RugFreeCoins Audit

Joker Token Audit

Smart Contract Security Audit

September 11, 2021

Contents

Audit details

Audited project

Joker Token

Contract Address

0xed554ce4d15d83858f57e9996296fcd742e76e24

Client contact

Joker Token Team

Blockchain

Binance smart chain

Project website

https://www.jokercoin.live/

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 Joker Token to perform an audit of the smart contract.

https://bscscan.com/token/0xed554ce4d15d83858f57e9996296fcd742e76e24

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

Joker Token is a token built on the Binance Smart Chain. Each transaction, purchase, and sale incur a 12% fee.

Joker is a coin created by a development team of 16 people who are behind the creation of an e-game peer-to-peer card game called JokerBank. The Joker is based on a community of people.

The goal of Joker is to create a strong community while working to expand its own Joker platform with the JokerBank pilot project already in operation. Joker will bring fresh air to the online entertainment segment in a completely new, unique way. The Joker lives up to its name and does things a little differently - in its own way. In practice, this means a unique e-gaming gamble peer-to-peer game where the Joker will be implemented as a currency.

At a later stage, the Joker aims to conquer the world of live chat and create its own live chat platform in a completely original way. This is all under one platform called the Joker app. Joker app means a revolution in using and understanding cryptocurrencies. The Joker app will unify all lifestyle-related segments within one application and offer users the opportunity to draw various discounts and bonuses according to the volume of coins held.

Project features

- ❖ Joker coin integration with an adult platform where people can make untraceable transactions.
- ❖ Joker app with all-in-one place.
- ❖ The project aims to be the first social network connected to real services based on cryptocurrency JOKER.
- ❖ Lottery games
- ❖ Joker webcam paying service
- ❖ Joker's NFT marketplace.
- ❖ Joker merch.

Features

❖ The fee of 7% when buying will be swapped into BNB and allocated for the lottery pool for lottery games.

❖ Investors can accumulate more Joker tokens by just holding as the smart contract automatically distributes 5% of every transaction tax amongst holders when selling tokens.

❖ The liquidity fee of 3% when buying and selling, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity. This is a key element for decentralized exchanges like Pancakeswap.

❖ The sustainability fee of 2% marketing when buying and selling is what allows Joker token to hold the aforementioned promise. Tokens will be swapped into BNB and will be sent to a marketing wallet per transaction. This way, Joker token will have enough funds to promote the coin and spend for future development without selling tokens as the traditional way.

❖ The fee of 2% when selling is getting sent tokens to the burn wallet, that token burn happens in every transaction.

Tokenomics

12% fee when buying

❖ 7% of trade goes to the lottery pool.
❖ 3% of trade goes to the liquidity pool.
❖ 2% of trade goes to marketing.

12% fee when selling

❖ 5% of trade goes to holders' pockets in Joker tokens.
❖ 3% of trade goes to the liquidity pool.
❖ 2% of trade goes to marketing.
❖ 2% of trade goes to the burn wallet**.**

Roadmap

The team

THE TEAM

DEPLOYMENT

Alessandro
The visionary and also the mediator of the Joker's will. A cryptocurrencies enthusiast since 2015. He entered the IT world as a designer and a visionary of lottery and gambling games.

Tom
Young baron of crypto. He takes care of designing functions and features in our contract and later will help with designing the dapps and implementing them to our platforms. Also core member of marketing team and social networks freak.

Robert
One the first persons involved in development of JokerBank. His focus is mostly on lottery and card games, but dont be scared, he is not a gambler, but only developer.

David
The engineer, CEO and the commander. Former officer in the army who got lured by IT projects and has not escaped since.

DEVELOPMENT

Pavel
guru and maniac in the binary world, programmer

Ales
genius of game models and combinations, programmer

John
input / output superhero, communication, servers, multiplayer, programmer

Johny
data specialist and conceptual engineer, database and API, star of nervous system

Paul II
visual magician and uncompromising dramatic artist

Martin
algorithm madman and kernel programmer

Target market and the concept

- ❖ Anyone who's interested in the Crypto space with long-term investment plans.
- ❖ Anyone who's ready to earn a passive income in Joker tokens by holding tokens.
- ❖ Anyone who's interested in trading tokens.
- ❖ Anyone who's interested in making untraceable transactions using Joker token when the project integrates payment method with such platforms.
- ❖ Anyone who's interested in taking part with the Joker app functionalities.
- ❖ Anyone who's interested in having real life as a game experience and gets rewarded.
- ❖ Anyone who's interested in playing Joker lottery games and win rewards.
- ❖ Anyone who's interested in taking part with the Joker webcam platform where the users can make payments using Joker tokens.
- ❖ Anyone who's interested in collecting NFTs or trading NFTs.
- ❖ Anyone who's interested in taking part in the future joker activities and ideas.
- ❖ Anyone who's interested in making financial transactions with any other party using Joker token as the currency.

