LFi

SMART CONTRACT AUDIT

zokyo

May 22th 2023 | v. 1.0

Security Audit Score

PASS

Zokyo Security has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.

TECHNICAL SUMMARY

This document outlines the overall security of the LFi smart contracts evaluated by the Zokyo Security team.

The scope of this audit was to analyze and document the LFi smart contracts codebase for quality, security, and correctness.

Contract Status

There were 0 critical issues found during the audit. (See Complete Analysis)

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contracts but rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that can withstand the Ethereum network's fast-paced and rapidly changing environment, we recommend that the LFi team put in place a bug bounty program to encourage further active analysis of the smart contracts.

Table of Contents

AUDITING STRATEGY AND TECHNIQUES APPLIED

The source code of the smart contract was taken from the LFi repository and Etherscan link:

Repository:

ETHEREUM MAINNET:

cLFi : https://etherscan.io/address/0xf33ae914c49d5804a3137d0ad05b80d400c265c9#code

Within the scope of this audit, the team of auditors reviewed the following contract(s):

- cLFI.sol
- cLFI Proxy

During the audit, Zokyo Security ensured that the contract:

- Implements and adheres to the existing standards appropriately and effectively;
- The documentation and code comments match the logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices, efficiently using resources without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the most resent vulnerabilities;
- Meets best practices in code readability, etc.

Zokyo Security has followed best practices and industry-standard techniques to verify the implementation of LFi smart contracts. To do so, the code was reviewed line by line by our smart contract developers, who documented even minor issues as they were discovered. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

01	Due diligence in assessing the overall code quality of the codebase.
02	Cross-comparison with other, similar smart contracts by industry leaders.
03	Thorough manual review of the codebase line by line.

Executive Summary

There was no critical issue found during the audit alongside some of low severity and one informational issue. All the mentioned findings may have an effect only in case of specific conditions performed by the contract owner and the investors interacting with it. They are described in detail in the “Complete Analysis” section. Contracts are already deployed.

STRUCTURE AND ORGANIZATION OF THE DOCUMENT

For the ease of navigation, the following sections are arranged from the most to the least critical ones. Issues are tagged as “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. The issues that are tagged as “Verified” contain unclear or suspicious functionality that either needs explanation from the Client or remains disregarded by the Client. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

FINDINGS SUMMARY

#	Title	Risk	Status
1	Renounce Ownership	Low	Acknowledged
2	Transfer Ownership	Low	Acknowledged
3	Centralization risk and token upgradeability	Low	Acknowledged
4	Use safe upgradeable mechanism	Informational	Acknowledged

Renounce Ownership

The renounceOwnership function can be called accidentally by the admin leading to the immediate renouncement of ownership to zero address after which any onlyOwner functions will not be callable which can be risky.

Recommendation:
It is advised that the Owner cannot call renounceOwnership without first transferring ownership to a different address. Additionally, if a multi-signature wallet is utilized, executing the renounceOwnership method for two or more users should be confirmed. Alternatively, the Renounce Ownership functionality can be disabled by overriding it.

Refer to this post for additional info- https://www.linkedin.com/posts/razzor_github-razzorsecrazzorsec-contracts-activity-6873251560864968705-HOS8

Transfer Ownership

The transferOwnership() function in the contract allows the current admin to transfer his privileges to another address. However, inside transferOwnership() , the newOwner is directly stored
into the storage owner, after validating the newOwner is a non-zero address, and immediately overwrites the current owner. This can lead to cases where the admin has transferred ownership to an incorrect address and wants to revoke the transfer of ownership or in the cases where the current admin comes to know that the new admin has lost access to his account.

Recommendation:

It is advised to make ownership transfer a two-step process.

Refer- https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable2Step.sol
and https://github.com/razzorsec/RazzorSec-Contracts/blob/main/AccessControl/SafeOwn.sol

LOW | ACKNOWLEDGED

Centralization risk and token upgradeability

An upgradeable token is not recommended as it is prone to centralization risk and contradicts the notion of the immutability of smart contracts on the blockchain. Upgrade to a malicious implementation can be detrimental as it can result in minting of numerous tokens by a malicious admin. This can result in a loss of value of funds of token holders or potential rug pulls.

Recommendation:

It is advised to use an upgradeable pattern for your smart contracts only if it is really necessary and cannot be done without. Additionally, it is advised to use a multisig for the owner of upgradeable contracts if possible in order to reduce centralization risks.

INFORMATIONAL | ACKNOWLEDGED

Use safe upgradeable mechanism

Using upgradeable proxies correctly and securely is a difficult task that requires deep knowledge of the proxy pattern (refer this). Incorrectly upgrading the contracts can lead to security issues.

Recommendation:
It is advised to use Openzeppelin plugins in Hardhat or Truffle (refer https://docs.openzeppelin.com/upgrades-plugins/1.x/) to upgrade your contracts.

	cLFI	cLFI Proxy
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

ECHIDNA RESULTS

Property testing using Echidna

cLFI

```
Echidna 2.0.3
Tests found: 11
Seed: -3933260567837386790
Unique instructions: 4076
Unique codehashes: 1
Corpus size: 26
Tests
crytic_less_than_total_ERC20Properties: PASSED!
crytic_transfer_to_other_ERC20PropertiesTransferable: PASSED!
crytic_revert_transfer_to_zero_ERC20PropertiesTransferable: PASSED!
crytic_revert_transfer_to_user_ERC20PropertiesTransferable: PASSED!
crytic_revert_transferFrom_to_zero_ERC20PropertiesTransferable: PASSED!
crytic_self_transferFrom_ERC20PropertiesTransferable: PASSED!
crytic_self_transfer_ERC20PropertiesTransferable: PASSED!
```

```
gitpod /workspace/Unity-chain-Hardhat-code (main) $ echidna-test . --contract TestcLFiTransferable --config echidna_config.yaml
Analyzing contract: /workspace/Unity-chain-Hardhat-code/contracts/TestcLFiTransferable.sol:TestcLFiTransferable
crytic_totalSupply_consistant_ERC20Properties:  passed! 🎉
crytic_approve_overwrites:  passed! 🎉
crytic_self_transferFrom_to_other_ERC20PropertiesTransferable:  passed! 🎉
crytic_zero_always_empty_ERC20Properties:  passed! 🎉
crytic_self_transfer_ERC20PropertiesTransferable:  passed! 🎉
crytic_self_transferFrom_ERC20PropertiesTransferable:  passed! 🎉
crytic_revert_transferFrom_to_zero_ERC20PropertiesTransferable:  passed! 🎉
crytic_revert_transfer_to_user_ERC20PropertiesTransferable:  passed! 🎉
crytic_revert_transfer_to_zero_ERC20PropertiesTransferable:  passed! 🎉
crytic_transfer_to_other_ERC20PropertiesTransferable:  passed! 🎉
crytic_less_than_total_ERC20Properties:  passed! 🎉

Unique instructions: 4076
Unique codehashes: 1
Corpus size: 26
Seed: -3933260567837386790
```

We are grateful for the opportunity to work with the LFi team.

The statements made in this document should not be interpreted as an investment or legal advice, nor should its authors be held accountable for the decisions made based on them.

Zokyo Security recommends the LFi team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo