LI.FI Security Review

LiFiTimelockController(v.1.0.0)

Independent Review By:

Sujith Somraaj (somraajsujith@gmail.com)

January 10, 2025

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/Security/LiFiTimelockController.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 2 hours a total of 2 issues were found.

Project Summary	
Project Name	LI.FI
Repository	lifinance/contracts
Commit Hashes	af28afab139a158650f4ef4f4442f2b5fcdfe497
Type of Project	
Audit Timeline	Jan 10, 2025
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 Use `onlyRole` instead of `onlyRoleOrOpenRole`

Context: LiFiTimelockController.sol#L46, LiFiTimelockController.sol#L59

Description: The **onlyRoleOrOpenRole** modifier contains logic that considers a role "open" if it's granted to the zero address (address(0)). While this might be intended functionality, it creates a potential security risk if roles are misconfigured or accidentally reset.

This is particularly concerning for the `unpauseDiamond()` function, which could become accessible to anyone if the **TIMELOCK_ADMIN_ROLE** is accidentally granted to `address(0)`.

Recommendation: Consider removing the "open role" concept unless absolutely necessary, and use the onlyRole modifier from AccessControl.

LI.FI: Fixed in b26f9526c408fb0e2731e095b1188677706e97cb

Researcher: Verified fix.

6.2 Informational

6.2.1 Timelock bypass in emergency unpause function

Context: LiFiTimelockController.sol#L57

Description: The `unpauseDiamond()` function in `LiFiTimelockController` is designed to reactivate the diamond contract after a pause, intentionally bypassing the timelock delay mechanism.

The `unpauseDiamond()` function allows immediate execution of unpausing operations without respecting the timelock delay. While the comments suggest this is safe because the function can only remove existing facets, this still represents a significant deviation from the timelock's security model.

Recommendation: Consider enforcing a short timelock for the `unpause()` function. Also, ensure that the TIMELOCK_ADMIN_ROLE multi-sig is trusted and safe.

LI.FI: Acknowledged.

Researcher: Acknowledged.