Core concept

Adult platform transaction

JOKER is the first cryptocurrency adult coin, that will let users enjoy favorite stuff without any traceable history. All transactions associated with girls, Cannabis, gambling, tobacco, alcohol, nightlife, and many other products and activities will be safe from judgment by others. Joker will provide transactions with approved partners, that do business with industries mentioned above with minimal transaction fees without any traceable history.

Joker app

JOKER unites everything under one JOKERAPP, where users can find barbers, tattoo salons, massages, alcohol groceries, gun shop, play online lottery games, watch webcams, bet on favorite sport plus much more, and get for using every single piece of JOKERAPP experiences, ranks and achievements so they can compete with their friends.

Jokerapp will work as a middleman for transactions between companies and consumers. By companies, mean partners that accept Jokercoin for their goods or services. They can do so only if they are producers or service providers from segments that we like to call an "adult lifestyle". Consumers then represent someone from Jokercoin´s community using Jokerapp and its services.

What will make this app unique?

Jokerapp will be so special because of its concept which is one of a kind on the whole market. It connects the demand of companies for promotion, bigger brand awareness, etc. with the demand of consumers for shopping their favorite goods and services even cheaper than the retail prices. Jokerapp will take only negligible fees for mediating these favorable conditions. That will create a triumph situation where all companies, Jokerapp, and consumers are fully satisfied with the outcome.

How does the concept work?

The core of this concept lies in the perfect combination of demand from both companies and consumers. Companies primarily want to gain new customers and increase their turnover and consumers want to buy their favorite goods and services cheaper. Companies will be able to become part of the Jokerapp and utilize its benefits for free! However, they will have to provide something else and that is sales for Jokerapp customers. Consumers will therefore be able to purchase goods and services cheaper than the original price based on how many coins they own! Consumers will simply connect their wallets with Jokerapp and can utilize their benefits based on the amount of Jokercoins held in their accounts. Bigger the amount, the bigger the sales and benefits across the whole Jokerapp. Even though Jokerapp will be mostly about Jokercoins, you will still have the right to pay with any other payment method. Jokerapp users will also be allowed to participate in giveaways, contests, and airdrops. You can apply the same rule here, the bigger the amount of the coins held in your wallet, the bigger the chance to win.

Lottery games

❖ **Jokerbank**

JOKERs first game. Like other upcoming games, the game is designed to keep money in its community, so there is no playing against casino algorithms. **There are NO FEES on buy-ins or withdrawals. The ONLY FEE there is 1% from EVERY WIN.**

0.4% of the fee goes to keep Joker servers running and to maintain every single game.

0.6% of the fee goes to **JACKPOTS,** which are also one of a kind. Jackpots are redistributed to players based on some conditions listed in every game.

❖ **Jokerplate**

It will be also based on cards and the peer-to-peer system where individual players bets against each other.

❖ **E-betting and more games**

Joker NFT market

JOKER is a huge universe where is a big space for NFT. For the real players, the Joker team will offer NFTs which will make their JOKERAPP profile look unique and astonishing. That means users will be able to buy and sell unique frames, profile pics (if you don't want to upload your real profile pic), and theme pics as NFT on JOKERs NFT market.

Another NFTs will appear in JOKERs games. Players will be able to buy special skins for cards, dice, and many more. There will be also space for special animations which will be shown to other players when you for example win or lose the game. Another important thing to keep in mind is that every deployed NFT will have its original appearance which comes with rarity or serial number. The value of each NFT will be defined by the market but obviously higher the rarity, the higher the demand.

Joker cam

One goal of JOKER is to conquer the world of webcams. Usually, on webcam servers, users are forced to buy tokens to be able to pay girls, but those tokens don't have any other usage. With JOKERs in your JOKERAPP, you will be able to simply pay to girls with JOKERs you own on your wallet. No need to buy another token. With JOKERs in your JOKERAPP, people will be able to pay for all of their favorite stuff in few clicks with zero transfers outside the app.

