



SAFUAUDIT

SMART CONTRACT AUDITS AND BLOCKCHAIN SECURITY



PROJECT: LAEEB INU

DATE: July 23, 2022



www.safuaudit.com

INTRODUCTION

Client	Laeeb Inu (LINU)
Language	Solidity
Contract address	0x2C82efF537Fa8079ED9C00a20Dff9B4a064671E1
Owner	0xa3358e4edc0f5e5ef277cb0cb226fe9ea967e724
Deployer	0xa3358e4edc0f5e5ef277cb0cb226fe9ea967e724
SHA1-Hash	2e740c6d6d530aee47ec278743265852fc765b26
Decimals	18
Supply	1,000,000,000
Platform	Binance Smart Chain
Compiler	v0.8.4+commit.c7e474f2
Optimization	No with 200 runs
Website	https://laeeb-inu.com/
Telegram	https://t.me/Labeeb_globalgroup
Twitter	https://twitter.com/Laeebinutoken



TABLE OF CONTENTS

01 INTRODUCTION

Introduction

Approach

Risk classification

02 CONTRACT INSPECTION

Contract Inspection

Inheritance Tree

Owner privileges

03 MANUAL ANALYSIS

Manual analysis

04 FINDINGS

Vulnerabilities Test

Findings list

Issues description

Good Practices

05 WEBSITE

Website Audit

06 CONCLUSIONS

Disclaimer

Rating

Conclusion



APPROACH



Audit Details

Our comprehensive audit report provides a full overview of the audited system's architecture, smart contract codebase, and details on any vulnerabilities found within the system.



Audit Goals

The audit goal is to ensure that the project is built to protect investors and users, preventing potentially catastrophic vulnerabilities after launch, that lead to scams and rugpulls.



Code Quality

Our analysis includes both automatic tests and manual code analysis for the following aspects:

- Exploits
 - Back-doors
 - Vulnerability
 - Accuracy
 - Readability
-



Tools

- Remix IDE
- Mythril
- Open Zeppelin Code Analyzer
- Solidity Code Compiler
- Hardhat



RISK CLASSIFICATION

CRITICAL

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

MEDIUM

Issues on this level could potentially bring problems and should eventually be fixed.

MINOR

Issues on this level are minor details and warning that can remain unfixed but would be better fixed at some point in the future

INFORMATIONAL

Information level is to offer suggestions for improvement of efficacy or security for features with a risk free factor.



OVERVIEW

Fees

- Buy Fees: 5%
- Sell Fees: 5%

Fees privileges

- Can't set fees

Ownership

- Owned

Minting

- No mint function

Max Tx Amount

- Can't set max Tx amount

Pause function

- Can't pause trading

Blacklist

- Can't blacklist



CONTRACT INSPECTION 🔍

Imported contracts or frameworks used:

```
| **IERC165** | Interface | |||
| **ERC165** | Implementation | IERC165 |||
| **Strings** | Library | |||
| **IAccessControl** | Interface | |||
| **Context** | Implementation | |||
| **IERC20Metadata** | Interface | IERC20 |||
| **IERC20** | Interface | |||
| **Ownable** | Implementation | Context |||
| **SafeMath** | Library | |||
| **AccessControl** | Implementation | Context, IAccessControl, ERC165 |||
| **Pausable** | Implementation | Context |||
| **ERC20** | Implementation | Context, IERC20, IERC20Metadata |||
```

Contract File:

