RugFreeCoins Audit

Talkaboat Token Audit

Smart Contract Security Audit

September 11, 2021

Contents

Audit details

Audited project

Talkaboat Token

Contract Address

0xC283f97CAABE21Ba0eC76862970376B5E3E65ce8

Client contact

Talkaboat Token Team

Blockchain

Binance smart chain

Project website

https://talkaboat.online/

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Rugfreecoins and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Rugfreecoins) owe no duty of care towards you or any other person, nor does Rugfreecoins make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Rugfreecoins hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Rugfreecoins hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Rugfreecoins, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. 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.

Background

Rugfreecoins was commissioned by Talkaboat Token to perform an audit of the smart contract.

https://bscscan.com/address/0xC283f97CAABE21Ba0eC76862970376B5E3E65ce8

Master Entertainer:

https://bscscan.com/address/0x013b705e27F21EdC2040465841439bb65575b2DC#code

Reward System:

https://bscscan.com/address/0x319Cbc449E622Ef53b06dD1b720649207e5D13B4#code

The focus of this audit is to verify that the smart contract is secure, resilient and working according to the specifications.

The information in this report should be used to understand the risk exposure of the smart contract, project feasibility, long term sustainability and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

About the project

Talkaboat is a token built on the Binance Smart Chain. Each transaction, purchase and sale incur 1-5% fee proportional to how many tokens each individual accrue or sell.

The project is the first step into a community-driven entertainment ecosystem where fans and creators can share the same dream. It aims to drop the need of creators to search for a chance to monetize their passion! It brings users a solution where everyone can share their passion fully free with a monetization system that runs even if they don't move a finger!

Free for everyone

It's available free of charge for everyone.

Monetization for everyone

Both parties of content creators and consumers can earn from it.

Community Driven

The project team will bring community ideas into entertainment and thus more opportunities for interaction for content creators and fans!

Features

From fees getting deducted around 1-5% per each transaction

❖ The fee of 40% from all each transaction of buy and sell is sent to the reward wallet/reward Smart contract.

❖ 20% is sent to a wallet for ABOAT development. Having the resources necessary to deliver a meticulously marketed project, such as professional branding, paid influencers, gives Talkaboat the ability for substantial growth.

❖ 10% is sent to a wallet for donation to allocate for donations and charity.

❖ The liquidity fee of 30%, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity. This is a key element for decentralized exchanges like Pancakeswap.

Tokenomics

1-5% fee when buying & selling

Calculation for Fee: Max (5, Min (1, Your total balance/Total Talkaboat supply * 10)

- ❖ 40% of trade goes to holders pockets in token.
- ❖ 30% of trade goes to the liquidity pool.
- ❖ 20% of trade goes to marketing.
- ❖ 10% of trade goes to charity.

Roadmap

Q3 2021

- ❖ [x] Launch Website
- ❖ [x] Release Documentation/Whitepaper
- ❖ [x] Release Contracts (Testnet)
- ❖ [x] Preview Podcast-Player
- ❖ [x] Create Social Media Accounts

Q4 2021

- ❖ [x] Create Database
- ❖ [x] Release Contract (Mainnet)
- ❖ [ ] Incorporation
- ❖ [ ] Marketing Campaign
- ❖ [ ] Conclude Pre-Sale
- ❖ [ ] Profile creation
- ❖ [ ] Claim for podcasters
- ❖ [ ] Partnerships
- ❖ [ ] Improve Podcast-Search
- ❖ [ ] Lottery Release

Q1 2022

- ❖ [ ] Introduce podcaster-services (hosting)
- ❖ [ ] App Preview
- ❖ [ ] List on CEX
- ❖ [ ] Introduce Development Governance
- ❖ [ ] Add more features to Podcast-Player

Q2 2022

- ❖ [ ] Release Talkaboat-App
- ❖ [ ] Add Community-Groups to Podcast
- ❖ [ ] Introduce Community-Battles
- ❖

Q3 2022

- ❖ [ ] Integrate own Wallet to app

Q4 2022

- ❖ [ ] Preview of Singaboat

2023/2024

- ❖ [ ] Gaming Network
- ❖ [ ] Game Streaming

Target market and the concept

- ❖ Anyone who's interested in Crypto space with long term investment plans.
- ❖ Anyone who's ready to earn a passive income in Talkaboat tokens by holding tokens.
- ❖ Anyone who's interested in trading tokens.
- ❖ Anyone who's interested in listening to podcasts for free or fair market prices or even earn by listening to podcasts.
- ❖ Anyone who's interested in becoming a content creator in the Talkabot podcast platform and earn.
- ❖ Anyone who's interested in boosting income by doing daily Talkaboat tasks.
- ❖ Anyone who's interested in taking part with Talkaboat yield farming and earn rewards.
- ❖ Anyone who's interested in taking part with the voting system of Talkaboat and being part of the community decisions.
- ❖ Anyone who's interested in taking part with portfolio tracking and notification features that Talkaboat team is planning to build.
- ❖ Anyone who's interested in taking part with the future plans of the project.
- ❖ Anyone who's interested in making financial transactions with any other party using Talkaboat as the currency.

Core concept

Aboat Token will be used in all future products which is going be centered around all aspects of media entertainment. At the start there will be 3 ways to get new Aboat Tokens:

- ❖ Buying
- ❖ Listening
- ❖ Yield Farming

Viewers and listeners

Holders can earn Aboat Tokens simply by listening to podcasts. And if that wasn't enough, they can boost their income by doing daily tasks. Talkaboat aims to become easily accessible, so everyone can find the right podcasts of their interest.