The lottery pool

7% of each transaction when buying gets sent in tokens to a separate pool allocating to use them for lottery games. This way, the lottery pool will have sufficient tokens to be distributed among the users and will be able to keep the platform in balance since tokens will be circulated in and out.

The token reward system

5% of each transaction when selling gets sent amongst all holders in tokens. The holders will be eligible to receive tokens, whenever a transaction occurs, and rewards are proportional to how many tokens each individual holds.

Sustainable mechanism

The liquidity fee of 3% when buying and selling, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

The **fee of 2% marketing** when buying and selling is what allows Joker token to promote the token and use funds to further the development of the platform. The tokens will be sent to the marketing wallet.

Joker token has the burn strategy that benefits and rewards those who invest long-term. It charges a 2% fee when selling from the holders. This feature slowly reduces supply making Joker token prices more and more valuable.

Potential to grow with score points

1.	Project efficiency	10/10
2.	Project uniqueness	10/10
3	Information quality	10/10
4	Service quality	9/10
5	System quality	10/10
6	Impact on the community	10/10
7	Impact on the business	10/10
8	Preparing for the future	10/10
Total Points		**9.88/10**

Contract details

Token contract details for 11th September 2021

Contract name	JokerCoin
Contract address	0xed554ce4d15d83858f57e9996296fcd742e76e24
Token supply	1,000,000,000
Token ticker	JOKER
Decimals	9
Token holders	14
Transaction count	89
Marketing wallet address	0x9df248a4a16572c84fce06bcc5de75cb50f9cb60
Contract deployer address	0x6A967a86436FF75bfd943b1C546B6C26582DE9A8
Contract's current owner address	0x6a967a86436ff75bfd943b1c546b6c26582de9a8

Token distribution

Tokens are distributed as follows:
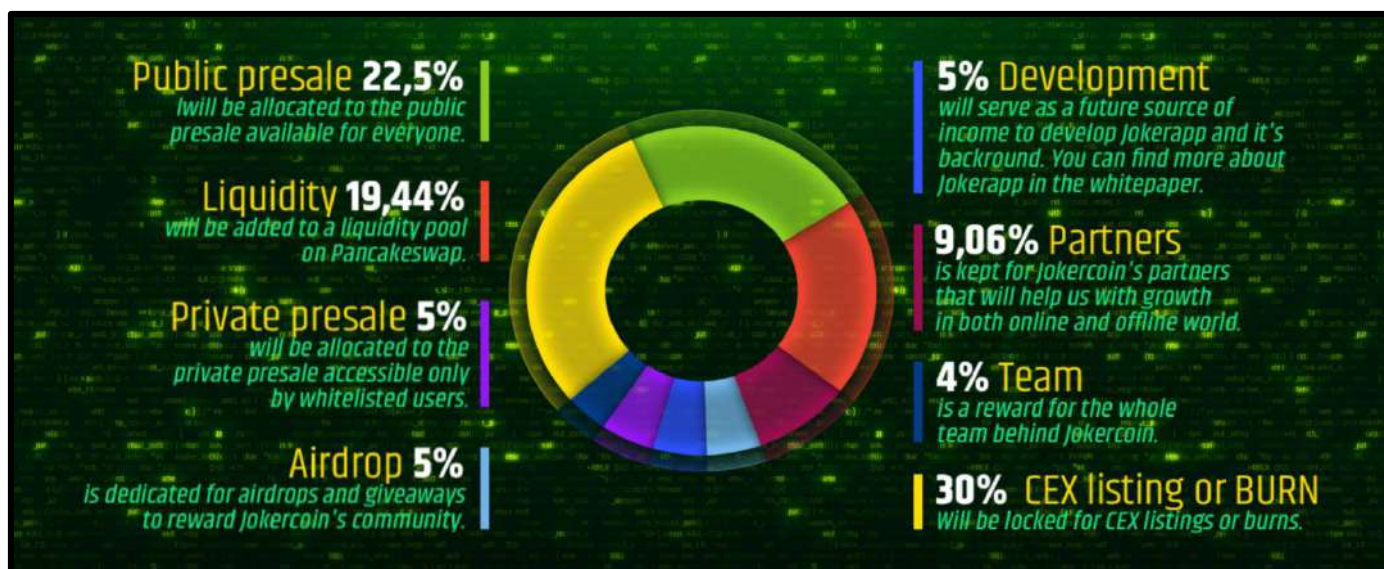
Contract code function details