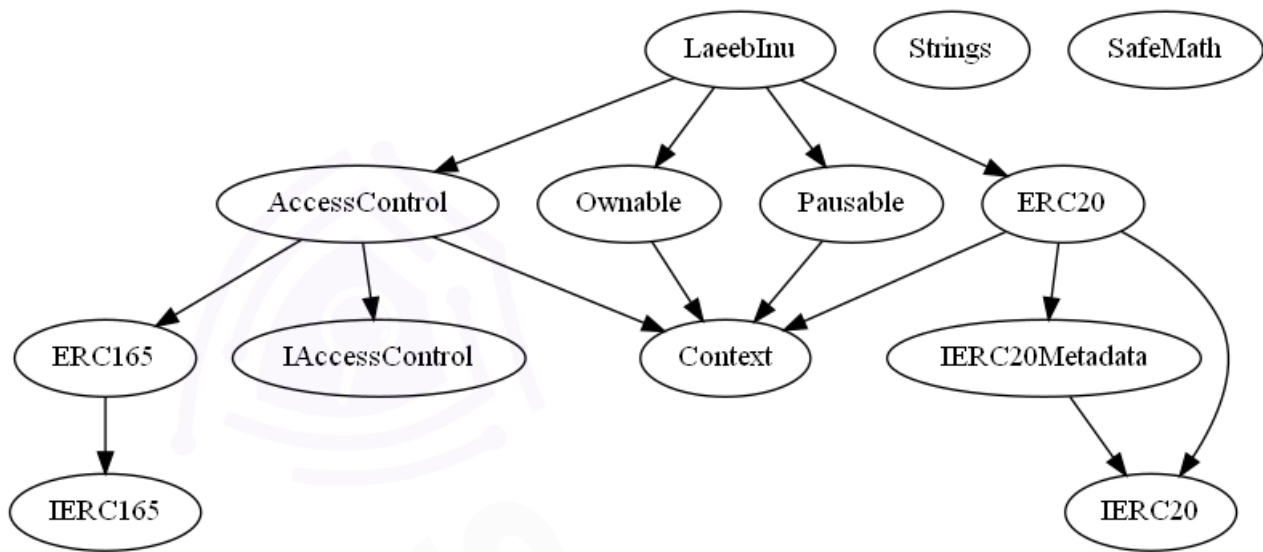
File Name	SHA-1 Hash
Laeeb.sol	2e740c6d6d530aee47ec278743265852fc765b26

```
| **LaeebInu** | Implementation | ERC20, Pausable, AccessControl, Ownable |||
| L | <Constructor> | Public ! | 🔴 | ERC20 |
| L | _beforeTokenTransfer | Internal 🔒 | 🔴 | whenNotPaused |
```

Symbol	Meaning
🛑	Function can modify state
💰	Function is payable
🔑	Private function
🔒	Internal function
NO !	Function has no modifier



INHERITANCE TREE



Inheritance is a feature of the object-oriented programming language. It is a way of extending the functionality of a program, used to separate the code, reduces the dependency, and increases the re-usability of the existing code. Solidity supports inheritance between smart contracts, where multiple contracts can be inherited into a single contract.



MANUAL FUNCTIONS ANALYSIS

The contract is verified to check if functions do and work as they should and malicious code is not inserted.

	Tested	Result
Transfer	Yes	Passed
Total Supply	Yes	Passed
Buy Back	Yes	N/A
Burn	Yes	N/A
Mint	Yes	N/A
Rebase	Yes	N/A
Pause	Yes	N/A
Blacklist	Yes	N/A
Lock	Yes	N/A
Max Transaction	Yes	N/A
Transfer Ownership	Yes	Passed
Renounce Ownership	Yes	Passed



VULNERABILITIES TEST

ID	Description	
V-01	Function Default Visibility	Passed
V-02	Integer Overflow and Underflow	Passed
V-03	Outdated Compiler Version	Passed
V-04	FloatingPragma	Minor
V-05	Unchecked Call Return Value	Passed
V-06	Unprotected Ether Withdrawal	Passed
V-07	Unprotected SELF-DESTRUCT Instruction	Passed
V-08	Re-entrancy	Passed
V-09	State Variable Default Visibility	Passed
V-10	Uninitialized Storage Pointer	Passed
V-11	Assert Violation	Passed
V-12	Use of Deprecated Solidity Functions	Passed
V-13	Delegate Call to Untrusted Callee	Passed
V-14	DoS with Failed Call	Passed
V-15	Transaction Order Dependence	Passed
V-16	Authorization through tx.origin	Passed
V-17	Block values as a proxy for time	Passed



V-18	Signature Malleability	Passed
V-19	Incorrect Constructor Name	Passed
V-20	Shadowing State Variables	Passed
V-21	Weak Sources of Randomness from Chain Attributes	Passed
V-22	Missing Protection against Signature Replay Attacks	Passed
V-23	Lack of Proper Signature Verification	Passed
V-24	Requirement Violation	Passed
V-25	Write to Arbitrary Storage Location	Passed
V-26	Incorrect Inheritance Order	Passed
V-27	Insufficient Gas Griefing	Passed
V-28	Arbitrary Jump with Function Type Variable	Passed
V-29	DoS With Block Gas Limit	Passed
V-30	Typographical Error	Passed
V-31	Right-To-Left-Override control character (U+202E)	Passed
V-32	Presence of unused variables	Passed
V-33	Unexpected Ether balance	Passed
V-34	Hash Collisions With Multiple Variable Length Arguments	Passed
V-35	Message call with the hardcoded gas amount	Passed
V-36	Code With No Effects (Irrelevant/Dead Code)	Informational
V-37	Unencrypted Private Data On-Chain	Passed