Content creators

Content creators don't have to pay for any service they provide. On the contrary: They earn Aboat Tokens every time anyone listens to their podcasts. The team also supports content creators with different monetization methods, like premium content, branding, affiliate or advertising.

Elastic Yield Farming

The team provides users with a unique yield farming experience.

Example:
It starts with a reward per block of 100. Estimating that the binance smart chain verifies 28,800 blocks per day and distributes 2,880,000 Aboat Tokens per day.

From the total supply 40% is initially reserved for yield farming which equals 400,000,000,000 Talkaboat. So, to reach the total supply it needs 138,888 days or 380.50 years.

Since it's quite a long time, the Talkaboat team has come up with this elastic yield feature. If the average price of the last 24h raises, the reward per block raises too. If it goes down, the reward per block goes down too.

At first, they will set the maximum raise and drop to 2.5% per day.

Portfolio Tracking

Talkaboat Portfolio Tracker shall help users to keep an eye on your coins no matter where they are.

Alpaca Notifications

Alpaca Notifications is our first feature for the portfolio tracker. It allows users to place notifications on your alpaca leveraged farm positions.

Most leveraged farms get liquidated at ~ 80% debt ratio. If users don't want to check it 24/7 you can use our service and set a notification boundary. In case the debt ratio passes the notification, boundary users will get information and can increase collateral or close the position before it gets liquidated.

Voting system

The team will deploy a voting system, where users can request and vote for features.

The reward system

1-5% of each transaction when buying and selling gets sent to the reward wallet. The sending percentage from the fees will be 40. The reward wallet will be used to provide incentives for the holders and anyone who's giving interest over the project as a content creator or a viewer.

Sustainable mechanism

The liquidity fee of 30% from total accumulated fees, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

The **distribution amount of 20% for development from 1-5% from fee** is what allows Talkaboat to promote the token and use funds to further development of the platform.

Good cause

Donations will be collected by **1-5% fees and 10%** are stored in a smart contract from the total fee with full governance authority by the community.

The community decides who will receive the donations.

Potential to grow with score points

1.	Project efficiency	10/10
2.	Project uniqueness	10/10
3	Information quality	9/10
4	Service quality	10/10
5	System quality	8/10
6	Impact on the community	10/10
7	Impact on the business	9/10
8	Preparing for the future	9/10
Total Points		**9.38/10**

Contract details

Token contract details for 11ᵗʰ September 2021

Contract name	Talkaboat
Contract address	0xC283f97CAABE21Ba0eC76862970376B5E3E65ce8
Token supply	600,000,000,000
Token ticker	ABOAT
Decimals	18
Token holders	1
Transaction count	1
Dev wallet	0x2ea9ca0ca8043575f2189cff9897b575b0c7e857
Donation wallet	0xa7c08aedce8cadc3bfb622bd7b651993d1cd24e4
Reward wallet	0x2ea9ca0ca8043575f2189cff9897b575b0c7e857
Contract deployer address	0x912E637522dC32A9653c5B7FD9028c7Ae3ae3Ff6
Contract's current owner address	0x912e637522dc32a9653c5b7fd9028c7ae3ae3ff6

Token distribution

Tokens are distributed as follows:
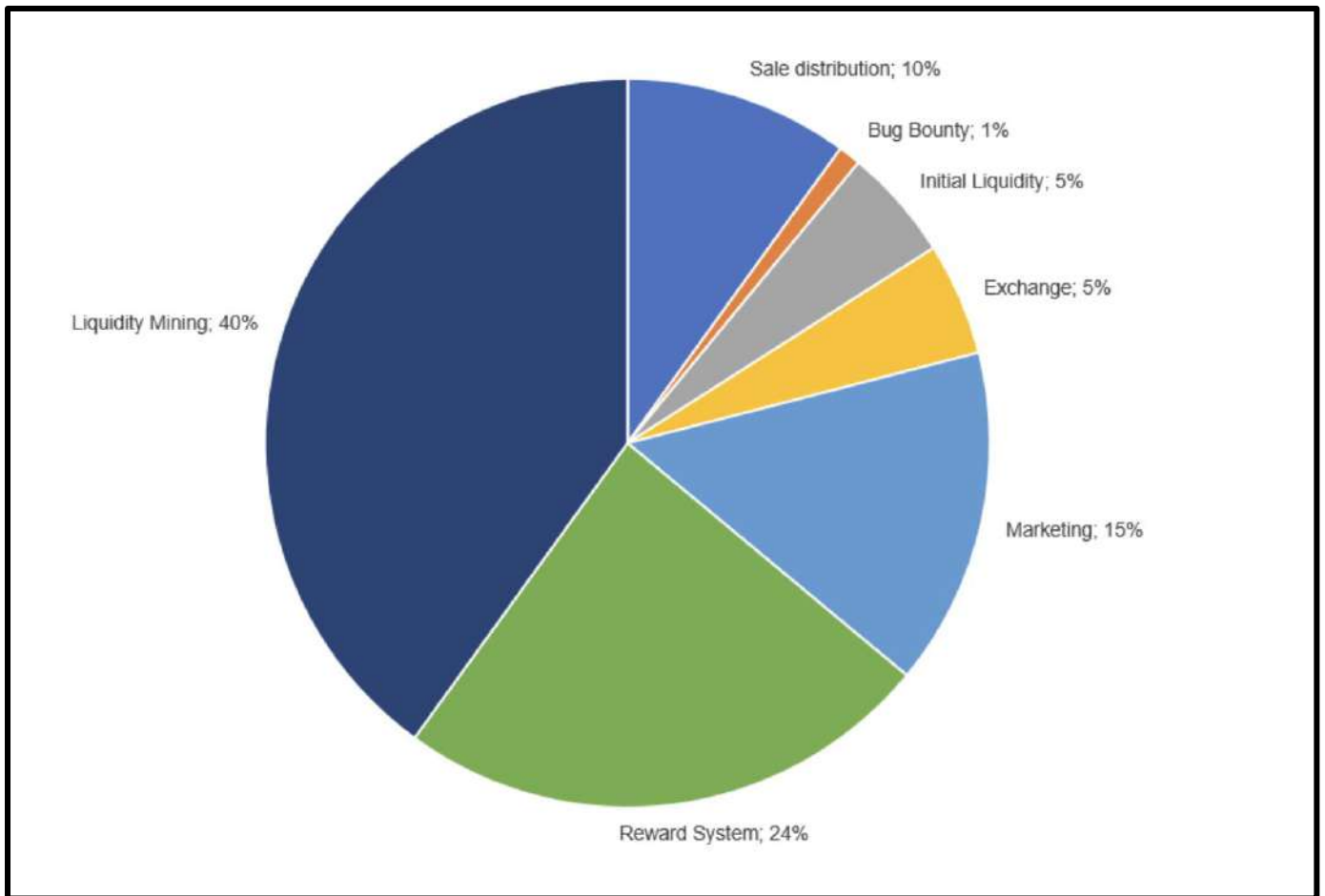
ICO/IDO is separated into three sales.