No	Category	Item	Result
1	Coding conventions	BRC20 Token standards	pass

			compile errors	pass
			Compiler version security	pass
			visibility specifiers	pass
			Gas consumption	low issue
			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	high issue
			Business logics	pass
			Business implementations	low issue
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		
└	Function Name	Visibility	Mutability	Modifiers
IBEP20	**Interface**			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
SafeMath	**Library**			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
Context	**Implementation**			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Address	**Library**			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	

16

∟	functionCall	Internal 🔒	🛑	
∟	functionCallWithValue	Internal 🔒	🛑	
∟	functionCallWithValue	Internal 🔒	🛑	
∟	_functionCallWithValue	Private 🔐	🛑	
Ownable	**Implementation**	**Context**		
∟		Public ❗	🛑	NO❗
∟	owner	Public ❗		NO❗
∟	renounceOwnership	Public ❗	🛑	onlyOwner
∟	transferOwnership	Public ❗	🛑	onlyOwner
∟	geUnlockTime	Public ❗		NO❗
∟	lock	Public ❗	🛑	onlyOwner
∟	unlock	Public ❗	🛑	NO❗
IPancakeFactory	**Interface**			
∟	feeTo	External ❗		NO❗
∟	feeToSetter	External ❗		NO❗
∟	getPair	External ❗		NO❗
∟	allPairs	External ❗		NO❗
∟	allPairsLength	External ❗		NO❗
∟	createPair	External ❗	🛑	NO❗
∟	setFeeTo	External ❗	🛑	NO❗
∟	setFeeToSetter	External ❗	🛑	NO❗
IPancakePair	**Interface**			
∟	name	External ❗		NO❗
∟	symbol	External ❗		NO❗

└	decimals	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🔴	NO❗
└	transfer	External ❗	🔴	NO❗
└	transferFrom	External ❗	🔴	NO❗
└	DOMAIN_SEPARATOR	External ❗		NO❗
└	PERMIT_TYPEHASH	External ❗		NO❗
└	nonces	External ❗		NO❗
└	permit	External ❗	🔴	NO❗
└	MINIMUM_LIQUIDITY	External ❗		NO❗
└	factory	External ❗		NO❗
└	token0	External ❗		NO❗
└	token1	External ❗		NO❗
└	getReserves	External ❗		NO❗
└	price0CumulativeLast	External ❗		NO❗
└	price1CumulativeLast	External ❗		NO❗
└	kLast	External ❗		NO❗
└	mint	External ❗	🔴	NO❗
└	burn	External ❗	🔴	NO❗
└	swap	External ❗	🔴	NO❗
└	skim	External ❗	🔴	NO❗
└	sync	External ❗	🔴	NO❗
└	initialize	External ❗	🔴	NO❗

IPancakeRouter01	Interface			
└	factory	External ❗		NO ❗
└	WETH	External ❗		NO ❗
└	addLiquidity	External ❗	🛑	NO ❗
└	addLiquidityETH	External ❗	💵	NO ❗
└	removeLiquidity	External ❗	🛑	NO ❗
└	removeLiquidityETH	External ❗	🛑	NO ❗
└	removeLiquidityWithPermit	External ❗	🛑	NO ❗
└	removeLiquidityETHWithPermit	External ❗	🛑	NO ❗
└	swapExactTokensForTokens	External ❗	🛑	NO ❗
└	swapTokensForExactTokens	External ❗	🛑	NO ❗
└	swapExactETHForTokens	External ❗	💵	NO ❗
└	swapTokensForExactETH	External ❗	🛑	NO ❗
└	swapExactTokensForETH	External ❗	🛑	NO ❗
└	swapETHForExactTokens	External ❗	💵	NO ❗
└	quote	External ❗		NO ❗
└	getAmountOut	External ❗		NO ❗
└	getAmountIn	External ❗		NO ❗
└	getAmountsOut	External ❗		NO ❗
└	getAmountsIn	External ❗		NO ❗
IPancakeRouter02	Interface	IPancakeRouter01		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗

