DeHacker

Code Security Assessment

Life Bank Chain

April 10th, 2025

Contents

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best prac tices. Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Issue Categories

Every issue in this report was assigned a severity level from the following:

Critical severity issues

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

Major severity issues

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

Medium severity issues

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

Minor severity issues

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

Informational

A vulnerability that has informational character but is not affecting any of the code.

Overview

Project Summary

Project Name	LifeBank Chain
Platform	Binance Smart Chain (BSC)
Website	https://lifebc.io
Type	DeSci
Language	Solidity
Codebase	https://bscscan.com/address/0x574d3724caa99b6be2b4782f84a551515c1e21f2

Vulnerability Summary

Vulnerability Level	Total	Mitigated	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	1	0	0	1	0	0
Medium	0	0	0	0	0	0
Minor	1	0	0	1	0	0
Informational	1	0	0	1	0	0
Discussion	1	0	0	1	0	0

Audit scope

ID	File	SHA256 Checksum
LBC	LBC.sol	075287d92e6a2e5a1ac252d34825734a6039e4251fd55e4f29c432e5aaae7c9f

Findings

ID	Title	Severity	Status
LIC-01	Initial Token Distribution	Major	Acknowledged
LIC-02	Outdated Solidity Version	Minor	Acknowledged
LIC-03	Unreachable Code	Informational	Acknowledged
LIC-04	Variable That Could Be Declared As immutable	Discussion	Acknowledged

MAJOR

LIC-01 | Initial Token Distribution

Issue	Severity	Location	Status
Centralization	Major	LBC.sol: 356~357	Acknowledged

Description

All of the LBC tokens are sent to the contract deployer or one or several externally-owned account (EOA) addresses. This isa centralization risk because the deployer or the owner(s) of the EOAs can distribute tokens without obtaining the consensusof the community. Any compromise to these addresses may allow a hacker to steal and sell tokens on the market, resultingin severe damage to the project.

Recommendation

It is recommended that the team be transparent regarding the initial token distribution process. The token distribution planshould be published in a public location that the community can access. The team should make efforts to restrict access tothe private keys of the deployer account or EOAs. A multi-signature (⅔, ⅗) wallet can be used to prevent a single point offailure due to a private key compromise. Additionally, the team can lock up a portion of tokens, release them with a vestingschedule for long-term success, and deanonymize the project team with a third-party KYC provider to create greateraccountability.

MINOR

LIC-02 | Outdated Solidity Version

Issue	Severity	Location	Status
Language Version	Minor	LBC.sol: 1	Acknowledged

Description

The provided smart contracts are using outdated and non-recommended versions of the Solidity compiler (0.5.17).An outdated solc version is prone to several disclosed bugs. These older compiler versions can introduce securityvulnerabilities, limit access to modern language features, and cause incompatibility with current development tools and bestpractices

Recommendation

We strongly recommend rewriting the contracts using Solidity ^0.8.0 or later and implementing modern libraries such asOpenZeppelin for standardized features like ERC20 and Ownable . This will improve security, ensure compliance with thelatest ERC standards, and simplify future upgrades or improvements.

INFORMATIONAL

LIC-03 | Unreachable Code

Issue	Severity	Location	Status
Coding Issue	Informational	LBC.sol: 526~532, 545~551, 580~583	Acknowledged

Description

The functions _mint , _burn , and _burnFrom are internal. There are no public or external functions within the LBCcontract itself that call these internal functions, makeing these functions unreachable. Unless a contract inheriting from LBCadds public wrappers, minting new tokens or burning existing tokens is impossible after deploymen

```
526     function _mint(address account, uint256 amount) internal {
```

```
545     function _burn(address account, uint256 amount) internal {
```

```
580     function _burnFrom(address account, uint256 amount) internal {
```

Recommendation

RecommendationWe recommend removing those unreachable functions.

DISCUSSION

LIC-04 | VariableThat Could Be Declared As Immutable

Issue	Severity	Location	Status
Gas Optimization	Discussion	LBC.sol: 348	Acknowledged

Description

The contract currently uses Solidity 0.5.17. Variables like _name , _symbol , and _decimals are set once in theconstructor. Migrating to Solidity v0.6.5+ (e.g., 0.8.x) would allow declaring these as immutable. Reads from immutablevariables are significantly cheaper (avoiding SLOADs) as their values are part of the deployed code.

Recommendation

If upgrading Solidity version, leverage the immutable keyword for _name , _symbol , and _decimals to optimize gasusage for their respective getter functions. Please note that this finding is not applicable if the contract remains on Solidity v0.5.17

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Appendix

Finding Categories

Centralization / Privilege
Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Coding Style
Coding Style findings usually do not affect the generated bytecode but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Volatile Code
Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Logical Issue
Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block. timestamp works.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux “sha256sum" command against the target file.

About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

BLOCKCHAIINS

Ethereum
Cosmos
Eos
Substrate

TECH STACK

Python
Solidity
Rust
c++

CONTACTS

https://dehacker.io
https://twitter.com/dehackerio
https://github.com/dehacker/audits_public
https://t.me/dehackerio
https://blog.dehacker.io/

DeHacker

April 10th 2025