- ❖ Private Sale: 15 %
- ❖ ICO/IDO: 40 %
- ❖ Public Sale: 45 %

The income of the single stages will be distributed by following scheme:

Sale Type	Limit	Price per coin	Softcap	Date	Time
Private Sale	min 50 BUSD/max 2000 BUSD	$0.0000065	$60,000	19.09.2021	TBA
IDO	TBA	$0.0000075	$100,000	TBA	TBA
Public Sale	TBA	$0.0000085	$120,000	TBA	TBA

Sale Type	Liquidity	Marketing	Audit	Development
Private Sale	0 %	45 %	25 % (CertiK)	30 %
IDO	80 %	20 %	0 %	0 %
Public Sale	50 %	10 %	0 %	40 %

❖ Estimated Liquidity on release: $425,000

   • At 100 % sold coins in IDO & Public Sale

❖ Release Price: ~ $0.0000085

Vesting

❖ There will be no vesting of Dev Token, as we don't receive any.
❖ Private Sale vesting: We don't have a vesting schedule on private sale funds, as we use them for

   • Incorporation (Dev Funds)
   • Increased Marketing for IDO and public sale
   • Increased Security with Audit

❖ No vesting on IDO as the most of it will be locked for liquidity and liquidity tokens will be burnt.
❖ Public Sale Dev Token will be vested over 1 year with 10 % direct release.
❖ Marketing Tokens will be vested over 2 years with 10 % instant release, starting October 2021.

Contract code function details

No	Category	Item	Result
1	Coding conventions	BRC20 Token standards	pass
		compile errors	pass
		Compiler version security	pass
		visibility specifiers	pass
		Gas consumption	pass
		SafeMath features	pass
		Fallback usage	pass
		tx.origin usage	pass
		deprecated items	pass
		Redundant code	pass
		Overriding variables	pass
2	Function call audit	Authorization of function call	pass
		Low level function (call/delegate call) security	pass
		Returned value security	pass
		Selfdestruct function security	pass
3	Business security	Access control of owners	pass
		Business logics	pass
		Business implementations	pass
4	Integer overflow/underflow		pass
5	Reentrancy		pass
6	Exceptional reachable state		pass
7	Transaction ordering dependence		pass
8	Block properties dependence		pass
9	Pseudo random number generator (PRNG)		pass
10	DoS (Denial of Service)		pass
11	Token vesting implementation		pass
12	Fake deposit		pass
13	Event security		pass

Contract description table

Below table represents the summary of the contracts and methods in the token contract. We scanned the whole contract and listed down all the Interfaces, functions and implementations with its visibility and mutability.

Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
ReentrancyGuard	**Implementation**			
L		Public 〖	⬢	NO〖
Context	**Implementation**			
L	_msgSender	Internal 🔒		
L	_msgData	Internal 🔒		
IERC20	**Interface**			
L	totalSupply	External 〖		NO〖
L	balanceOf	External 〖		NO〖
L	transfer	External 〖	⬢	NO〖
L	allowance	External 〖		NO〖
L	approve	External 〖	⬢	NO〖
L	transferFrom	External 〖	⬢	NO〖

TransferHelper	Library			
L	safeApprove	Internal 🔒	⬤	
L	safeTransfer	Internal 🔒	⬤	
L	safeTransferFrom	Internal 🔒	⬤	
L	safeTransferETH	Internal 🔒	⬤	
IUniswapV2Factory	**Interface**			
L	feeTo	External ❗️		NO❗️
L	feeToSetter	External ❗️		NO❗️
L	getPair	External ❗️		NO❗️
L	allPairs	External ❗️		NO❗️
L	allPairsLength	External ❗️		NO❗️
L	createPair	External ❗️	⬤	NO❗️
L	setFeeTo	External ❗️	⬤	NO❗️
L	setFeeToSetter	External ❗️	⬤	NO❗️
IUniswapV2Pair	**Interface**			
L	name	External ❗️		NO❗️
L	symbol	External ❗️		NO❗️