FINDINGS

ID	Category	Issue	Severity
V-01	Vulnerabilities	Unlocked Compiler	Minor
CS-01	Coding Standards	Dead Code	Informational



V-01: Unlocked Compiler

Line #7

```
pragma solidity ^0.8.4;
```

Description

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Recommendation

- Lock the pragma version and also consider known bugs (<https://github.com/ethereum/solidity/releases>) for the compiler version that is chosen.



CS-01: Dead Code

Description

- AccessControl contract, that can provide role-based access control, is never used.
- Pausable contract is never used.
- Safemath library is not used.
- Contract is ownable(owner can renounce or transfer ownership) but there are no functions with owner privileges.

Recommendation

- Remove unused contracts / functions / code for code clarity and easier review.



GOOD PRACTICES ✓

- The owner cannot mint new tokens after deployment
- The owner cannot stop or pause the contract
- The owner cannot set a transaction limit
- The owner cannot set the fees



Website	https://laeeb-inu.com/
Domain Registry	http://www.godaddy.com
Domain Expiry Date	2023-07-20
Response Code	200
SSL Checker and HTTPS Test	Passed
Deprecated HTML tags	Passed
Robots.txt	Informative
Sitemap Test	Informative
SEO Friendly URL	Passed
Responsive Test	Passed
JS Error Test	Passed
Console Errors Test	Passed
Site Loading Speed Test	3.36 seconds - Passed
HTTP2 Test	Passed
Safe Browsing Test	Passed



DISCLAIMER

SafuAudit.com is not a financial institution and the information provided on this website does not constitute investment advice, financial advice, trading advice, or any other sort of advice. You should not treat any of the website's content as such. Investing in crypto assets carries a high level of risk and does not hold guarantees for not sustaining financial loss due to their volatility.

Accuracy of Information

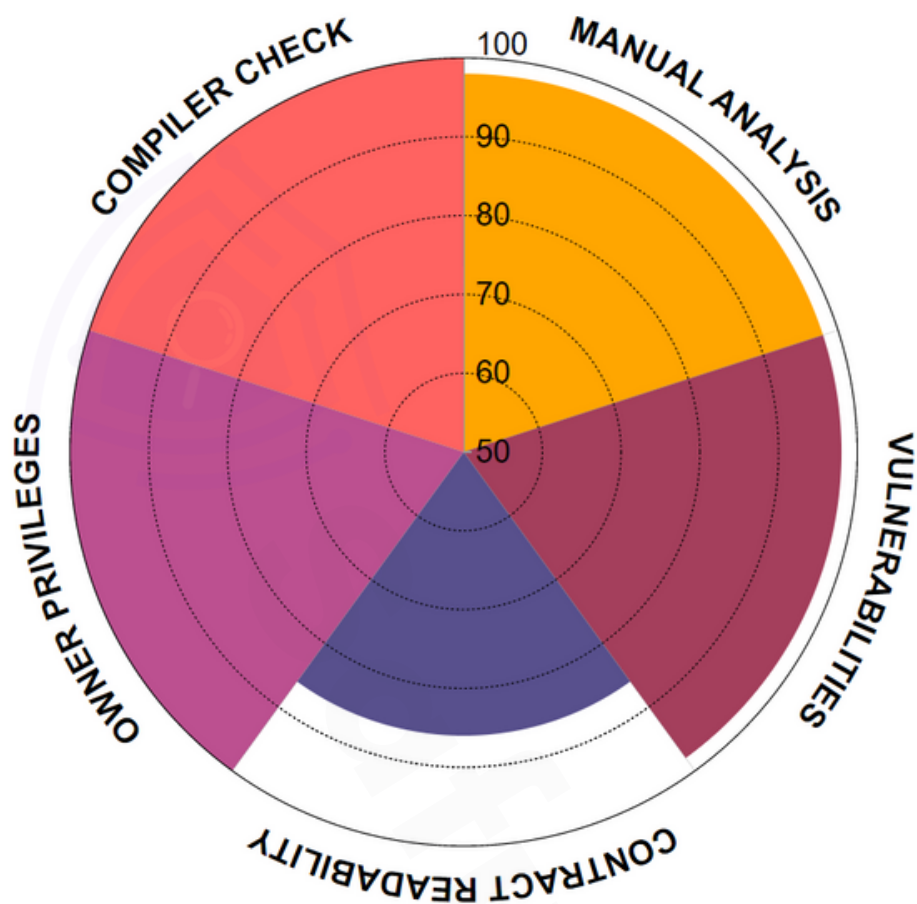
SafuAudit will strive to ensure the accuracy of the information listed on this website although it will not hold any responsibility for any missing or wrong information. SafuAudit provides all information as is. You understand that you are using any and all information available here at your own risk. Any use or reliance on our content and services is solely at your own risk and discretion.

