



SAFUAUDIT

SMART CONTRACT AUDITS AND BLOCKCHAIN SECURITY



PROJECT: BREAD BAKERY

DATE: May 29, 2022



www.safuaudit.com

INTRODUCTION

Client	Bread Bakery (BREAD)
Language	Solidity
Contract address	0xAc8850C05b8D26BccAd9b3815C5B9D97b7d7af8D
Owner	0xD74650b3e35673dE488DC94e6E305df5E752e039
Deployer	0xD74650b3e35673dE488DC94e6E305df5E752e039
SHA1-Hash	3f58431742f1d7caddb877f74cd0b55bd5ee6023
Decimals	9
Supply	1,950,000,000,000
Platform	Binance Smart Chain
Compiler	v0.8.4+commit.c7e474f2
Optimization	Yes with 200 runs
Website	https://breadtoken.finance/
Telegram	https://t.me/breadbakerycommunity
Twitter	https://twitter.com/BREADbakerycoin



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: 8%
- Sell Fees: 8%

Fees privileges

- Can't set fees above 25%

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

Other privileges

- Can exclude from fees
- Can exclude from rewards



CONTRACT INSPECTION 🔍

Imported contracts or frameworks used:

```
| **IERC20** | Interface | |||  
| **Context** | Implementation | |||  
| **Ownable** | Implementation | Context |||  
| **SafeMath** | Library | |||  
| **Address** | Library | |||  
| **IUniswapV2Router01** | Interface | |||  
| **IUniswapV2Router02** | Interface | IUniswapV2Router01 |||  
| **IUniswapV2Factory** | Interface | |||  
| **BaseToken** | Implementation | |||
```

Tested Contract File:

File Name	SHA-1 Hash
BreadBakery.sol	3f58431742f1d7caddb877f74cd0b55bd5ee6023

```
| **LiquidityGeneratorToken** | Implementation | IERC20, Ownable, BaseToken |||  
| L | <Constructor> | Public | | NO |  
| L | name | Public | | NO |  
| L | symbol | Public | | NO |  
| L | decimals | Public | | NO |  
| L | totalSupply | Public | | NO |  
| L | balanceOf | Public | | NO |  
| L | transfer | Public | | NO |  
| L | allowance | Public | | NO |  
| L | approve | Public | | NO |  
| L | transferFrom | Public | | NO |  
| L | increaseAllowance | Public | | NO |  
| L | decreaseAllowance | Public | | NO |  
| L | isExcludedFromReward | Public | | NO |  
| L | totalFees | Public | | NO |  
| L | deliver | Public | | NO |  
| L | reflectionFromToken | Public | | NO |  
| L | tokenFromReflection | Public | | NO |  
| L | excludeFromReward | Public | | onlyOwner |  
| L | includeInReward | External | | onlyOwner |  
| L | _transferBothExcluded | Private | | |  
| L | excludeFromFee | Public | | onlyOwner |  
| L | includeInFee | Public | | onlyOwner |  
| L | setTaxFeePercent | External | | onlyOwner |  
| L | setLiquidityFeePercent | External | | onlyOwner |  
| L | setSwapAndLiquifyEnabled | Public | | onlyOwner |  
| L | <Receive Ether> | External | | NO |  
| L | _reflectFee | Private | | |  
| L | _getValues | Private | | |  
| L | _getTValues | Private | | |  
| L | _getRValues | Private | | |  
| L | _getRate | Private | | |  
| L | _getCurrentSupply | Private | | |  
| L | _takeLiquidity | Private | | |  
| L | _takeCharityFee | Private | | |
```