L	decimals	External ❚		NO❚
L	totalSupply	External ❚		NO❚
L	balanceOf	External ❚		NO❚
L	allowance	External ❚		NO❚
L	approve	External ❚	⬢	NO❚
L	transfer	External ❚	⬢	NO❚
L	transferFrom	External ❚	⬢	NO❚
L	DOMAIN_SEPARATOR	External ❚		NO❚
L	PERMIT_TYPEHASH	External ❚		NO❚
L	nonces	External ❚		NO❚
L	permit	External ❚	⬢	NO❚
L	MINIMUM_LIQUIDITY	External ❚		NO❚
L	factory	External ❚		NO❚
L	token0	External ❚		NO❚
L	token1	External ❚		NO❚
L	getReserves	External ❚		NO❚
L	price0CumulativeLast	External ❚		NO❚
L	price1CumulativeLast	External ❚		NO❚

L	kLast	External ❗️		NO❗️
L	mint	External ❗️	⬤	NO❗️
L	burn	External ❗️	⬤	NO❗️
L	swap	External ❗️	⬤	NO❗️
L	skim	External ❗️	⬤	NO❗️
L	sync	External ❗️	⬤	NO❗️
L	initialize	External ❗️	⬤	NO❗️
IUniswapV2Router 01	**Interface**			
L	factory	External ❗️		NO❗️
L	WETH	External ❗️		NO❗️
L	addLiquidity	External ❗️	⬤	NO❗️
L	addLiquidityETH	External ❗️	💵	NO❗️
L	removeLiquidity	External ❗️	⬤	NO❗️
L	removeLiquidityETH	External ❗️	⬤	NO❗️
L	removeLiquidityWithPermit	External ❗️	⬤	NO❗️
L	removeLiquidityETHWithPermit	External ❗️	⬤	NO❗️
L	swapExactTokensForTokens	External ❗️	⬤	NO❗️

18

L	swapTokensForExact Tokens	External ▯	⬢	NO▯
L	swapExactETHForTok ens	External ▯	▥	NO▯
L	swapTokensForExact ETH	External ▯	⬢	NO▯
L	swapExactTokensFor ETH	External ▯	⬢	NO▯
L	swapETHForExactTok ens	External ▯	▥	NO▯
L	quote	External ▯		NO▯
L	getAmountOut	External ▯		NO▯
L	getAmountIn	External ▯		NO▯
L	getAmountsOut	External ▯		NO▯
L	getAmountsIn	External ▯		NO▯
IUniswapV2Router02	**Interface**	**IUniswapV2Router 01**		
L	removeLiquidityETHS upportingFeeOnTrans ferTokens	External ▯	⬢	NO▯
L	removeLiquidityETHW ithPermitSupportingFe eOnTransferTokens	External ▯	⬢	NO▯
L	swapExactTokensFor TokensSupportingFee OnTransferTokens	External ▯	⬢	NO▯
L	swapExactETHForTok ensSupportingFeeOn TransferTokens	External ▯	▥	NO▯

L	swapExactTokensFor ETHSupportingFeeOn TransferTokens	External 〚	⬢	NO〚
SafeMath	**Library**			
L	tryAdd	Internal 🔒		
L	trySub	Internal 🔒		
L	tryMul	Internal 🔒		
L	tryDiv	Internal 🔒		
L	tryMod	Internal 🔒		
L	add	Internal 🔒		
L	sub	Internal 🔒		
L	mul	Internal 🔒		
L	div	Internal 🔒		
L	mod	Internal 🔒		
L	sub	Internal 🔒		
L	div	Internal 🔒		
L	mod	Internal 🔒		
Ownable	**Implementation**	**Context**		
L		Public 〚	⬢	NO〚

L	owner	Public 🛡		NO🛡
L	renounceOwnership	Public 🛡	⬤	onlyOwner
L	transferOwnership	Public 🛡	⬤	onlyOwner
L	_setOwner	Private 🔐	⬤	
Address	**Library**			
L	isContract	Internal 🔒		
L	sendValue	Internal 🔒	⬤	
L	functionCall	Internal 🔒	⬤	
L	functionCall	Internal 🔒	⬤	
L	functionCallWithValue	Internal 🔒	⬤	
L	functionCallWithValue	Internal 🔒	⬤	
L	functionStaticCall	Internal 🔒		
L	functionStaticCall	Internal 🔒		
L	functionDelegateCall	Internal 🔒	⬤	
L	functionDelegateCall	Internal 🔒	⬤	
L	verifyCallResult	Internal 🔒		
IERC20Metadata	**Interface**	**IERC20**		

L	name	External 〚		NO〛
L	symbol	External 〚		NO〛
L	decimals	External 〚		NO〛
ERC20	**Implementation**	**Context, IERC20, IERC20Metadata**		
L		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	_transfer	Internal 🔒	⬢	

