QuillAudits

Audit Report
April, 2023

For

Lybra Protocol

Table of Content

Executive Summary

Project Name	Lybra Finance
Overview	Lybra Finance Audit Scope Consist of Two Contracts, LBR Confirmed and IDO Confirmed. LBR is the token contract, The total supply of LBR is 100000000 and IDO is the contract used for LBR IDO. The payment will be in ETH. There will be a soft cap and a hard cap. People can always withdraw their fund as long as soft cap is not met, but once it is met, no withdrawable of invested fund will be allowed..
Timeline	15 April, 2023 to 17 April, 2023
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyse Lybra Finance codebase for quality, security, and correctness. *https://github.com/LybraFinance/Audits/tree/main/Old* Contracts Under Audit Scope: LBR confirmed.sol and IDo Confirmed.sol
Fixed in	*https://github.com/LybraFinance/Audits/tree/main/New*
Commit Hash	77c258d789e0fe871aaf68ac7fd2c87825d6e017

4
Issues Found

🟥 High 🟨 Medium
🟩 Low 🟪 Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	1	1	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	2	0

Types of Severities

High

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

✓ Re-entrancy	✓ Tautology or contradiction
✓ Timestamp Dependence	✓ Return values of low-level calls
✓ Gas Limit and Loops	✓ Missing Zero Address Validation
✓ Exception Disorder	✓ Private modifier
✓ Gasless Send	✓ Revert/require functions
✓ Use of tx.origin	✓ Using block.timestamp
✓ Compiler version not fixed	✓ Multiple Sends
✓ Address hardcoded	✓ Using SHA3
✓ Divide before multiply	✓ Using suicide
✓ Integer overflow/underflow	✓ Using throw
✓ Dangerous strict equalities	✓ Using inline assembly

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Testing

A. Contract - LBR Confirmed.sol

High Severity Issues

No issues were found

Medium Severity Issues

A1. Centralization issue

In contract LBR there is burn() function which is controlled by a lybra address which might be an owner address. But it can mint() and burn also. If the lybra address private key gets leaked or the account is hacked then a malicious user can burn anyones tokens.

Recommendation

To solve the issue please make sure that the tokens can only be burnt by the user who owns them. Also make sure to use gnosis multisig.

Status

Acknowledged

Lybra Team's Comment: This is Intended behaviour

Low Severity Issues

No issues were found

Informational Issues

No issues were found

B. Contract - IDO Confirmed.sol

Note: In contract PublicSale there is no functionality which can let users withdraw their deposits if something goes wrong. It would be beneficial to add functionality where they can withdraw their eth deposits even after the softcap is reached.

High Severity Issues

No issues were found

Medium Severity Issues

No issues were found

Low Severity Issues

B1. Low test coverage

Description

The test coverage for the contracts is not done.

Remediation

Please make sure to add unit test cases.

Status

Acknowledged

B2. Use of transfer function instead of call for eth transfer

Description

In PublicSale contract transfer is used for eth transfer which is not recommended. Transfer will always send ETH with a 2300 gas. This can be problematic for interacting smart contracts if gas cost change because their interaction may abruptly break.

Remediation

Please make sure to use call instead of transfer to send eth

Status

Resolved

B3. Use latest libraries

Description

In contract PublicSale there is use of MerkleProof library which is a bit older. It can have bugs which might lead to unknown consequences.

Remediation

Always make sure to use latest libraries

Status

Resolved

Informational Issues

No issues were found

General Recommendation

- lbr contract address variable can be immutable to save gas.
- Softcap and hardcap variables can be constants to save gas.
- In the require statements wherever there is require(block.timestamp >= endTime) it should be require(block.timestamp > endTime).
- Use safeTranfer instead of just transfer in sendToken function because you might not know which token was sent and can be stuck if it is not completely erc20 compliant.

Functional Testing

Some of the tests performed are mentioned below:

Contract: LBR Confirmed.sol

- ✓ should be able to get Contract Name
- ✓ Should be Able to get Symbol
- ✓ Should be Able to get Total Supply
- ✓ Should be Able to increase Allowance
- ✓ Should be Able to decrease Allowance
- ✓ Should be able to check balance
- ✓ Should be able to transfer
- ✓ Should be able to approve
- ✓ Should be able to get allowance
- ✓ Should be able to transferFrom
- ✓ Only Lybrafund address should able to mint
- ✓ Only Lybrafund address should able burn

Contract: IDOConfirmed.sol

- ✓ test MintFunction
- ✓ Test RevertIf HardCapReached
- ✓ Test RevertIf Owner Tried To Withdraw Before IDO End
- ✓ Test RevertIf Owner Tried To Withdraw Before SoftCap Is Reached
- ✓ Test RevertIf Public Sale Hasnt Started
- ✓ Test RevertIf SoftCap Is Reached
- ✓ Test RevertIf User Tried To MInt Before IDOEnds
- ✓ Test RevertIf User Tried To Mint Before SoftCap Is Reached
- ✓ Test RevertIf Withdrawal Amount Is Greater Than Deposited Amount
- ✓ Test RevertIf set Price Is Set After IDO Start
- ✓ Test WithdrawEtherFunction
- ✓ Test join
- ✓ Test leave
- ✓ Test sendTokenFunction
- ✓ Test setTime

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of Lybra Finance. We performed our audit according to the procedure described above.

Some issues of Medium, Low and informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the Lybra Finance Platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Lybra Finance Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

700+
Audits Completed

$16B
Secured

700K
Lines of Code Audited

Follow Our Journey

Audit Report
April, 2023

For

Lybra Protocol

QuillAudits

Canada, India, Singapore, United Kingdom

www.quillaudits.com

audits@quillhash.com