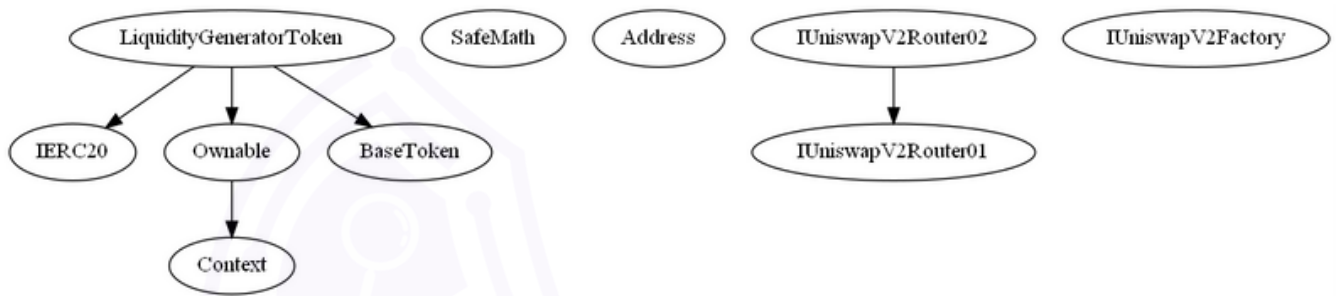


L	calculateTaxFee	Private	🔒			
L	calculateLiquidityFee	Private	🔒			
L	calculateCharityFee	Private	🔒			
L	removeAllFee	Private	🔒	🛑		
L	restoreAllFee	Private	🔒	🛑		
L	isExcludedFromFee	Public	!		NO	
L	_approve	Private	🔒	🛑		
L	_transfer	Private	🔒	🛑		
L	swapAndLiquify	Private	🔒	🛑	lockTheSwap	
L	swapTokensForEth	Private	🔒	🛑		
L	addLiquidity	Private	🔒	🛑		
L	_tokenTransfer	Private	🔒	🛑		
L	_transferStandard	Private	🔒	🛑		
L	_transferToExcluded	Private	🔒	🛑		
L	_transferFromExcluded	Private	🔒	🛑		

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	Passed
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	Minor
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)	Passed
V-37	Unencrypted Private Data On-Chain	Passed



FINDINGS

ID	Category	Issue	Severity
CE-OF	Centralization	Owner Accessible Functions	Minor
V-01	Vulnerabilities	State Variable Default Visibility	Minor
CS-01	Coding Standards	Dead Code	Informational



CE-OF: Owner Accessible Functions

Description

The owner has the permission through onlyOwner modifier to the following:

- 1.renounceOwnership()
- 2.transferOwnership()
- 3.excludeFromReward()
- 4.includeInReward()
- 5.excludeFromFee()
- 6.includeInFee()
- 7.setTaxFeePercent()
- 8.setLiquidityFeePercent()
- 9.setSwapAndLiquifyEnabled()

The role OnlyOwner has authority over the above functions that can manipulate the project functionality without restrictions. Any compromise to the owner account may allow a hacker to take advantage of this authority.

Recommendation

- We advise the client to carefully manage the privilege accounts' private key to avoid any potential risks of being hacked.
- Renounce Ownership at some point in time.



V-01: State Variable Default Visibility

Line #957

```
bool inSwapAndLiquify;
```

Description

Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.

Recommendation

Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables.



CS-01: Coding Standards

Dead Code

Description

Address library functions: *functionCall()*, *functionCallWithValue()*, *functionDelegateCall()*, *functionStaticCall()*, *isContract()*, *sendValue()*, *verifyCallResult()* are never used.

Some SafeMath library functions: *div()*, *mod()*, *tryAdd()*, *tryDiv()*, *tryMod*, *tryMul()*, *trySub()* are never used.

Recommandation

- Remove unused functions 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 above 25%
- The smart contract utilizes "SafeMath" to prevent overflows

```
function tryAdd(uint256 a, uint256 b) internal pure returns (bool, uint256) {
    unchecked {
        uint256 c = a + b;
        if (c < a) return (false, 0);
        return (true, c);
    }
}

function trySub(uint256 a, uint256 b) internal pure returns (bool, uint256) {
    unchecked {
        if (b > a) return (false, 0);
        return (true, a - b);
    }
}

function tryMul(uint256 a, uint256 b) internal pure returns (bool, uint256) {
    unchecked {
        if (a == 0) return (true, 0);
        uint256 c = a * b;
        if (c / a != b) return (false, 0);
        return (true, c);
    }
}

function tryDiv(uint256 a, uint256 b) internal pure returns (bool, uint256) {
    unchecked {
        if (b == 0) return (false, 0);
        return (true, a / b);
    }
}

function tryMod(uint256 a, uint256 b) internal pure returns (bool, uint256) {
    unchecked {
        if (b == 0) return (false, 0);
        return (true, a % b);
    }
}
```



Website	https://breadtoken.finance/
Domain Registry	http://www.one.com
Domain Expiry Date	2023-05-26
Response Code	200
SSL Checker and HTTPS Test	Passed
Deprecated HTML tags	Passed
Robots.txt	Informative
Sitemap Test	Passed
SEO Friendly URL	Passed
Responsive Test	Passed
JS Error Test	Passed
Console Errors Test	Passed
Site Loading Speed Test	2.52 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