L	_mint	Internal 🔒	⬣	
L	_burn	Internal 🔒	⬣	
L	_approve	Internal 🔒	⬣	
L	_beforeTokenTransfer	Internal 🔒	⬣	
L	_afterTokenTransfer	Internal 🔒	⬣	
TimeLock	**Implementation**	**Ownable**		
L		Public ❗	⬣	NO❗
L	maintainer	Public ❗		NO❗
L	setMaintainer	Public ❗	⬣	onlyMaintainerOrOwner locked
L	setTimelockEnabled	Public ❗	⬣	onlyMaintainerOrOwner
L	unlockFunction	Public ❗	⬣	onlyMaintainerOrOwner
L	lockFunction	Public ❗	⬣	onlyMaintainerOrOwner
PriceTicker	**Implementation**	**Ownable, TimeLock**		
L		Public ❗	⬣	NO❗
L	setCoin	Public ❗	⬣	onlyOwner locked

└	getAveragePrice	Public ▯		NO▯
└	getPriceDifference	Public ▯		NO▯
└	getTokenPrice	Public ▯	⬤	NO▯
└	updateLastAveragePrice	Internal 🔒	⬤	
└	checkPriceUpdate	Public ▯	⬤	NO▯
SafeERC20	**Library**			
└	safeTransfer	Internal 🔒	⬤	
└	safeTransferFrom	Internal 🔒	⬤	
└	safeApprove	Internal 🔒	⬤	
└	safeIncreaseAllowance	Internal 🔒	⬤	
└	safeDecreaseAllowance	Internal 🔒	⬤	
└	_callOptionalReturn	Private 🔐	⬤	
MasterEntertainer	**Implementation**	**Ownable, ReentrancyGuard, PriceTicker**		
└		Public ▯	⬤	NO▯
└	setDevAddress	Public ▯	⬤	onlyOwner locked
└	massUpdatePools	Public ▯	⬤	NO▯

24

L	setPoolVariables	Public 〚	⬣	onlyOwner
L	updateEmissionRate	Public 〚	⬣	onlyOwner locked
L	updateEmissionRateInternal	Internal 🔒	⬣	
L	setMaxEmissionIncrease	Public 〚	⬣	onlyOwner
L	poolLength	External 〚		NO〚
L	canClaimRewards	Public 〚		NO〚
L	getNewEmissionRate	Public 〚		NO〚
L	getLpSupply	Public 〚		NO〚
L	pendingCoin	External 〚		NO〚
L	add	Public 〚	⬣	onlyOwner nonDuplicated
L	updatePool	Public 〚	⬣	NO〚
L	deposit	Public 〚	⬣	nonReentrant
L	withdraw	Public 〚	⬣	nonReentrant
L	claim	Public 〚	⬣	nonReentrant
L	withdrawWithoutRewards	Public 〚	⬣	nonReentrant
L	safeCoinTransfer	Internal 🔒	⬣	
L	checkPriceUpdate	Public 〚	⬣	NO〚
L	updateEmissionRateByPriceDifference	Internal 🔒	⬣	

Liquify	Implementation	ERC20, ReentrancyGuard, Ownable, TimeLock		
L		Public 〗	⬢	NO〗
L	setLiquidityPair	Public 〗	⬢	onlyMaintainerOrOwner locked
L	setDevWallet	Public 〗	⬢	onlyMaintainerOrOwner
L	setDonationWallet	Public 〗	⬢	onlyMaintainerOrOwner
L	setRewardWallet	Public 〗	⬢	onlyMaintainerOrOwner
L	disableLiquify	Public 〗	⬢	onlyMaintainerOrOwner
L	enableLiquify	Public 〗	⬢	onlyMaintainerOrOwner
L	excludeFromAll	Public 〗	⬢	onlyMaintainerOrOwner
L	excludeTransferFeeAsSender	Public 〗	⬢	onlyMaintainerOrOwner
L	excludeFromFeesAsReciever	Public 〗	⬢	onlyMaintainerOrOwner
L	includeForAll	Public 〗	⬢	onlyMaintainerOrOwner

L	Function	Visibility		Modifiers
L	includeTransferFeeAsSender	Public 〚	⬤	onlyMaintainerOrOwner
L	includeForFeesAsReciever	Public 〚	⬤	onlyMaintainerOrOwner
L	updateMinimumTransferTaxRate	Public 〚	⬤	onlyMaintainerOrOwner locked
L	updateMaximumTransferTaxRate	Public 〚	⬤	onlyMaintainerOrOwner locked
L	updateRedistributionRate	Public 〚	⬤	onlyMaintainerOrOwner locked
L	updateDevRate	Public 〚	⬤	onlyMaintainerOrOwner locked
L	updateDonationRate	Public 〚	⬤	onlyMaintainerOrOwner locked
L	updateRouter	Public 〚	⬤	onlyMaintainerOrOwner locked
L	swapAndLiquify	Public 〚	⬤	taxFree
L	swapForEth	Private 🔐	⬤	
L	swapAndLiquifyEth	Private 🔐	⬤	
L	swapAndLiquifyTokens	Private 🔐	⬤	
L	swapEthForTokens	Private 🔐	⬤	
L	swapTokensForEth	Private 🔐	⬤	
L	addLiquidity	Internal 🔒	⬤	