L	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
L	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
Utils	**Library**			
L	swapTokensForEth	Internal 🔒	🛑	
L	swapETHForTokens	Internal 🔒	🛑	
L	addLiquidity	Internal 🔒	🛑	
ReentrancyGuard	**Implementation**			
L		Public ❗	🛑	NO❗
JOKER	**Implementation**	**Context, IBEP20, Ownable, ReentrancyGuard**		
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	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	setMaxTxPercent	Public ❗	🛑	onlyOwner
L	setMinTokenNumberToSell	Public ❗	🛑	onlyOwner
L	setExcludeFromMaxTx	Public ❗	🛑	onlyOwner
L	setBuyback	Public ❗	🛑	onlyOwner
L	includeAndExcludeFromFee	Public ❗	🛑	onlyOwner
L	includeAndExcludeFromBuyFee	Public ❗	🛑	onlyOwner
L	includeAndExcludeFromSellFee	Public ❗	🛑	onlyOwner
L	setTaxFeePercent	External ❗	🛑	onlyOwner
L	setLiquidityFeePercent	External ❗	🛑	onlyOwner
L	setLotteryFeePercent	External ❗	🛑	onlyOwner
L	setMarketFeePercent	External ❗	🛑	onlyOwner

L	setBurnFeePercent	External ❗	🛑	onlyOwner
L	setMinBuy	External ❗	🛑	onlyOwner
L	setTimeForSniping	External ❗	🛑	onlyOwner
L	startTrading	External ❗	🛑	onlyOwner
L	setSwapAndLiquifyEnabled	Public ❗	🛑	onlyOwner
L	setReflectionFees	External ❗	🛑	onlyOwner
L	setMarketAddress	External ❗	🛑	onlyOwner
L	setPancakeRouter	External ❗	🛑	onlyOwner
L	setLotteryPools	External ❗	🛑	onlyOwner
L	setLotteryPoolsPercent	External ❗	🛑	onlyOwner
L		External ❗	💵	NO❗
L	totalFeePerTx	Internal 🔒		
L	_reflectFee	Private 🔐	🛑	
L	_getRate	Private 🔐		
L	_getCurrentSupply	Private 🔐		
L	_takeBothPoolFee	Internal 🔒	🛑	
L	_takeMarketFee	Internal 🔒	🛑	
L	_takeBurnFee	Internal 🔒	🛑	
L	removeAllFee	Private 🔐	🛑	
L	removeBuyFee	Private 🔐	🛑	
L	removeSellFee	Private 🔐	🛑	
L	restoreAllFee	Private 🔐	🛑	
L	isExcludedFromFee	Public ❗		NO❗
L	_approve	Private 🔐	🛑	
L	getContractBalance	Public ❗		NO❗
L	getCurrentPrice	Public ❗		NO❗

L	_addSniperInList	External ❗	🛑	onlyOwner
L	_removeSniperFromList	External ❗	🛑	onlyOwner
L	_transfer	Private 🔐	🛑	
L	_tokenTransfer	Private 🔐	🛑	
L	_transferStandard	Private 🔐	🛑	
L	_transferToExcluded	Private 🔐	🛑	
L	_transferFromExcluded	Private 🔐	🛑	
L	_transferBothExcluded	Private 🔐	🛑	
L	checkLottery	Internal 🔒	🛑	
L	distributeLottery	Internal 🔒	🛑	
L	swapAndLiquify	Private 🔐	🛑	

Legend

Symbol	Meaning
🛑	**Function can modify state**
💵	**Function is payable**

Inheritance Hierarchy

Security issue checking status

❖ **High severity issues**

23
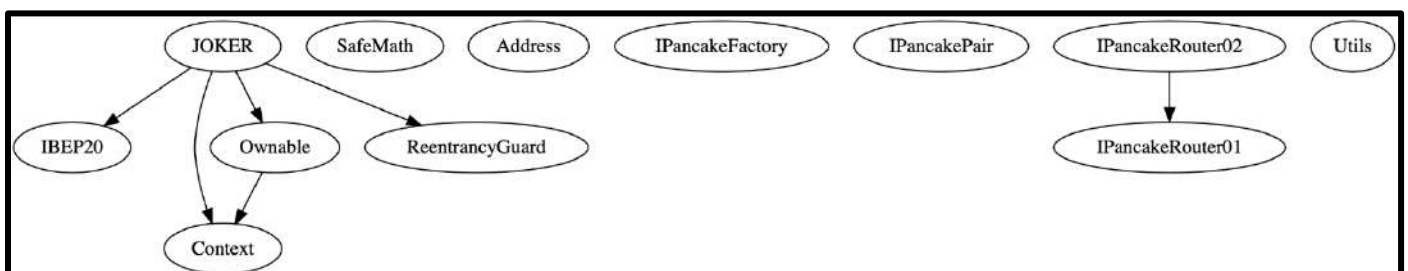
- **The owner can change the Snipping time**

The owner can add sniping time so if someone tries to sell tokens within less than launch time+ snipping time they will blacklist from the contract currently owner can change this any time without any limitation.