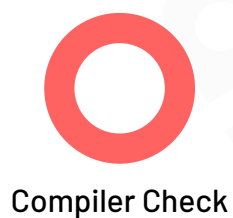
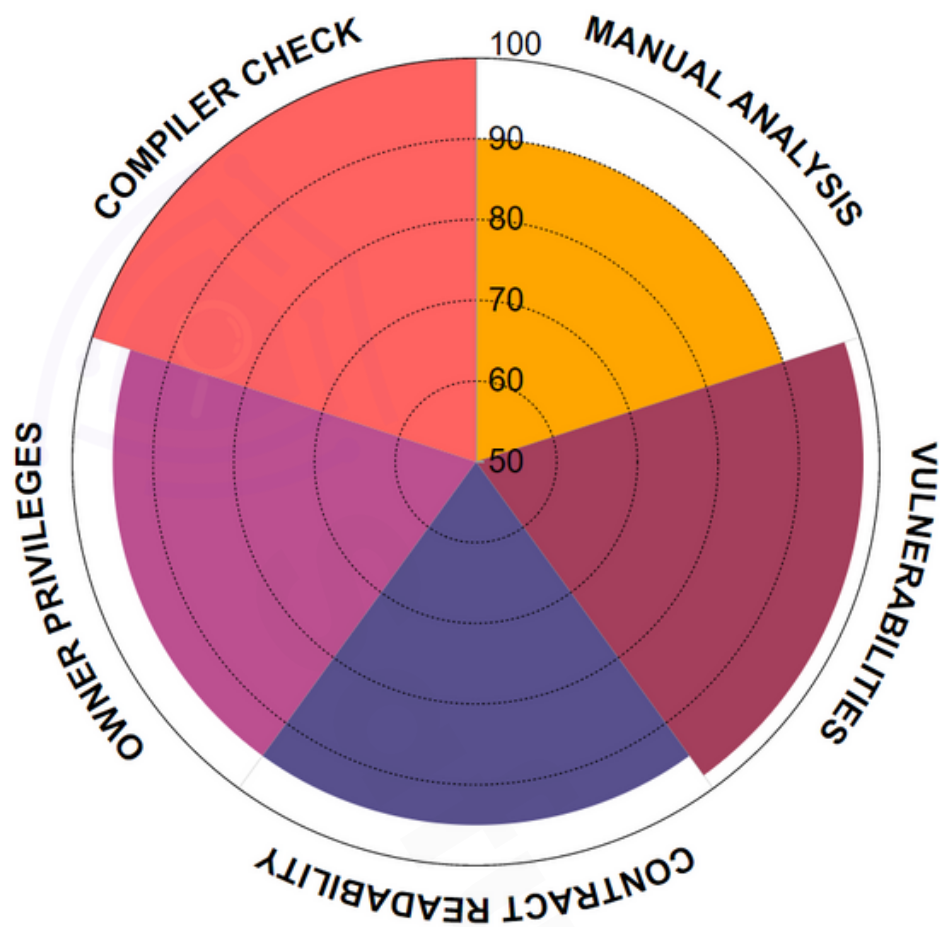
SafuAudit will strive to ensure the accuracy of the information listed on this website although it will not hold any responsibility for any missing or wrong information. SafuAudit provides all information as is. You understand that you are using any and all information available here at your own risk.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

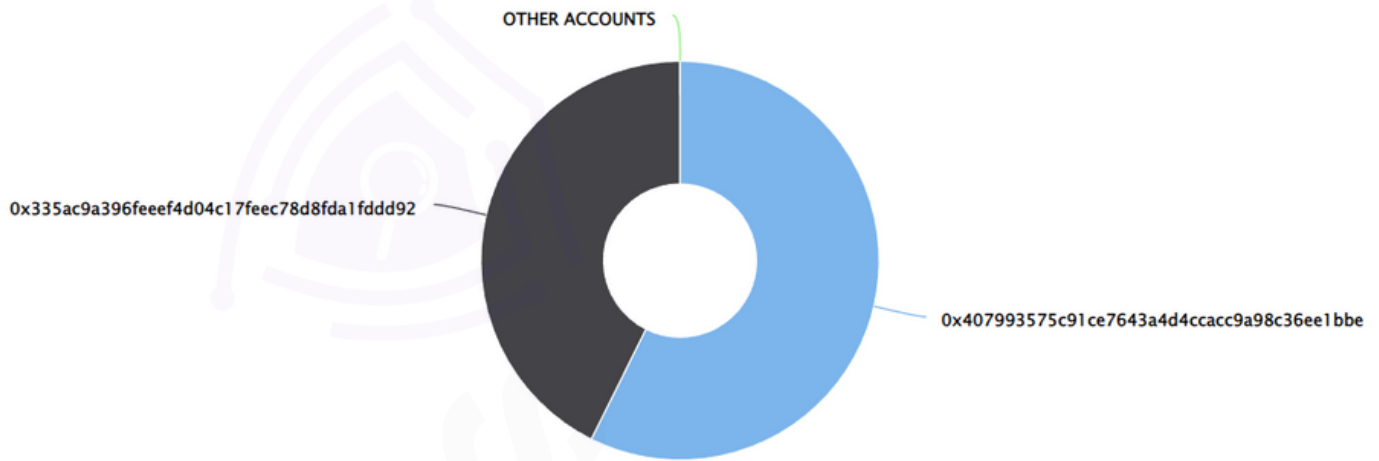


Final Score: **95.6**



SUMMARY

Top 10 holders



Rank	Address	Quantity (Token)	Percentage
1	0x407993575c91ce7643a4d4ccacc9a98c36ee1bbe	1,118,325,000,000	57.3500%
2	0x335ac9a396feef4d04c17fec78d8fda1fddd92	831,675,000,000	42.6500%

CONCLUSION

Project Bread Bakery (BREAD) 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