AboatToken	Implementation	ERC20, Liquify		
L	addLiquidityETH	Internal 🔒	⬢	
AboatToken	**Implementation**	**ERC20, Liquify**		
L		Public 🗒	⬢	ERC20
L	setMasterEntertainer	Public 🗒	⬢	onlyMaintainerOrOwner locked
L	setMaxDistribution	Public 🗒	⬢	onlyMaintainerOrOwner locked
L	setMaxAccBalance	Public 🗒	⬢	onlyMaintainerOrOwner
L	setMaxTransactionQuantity	Public 🗒	⬢	onlyMaintainerOrOwner
L	setGasCost	Public 🗒	⬢	onlyMaintainerOrOwner
L	isExcludedFromSenderTax	Public 🗒		NO🗒
L	isExcludedFromRecieverTax	Public 🗒		NO🗒
L	hasRequestedWhitelist	Public 🗒		onlyMaintainerOrOwner
L	getTaxFee	Public 🗒		NO🗒
L	liquidityPair	Public 🗒		NO🗒
L	getLiquidityTokenAddress	Public 🗒		NO🗒
L	liquidityTokenBalance	Public 🗒		NO🗒

L	canMintNewCoins	Public 〚		NO〚
L		External 〚	🔲	NO〚
L	activateHighFee	Public 〚	⬤	onlyMaintainerOrOwner
L	deactivateHighFee	Public 〚	⬤	onlyMaintainerOrOwner
L	blacklist	Public 〚	⬤	onlyMaintainerOrOwner
L	whitelist	Public 〚	⬤	onlyMaintainerOrOwner
L	requestWhitelist	Public 〚	🔲	NO〚
L	claimExceedingLiquidityTokenBalance	Public 〚	⬤	onlyMaintainerOrOwner
L	mint	Public 〚	⬤	onlyOwner
L	burn	Public 〚	⬤	onlyMaintainerOrOwner
L	_transfer	Internal 🔒	⬤	
L	checkPriceUpdate	Public 〚	⬤	NO〚

Legend

Symbol	Meaning
🔴	**Function can modify state**
🟨	**Function is payable**

Inheritance Hierarchy
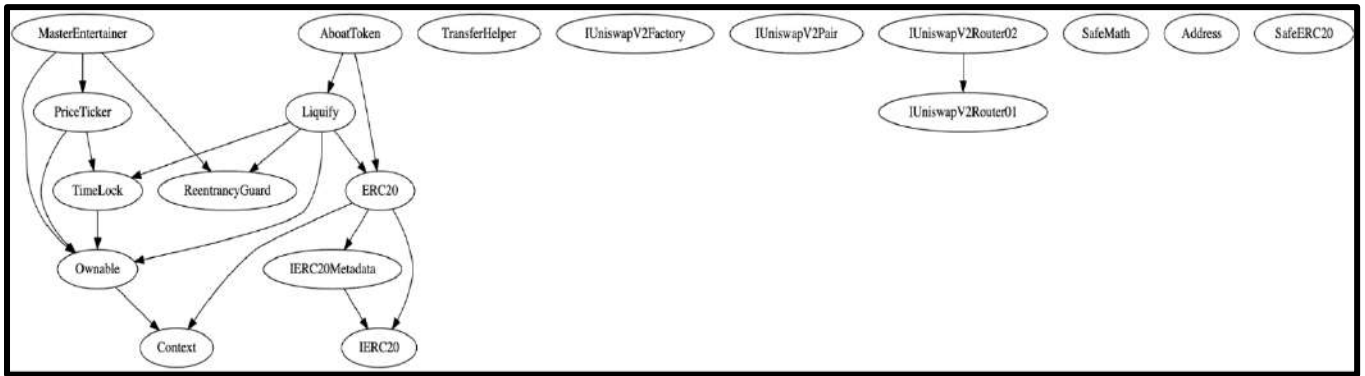
Security issue checking status

❖ **High severity issues**

- **No high severity issues found.**

❖ **Medium severity issues**

- **No medium severity issues found.**

❖ **Low severity issues**

- **No low severity issues found.**

Owner privileges

❖ The owner can renounce and transfer ownership.

```
ftrace | funcSig
function renounceOwnership() public virtual onlyOwner {
    _setOwner(address(0));
}


/**
 * @dev Transfers ownership of the contract to a new account (`newOwner`).
 * Can only be called by the current owner.
 */
ftrace | funcSig
function transferOwnership(address newOwner↑) public virtual onlyOwner {
    require(
        newOwner↑ != address(0),
        "Ownable: new owner is the zero address"
    );
    _setOwner(newOwner↑);
}
```

❖ The owner can add a new maintainer.

```
ftrace | funcSig
function setMaintainer(address _newMaintainer↑)
    public
    onlyMaintainerOrOwner
    locked("maintainer")
{
    require(
        _newMaintainer↑ != _maintainer && _newMaintainer↑ != address(0),
        "ABOAT::setMaintainer: Maintainer can't equal previous maintainer or zero address"
    );
    address previousMaintainer = _maintainer;
    _maintainer = _newMaintainer↑;
    emit MaintainerTransferred(previousMaintainer, _maintainer);
}
```

❖ The owner can enable time lock.

```
ftrace | funcSig
function setTimelockEnabled() public onlyMaintainerOrOwner {
    isLockEnabled = true;
    emit EnabledLock();
}
```

❖ The owner can lock and unlock functions.

```
//unlock timelock
ftrace | funcSig
function unlockFunction(string memory _fn) public onlyMaintainerOrOwner {
    timelock[_fn] = block.timestamp + _TIMELOCK;
    emit UnlockedFunction(_fn, timelock[_fn]);
}


//lock timelock
ftrace | funcSig
function lockFunction(string memory _fn) public onlyMaintainerOrOwner {
    timelock[_fn] = 0;
}
```