The purpose of the audit is to analyze the on-chain smart contract source code and to provide a basic overview of the project.

While we have used all the information available to us for this straightforward investigation, you should not rely on this report only – we recommend proceeding with several independent audits. Be aware that smart contracts deployed on a blockchain aren't secured enough against external vulnerability or a hack. Be aware that active smart contract owner privileges constitute an elevated impact on the smart contract safety and security. Therefore, SafuAudit does not guarantee the explicit security of the audited smart contract. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.



RATING



Manual Analysis



Vulnerabilities



Contract Readability



Owner Privileges

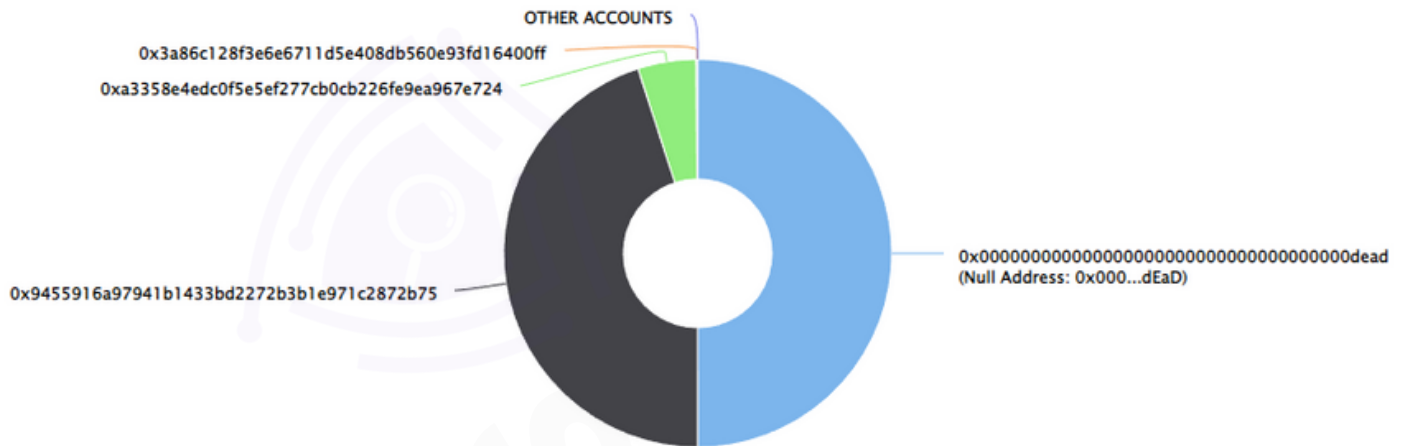


Compiler Check

Final Score: **96.4**



Top 10 holders



Rank	Address	Quantity (Token)	Percentage
1	Null Address: 0x000...dEaD	500,000,000	50.0000%
2	📄 0x9455916a97941b1433bd272b3b1e971c2872b75	450,274,000	45.0274%
3	0xa3358e4edc0f5e5ef277cb0cb226fe9ea967e724	48,601,020.07	4.8601%
4	0x3a86c128f3e6e6711d5e408db560e93fd16400ff	1,124,979.93	0.1125%

Project Laeab Inu (LINU) does not contain any severe issues or risk characteristics.

SafuAudit has tested the security based on manual and automated tests.
Please note that we don't offer any warranties for the business model.



SAFUAUDIT

SMART CONTRACT AUDITS AND BLOCKCHAIN SECURITY



"Only in growth, reform, and change, paradoxically enough, is true security to be found."



www.safuaudit.com