Recommendation

1. Remove this function and add necessary snipping time in the contract (hard code).
2. Make this function to call only one time.

```
ftrace | funcSig
function setTimeForSniping(uint256 _time↑) external onlyOwner {
    antiSnipingTime = _time↑;
}
```

- **The owner can change the max transaction amount without any limitation.**

Recommendation

Add validation to set minimum transaction amount.

```
ftrace | funcSig
function setMaxTxPercent(uint256 maxTxAmount↑) public onlyOwner {
    _maxTxAmount = _tTotal.mul(maxTxAmount↑).div(10000);
}
```

- **The owner can change all fees without any limit.**

Recommendation

Add validation to a maximum fee.

```
ftrace | funcSig
function setTaxFeePercent(uint256 taxFee↑) external onlyOwner {
    _taxFee = taxFee↑;
}

ftrace | funcSig
function setLiquidityFeePercent(uint256 liquidityFee↑) external onlyOwner {
    _liquidityFee = liquidityFee↑;
}

ftrace | funcSig
function setLotteryFeePercent(uint256 lotteryFee↑) external onlyOwner {
    _lotteryFee = lotteryFee↑;
}

ftrace | funcSig
function setMarketFeePercent(uint256 marketFee↑) external onlyOwner {
    _marketFee = marketFee↑;
}

ftrace | funcSig
function setBurnFeePercent(uint256 burnFee↑) external onlyOwner {
    _burnFee = burnFee↑;
}
```

❖ **Medium severity issues**

- **No medium severity issues found.**

❖ **Low severity issues**

- In the includeInReward,getCurrentSupply, and removeSniperList functions, if they use a long wallet list there can be an OUT_OF_GAS issue, better to use a small array list at once.

```
ftrace | funcSig
function includeInReward(address account↑) external onlyOwner {
    require(_isExcluded[account↑], "Account is already excluded");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account↑) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _rOwned[account↑] = _tOwned[account↑].mul(_getRate());
            _tOwned[account↑] = 0;
            _isExcluded[account↑] = false;
            _excluded.pop();
            break;
        }
    }
}
```

```
ftrace | funcSig
function _getCurrentSupply() private view returns (uint256, uint256) {
    uint256 rSupply = _rTotal;
    uint256 tSupply = _tTotal;
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (
            _rOwned[_excluded[i]] > rSupply ||
            _tOwned[_excluded[i]] > tSupply
        ) return (_rTotal, _tTotal);
        rSupply = rSupply.sub(_rOwned[_excluded[i]]);
        tSupply = tSupply.sub(_tOwned[_excluded[i]]);
    }
    if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
    return (rSupply, tSupply);
}
```

```
function _removeSniperFromList(address account↑) external onlyOwner {
    require(_isSniper[account↑], "Account is not blacklisted");
    for (uint256 i = 0; i < _confirmedSnipers.length; i++) {
        if (_confirmedSnipers[i] == account↑) {
            _confirmedSnipers[i] = _confirmedSnipers[
                _confirmedSnipers.length - 1
            ];
            _isSniper[account↑] = false;
            _confirmedSnipers.pop();
            break;
        }
    }
}
```

❖ **The addLiquidity function calls the uniswapV2Router.addLiquidityETH function with the address specified as owner() for acquiring the generated LP tokens from the Joker Token-BNB pool. As a result, over time the _owner address will accumulate a significant portion of LP tokens. If the _owner is an EOA(Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.**

```
ftrace | funcSig
function addLiquidity(
    address routerAddress↑,
    address owner↑,
    uint256 tokenAmount↑,
    uint256 ethAmount↑
) internal {
    IPancakeRouter02 pancakeRouter = IPancakeRouter02(routerAddress↑);

    // add the liquidity
    pancakeRouter.addLiquidityETH{value: ethAmount↑}(
        address(this),
        tokenAmount↑,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        owner↑,
        block.timestamp + 360
    );
}
```

Recommendation

We advise the address of the uniswapV2Router.addLiquidityETH function call to be replaced by the contract itself.