❖ The owner can change pool variables.

```
ftrace | funcSig
function setPoolVariables(
    uint256 _pid,
    uint256 _allocPoint,
    uint16 _depositFee,
    bool _withUpdate
) public onlyOwner {
    require(_depositFee <= 10000, "set: deposit fee can't exceed 10 %");
    if (_withUpdate) {
        massUpdatePools();
    }
    totalAllocPoint = totalAllocPoint.sub(poolInfos[_pid].allocPoint).add(
        _allocPoint
    );
    poolInfos[_pid].allocPoint = _allocPoint;
    poolInfos[_pid].depositFee = _depositFee;
}
```

❖ The owner can update the Emission rate.

```
ftrace | funcSig
function updateEmissionRate(uint256 _coinPerBlock↑)
    public
    onlyOwner
    locked("updateEmissionRate")
{
    massUpdatePools();
    coinPerBlock = _coinPerBlock↑;
    emit UpdateEmissionRate(msg.sender, _coinPerBlock↑);
}

ftrace | funcSig
function updateEmissionRateInternal(uint256 _coinPerBlock↑) internal {
    massUpdatePools();
    coinPerBlock = _coinPerBlock↑;
    emit UpdateEmissionRate(address(this), _coinPerBlock↑);
}
```

❖ The owner can change the max distribution amount.

```
ftrace | funcSig
function setMaxDistribution(uint256 _newDistribution↑)
    public
    onlyMaintainerOrOwner
    locked("max_distribution")
{
    require(
        _newDistribution↑ > totalSupply(),
        "ABOAT::setMaxDistribution: Distribution can't be lower than the current total supply"
    );
    maxDistribution = _newDistribution↑;
    emit MaxDistributionChanged(_newDistribution↑);
}
```

❖ The owner can change max acc balance.

```
ftrace | funcSig
function setMaxAccBalance(uint16 maxBalance↑) public onlyMaintainerOrOwner {
    uint16 previousMaxBalance = maxAccBalance;
    maxAccBalance = maxBalance↑;
    emit MaxAccBalanceChanged(previousMaxBalance, maxAccBalance);
}
```

❖ The owner can change max transaction Quantity.

```
ftrace | funcSig
function setMaxTransactionQuantity(uint16 quantity⬆)
    public
    onlyMaintainerOrOwner
{

    uint16 previous = maxTxQuantity;
    maxTxQuantity = quantity⬆;
    emit MaxTransactionQuantityChanged(previous, maxTxQuantity);

}
```

❖ The owner can change gas costs.

```
ftrace | funcSig
function setGasCost(uint256 cost⬆) public onlyMaintainerOrOwner {
    uint256 previous = gasCost;
    gasCost = cost⬆;
    emit GasCostChanged(previous, gasCost);

}
```

❖ The owner can activate and deactivate high fees.

```
ftrace | funcSig
function activateHighFee() public onlyMaintainerOrOwner {
    isHighFeeActive = true;
    emit ChangedHighFeeState(isHighFeeActive);
}


ftrace | funcSig
function deactivateHighFee() public onlyMaintainerOrOwner {
    isHighFeeActive = false;
    isContractActive = true;
    emit ChangedHighFeeState(isHighFeeActive);

}
```

❖ The owner can blacklist and whitelist users.

```
ftrace | funcSig
function blacklist(address user↑) public onlyMaintainerOrOwner {
    blacklisted[user↑] = true;
    emit Blacklisted(user↑);
}

ftrace | funcSig
function whitelist(address user↑) public onlyMaintainerOrOwner {
    blacklisted[user↑] = true;
    requestedWhitelist[user↑] = false;
}
```

❖ The owner can claim exceeding liquidity balance.

```
ftrace | funcSig
function claimExceedingLiquidityTokenBalance()
    public
    onlyMaintainerOrOwner
{
    require(
        liquidityTokenBalance() > 0,
        "ABOAT::claimExceedingLiquidityTokenBalance: No exceeding balance"
    );
    TransferHelper.safeTransferFrom(
        getLiquidityTokenAddress(),
        address(this),
        msg.sender,
        liquidityTokenBalance()
    );
}
```

❖ The owner can mint and burn tokens.

```
ftrace | funcSig
function mint(address _to↑, uint256 _amount↑) public onlyOwner {
    require(
        canMintNewCoins(_amount↑),
        "ABOAT::mint: Can't mint more aboat token than maxDistribution allows"
    );
    _mint(_to↑, _amount↑);
}


ftrace | funcSig
function burn(uint256 _amount↑) public onlyMaintainerOrOwner {
    require(
        _amount↑ <= balanceOf(address(this)),
        "ABOAT::burn: amount exceeds balance"
    );
    _burn(address(this), _amount↑);
}
```

❖ The owner can change dev/donation and reward wallets.

```
ftrace | funcSig
function setDevWallet(address wallet↑) public onlyMaintainerOrOwner {
    require(
        wallet↑ != address(0),
        "ABOAT::setDevWallet: Address can't be zero address"
    );
    _devWallet = wallet↑;
    excludeFromAll(_devWallet);
}


ftrace | funcSig
function setDonationWallet(address wallet↑) public onlyMaintainerOrOwner {
    require(
        wallet↑ != address(0),
        "ABOAT::setDevWallet: Address can't be zero address"
    );
    _donationWallet = wallet↑;
}


ftrace | funcSig
function setRewardWallet(address wallet↑) public onlyMaintainerOrOwner {
    require(
        wallet↑ != address(0),
        "ABOAT::setDevWallet: Address can't be zero address"
    );
    _rewardWallet = wallet↑;
    excludeFromAll(_rewardWallet);
}
```

