-- This comment is incorrect
        LibDiamond.removeFunctions(
            address(0),
            LibDiamondLoupe.facetFunctionSelectors(_blacklist[i])
        );
        // ...
    }
    // ...
}
```

Recommendation: Remove the incorrect comment and replace it with an accurate description of the code's functionality

LI.FI: Fixed in 72e39cd80f406875c25855788899887bbb05893a

Researcher: Confirmed.

6.3.3 Lack of event for blacklisted facets

Context: EmergencyPauseFacet.sol#L138

Description: The `unpauseDiamond` function doesn't emit events for blacklisted facets, which could make it challenging to track which facets were not reinstated.

Recommendation: Add an event emission for each blacklisted facet that is not reinstated.

LI.FI: Fixed in 7fcf18641a45b9ecbdf744a014652e20b7092455 by using the diamondCut function that emits an event for every removed facet.

Researcher: Confirmed.

6.3.4 Unnecessary delete operation in `_getAllFacetFunctionSelectorsToBeRemoved` function

Context: EmergencyPauseFacet.sol#L193

Description: In the _getAllFacetFunctionSelectorsToBeRemoved function, there's an unnecessary `delete toBeRemoved;` operation.

Recommendation: Remove the line `delete toBeRemoved;` as the new array assignment immediately overwrites it.

LI.FI: Fixed in 54d4142c988e28b90c809e1b2e51560d5b39e040

Researcher: Confirmed.

6.3.5 Revert if `facets.length` is zero in `pauseDiamond()` function

Context: EmergencyPauseFacet.sol#L88

Description: The `pauseDiamond()` function does not revert if the `facets` length is zero. This can lead to unexpected behavior if external agents/monitoring services are built based on the `EmergencyPaused` event.

Recommendation: Consider adding a revert if the `facets` length is zero in the `pauseDiamond()` function.

```
function pauseDiamond() external OnlyPauserWalletOrOwner {
    ....
    IDiamondLoupe.Facet[]
            memory facets = _getAllFacetFunctionSelectorsToBeRemoved();
    ....
+  if(facets.length == 0) revert NoFacetToPause();
}
```

LI.FI: Fixed in af6b003457da7a56e744dbf24d5cfa6168887508

Researcher: Confirmed.

6.3.6 Add batch remove function

Context: EmergencyPauseFacet.sol

Description: The `removeFacet` function allows the pauser (or) owner to pause a specific facet address passed in. However, in some emergency scenarios, there might be a reason to remove multiple facets simultaneously, which can now be batched through a multi-sig. However, an additional batch remove function could be helpful if the pauser wallet is an EOA.

Recommendation: Consider adding a new `batchRemoveFacet` function that simultaneously accepts and pauses multiple facets.

LI.FI: Thanks for pointing it out but I think we dont need this. If one facet is affected, we will remove it. If several facets are affected, we will pause the diamond and unpause it with the list of facets that should be removed.

Researcher: Acknowledged.

6.3.7 Remove unused `_containsAddress` internal function

Context: EmergencyPauseFacet.sol#L167-L183

Description: The `_containsAddress()` internal function is used to parse an address array and find if a given address is present. However, this function is not used anywhere in the code, leading to inflated code size.

Recommendation: Consider removing the unused `_containsAddress` function.

LI.FI: Fixed in 8effe9a6af409657fa4cfc395fdb1dfb4e36a989

Researcher: Confirmed.

6.3.8 Index event parameters

Context: EmergencyPauseFacet.sol#L19-L21

Description: The events `EmergencyFacetRemoved`, `EmergencyPaused`, and `EmergencyUnpaused` lacks the `indexed` keyword for their parameters. Indexing event parameters improves off-chain tracking.

Recommendation: Add `indexed` keyword to event params

LI.FI: Fixed in 111a2e4f6e7986a2d4c56f95524eec76796e8beb

Researcher: Confirmed

6.3.9 Remove unused imports in `EmergencyPauseFacet`

Context: EmergencyPauseFacet.sol#L7, EmergencyPauseFacet.sol#L10

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

Recommendation: Consider removing the unused import.

LI.FI: Fixed in d5c7244dc65708ee5dbc3dcffb9c39b920ba0f33

Researcher: Confirmed.