❖ The owner can enable and disable liquidity add.

```
ftrace | funcSig
function disableLiquify() public onlyMaintainerOrOwner {
    isLiquifyDisabled = true;
}


ftrace | funcSig
function enableLiquify() public onlyMaintainerOrOwner {
    isLiquifyDisabled = false;
}
```

❖ The owner can exclude wallets fees from send/receive tokens or both.

```
ftrace | funcSig
function excludeFromAll(address _excludee) public onlyMaintainerOrOwner {
    _excludedFromFeesAsSender[_excludee] = true;
    _excludedFromFeesAsReciever[_excludee] = true;
}


ftrace | funcSig
function excludeTransferFeeAsSender(address _excludee)
    public
    onlyMaintainerOrOwner
{
    _excludedFromFeesAsSender[_excludee] = true;
}


ftrace | funcSig
function excludeFromFeesAsReciever(address _excludee)
    public
    onlyMaintainerOrOwner
{
    _excludedFromFeesAsReciever[_excludee] = true;
}
```

❖ The owner can include wallets from send/receive tokens or both.

```
ftrace | funcSig
function includeForAll(address _excludee↑) public onlyMaintainerOrOwner {
    _excludedFromFeesAsSender[_excludee↑] = false;
    _excludedFromFeesAsReciever[_excludee↑] = false;
}


ftrace | funcSig
function includeTransferFeeAsSender(address _excludee↑)
    public
    onlyMaintainerOrOwner
{

    _excludedFromFeesAsSender[_excludee↑] = false;
}


ftrace | funcSig
function includeForFeesAsReciever(address _excludee↑)
    public
    onlyMaintainerOrOwner
{

    _excludedFromFeesAsReciever[_excludee↑] = false;
}
```

❖ The owner can update the minimum transfer tax rate.

```
ftrace | funcSig
function updateMinimumTransferTaxRate(uint16 _transferTaxRate↑)
    public
    onlyMaintainerOrOwner
    locked("min_tax")
{
    require(
        _transferTaxRate↑ <= maximumTransferTaxRate,
        "ABOAT::updateMinimumTransferTaxRate: minimumTransferTaxRate must not exceed maximumTransferTaxRate."
    );
    emit MinimumTransferTaxRateUpdated(
        msg.sender,
        minimumTransferTaxRate,
        _transferTaxRate↑
    );
    minimumTransferTaxRate = _transferTaxRate↑;
}
```

❖ The owner can update the maximum transfer tax rate.

```
ftrace | funcSig
function updateMaximumTransferTaxRate(uint16 _transferTaxRate↑)
    public
    onlyMaintainerOrOwner
    locked("max_tax")
{
    require(
        _transferTaxRate↑ >= minimumTransferTaxRate,
        "ABOAT::updateMaximumTransferTaxRate: maximumTransferTaxRate must not be below minimumTransferTaxRate."
    );
    require(
        _transferTaxRate↑ <= MAXIMUM_TAX,
        "ABOAT::updateMaximumTransferTaxRate: maximumTransferTaxRate must exceed MAXIMUM_TAX."
    );
    emit MaximumTransferTaxRateUpdated(
        msg.sender,
        minimumTransferTaxRate,
        _transferTaxRate↑
    );
    maximumTransferTaxRate = _transferTaxRate↑;
}
```

❖ The owner can update the redistribution rate.

```
ftrace | funcSig
function updateRedistributionRate(uint16 _rate↑)
    public
    onlyMaintainerOrOwner
    locked("redistribution_rate")
{
    require(
        _rate↑ + devRate + donationRate <= 100,
        "ABOAT::updateRedistributionRate: Redistribution rate must not exceed the maximum rate."
    );
    emit ReDistributionRateUpdated(msg.sender, reDistributionRate, _rate↑);
    reDistributionRate = _rate↑;
}
```

❖ The owner can update dev and donation rate.

```
ftrace | funcSig
function updateDevRate(uint16 _rate⬆)
    public
    onlyMaintainerOrOwner
    locked("dev_rate")
{
    require(
        _rate⬆ + donationRate + reDistributionRate <= 100,
        "ABOAT::updateDevRate: Burn rate must not exceed the maximum rate."
    );
    emit DevRateUpdated(msg.sender, devRate, _rate⬆);
    devRate = _rate⬆;
}


ftrace | funcSig
function updateDonationRate(uint16 _rate⬆)
    public
    onlyMaintainerOrOwner
    locked("donation_rate")
{
    require(
        _rate⬆ + devRate + reDistributionRate <= 100,
        "ABOAT::updateDonationRate: Burn rate must not exceed the maximum rate."
    );
    emit DonationRateUpdated(msg.sender, donationRate, _rate⬆);
    donationRate = _rate⬆;
}
```

❖ The owner can update the v2 router address.

```
ftrace | funcSig
function updateRouter(address router⬆)
    public
    onlyMaintainerOrOwner
    locked("router")
{
    _router = IUniswapV2Router02(router⬆);
    setLiquidityPair(_router.WETH());
    excludeTransferFeeAsSender(router⬆);
    emit RouterUpdated(msg.sender, router⬆, _liquidityPair);
}
```

Audit conclusion

While conducting the audit of the Talkaboat Token smart contract, it was observed that there is nothing alarming with the code.