Owner privileges

❖ The owner can renounce and transfer ownership.

```
ftrace | funcSig
function renounceOwnership() public virtual onlyOwner {
    emit OwnershipTransferred(_owner, address(0));
    _owner = payable(address(0));
}


ftrace | funcSig
function transferOwnership(address payable newOwner⬆)
    public
    virtual
    onlyOwner
{

    require(
        newOwner⬆ != address(0),
        "Ownable: new owner is the zero address"
    );
    emit OwnershipTransferred(_owner, newOwner⬆);
    _owner = newOwner⬆;
}
```

❖ The owner can include and exclude wallets from rewards.

```
ftrace | funcSig
function excludeFromReward(address account↑) public onlyOwner {
    require(!_isExcluded[account↑], "Account is already excluded");
    if (_rOwned[account↑] > 0) {
        _tOwned[account↑] = tokenFromReflection(_rOwned[account↑]);
    }
    _isExcluded[account↑] = true;
    _excluded.push(account↑);
}

ftrace | funcSig
function includeInReward(address account↑) external onlyOwner {
    require(_isExcluded[account↑], "Account is already excluded");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account↑) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _rOwned[account↑] = _tOwned[account↑].mul(_getRate());
            _tOwned[account↑] = 0;
            _isExcluded[account↑] = false;
            _excluded.pop();
            break;
        }
    }
}
```

❖ The owner can change max transaction percentage.

```
ftrace | funcSig
function setMaxTxPercent(uint256 maxTxAmount↑) public onlyOwner {
    _maxTxAmount = _tTotal.mul(maxTxAmount↑).div(10000);
}
```

❖ The owner can set a minimum number of tokens to sell to add buyback.

```
ftrace | funcSig
function setMinTokenNumberToSell(uint256 _amount↑) public onlyOwner {
    minTokenNumberToSell = _amount↑;
}
```

❖ The owner can exclude wallets from max transaction amount.

```
ftrace | funcSig
function setExcludeFromMaxTx(address _address↑, bool value↑)
    public
    onlyOwner
{
    _isExcludedFromMaxTx[_address↑] = value↑;
}
```

❖ The owner can set buyback settings.

```
ftrace | funcSig
function setBuyback(
    uint256 _upperAmount↑,
    uint256 _lowerAmount↑,
    bool _state↑
) public onlyOwner {
    buyBackEnabled = _state↑;
    buyBackUpperLimit = _upperAmount↑;
    buyBackLowerLimit = _lowerAmount↑;
}
```

❖ The owner can include/exclude wallets from normal, sell and buy fees.

```
ftrace | funcSig
function includeAndExcludeFromFee(address account, bool _state)
    public
    onlyOwner
{
    _isExcludedFromFee[account] = _state;
}

ftrace | funcSig
function includeAndExcludeFromBuyFee(address account, bool _state)
    public
    onlyOwner
{
    _isExcludedFromBuyFee[account] = _state;
}

ftrace | funcSig
function includeAndExcludeFromSellFee(address account, bool _state)
    public
    onlyOwner
{
    _isExcludedFromSellFee[account] = _state;
}
```

❖ The owner can blacklist wallets.

```
ftrace | funcSig
function _addSniperInList(address account) external onlyOwner {
    require(
        account != address(pancakeRouter),
        "We can not blacklist pancakeRouter"
    );
    require(!_isSniper[account], "Account is already blacklisted");
    _isSniper[account] = true;
    _confirmedSnipers.push(account);
}
```

❖ The owner can change all fees.

```solidity
ftrace | funcSig
function setTaxFeePercent(uint256 taxFee↑) external onlyOwner {
    _taxFee = taxFee↑;
}

ftrace | funcSig
function setLiquidityFeePercent(uint256 liquidityFee↑) external onlyOwner {
    _liquidityFee = liquidityFee↑;
}

ftrace | funcSig
function setLotteryFeePercent(uint256 lotteryFee↑) external onlyOwner {
    _lotteryFee = lotteryFee↑;
}

ftrace | funcSig
function setMarketFeePercent(uint256 marketFee↑) external onlyOwner {
    _marketFee = marketFee↑;
}

ftrace | funcSig
function setBurnFeePercent(uint256 burnFee↑) external onlyOwner {
    _burnFee = burnFee↑;
}
```

❖ The owner can set a minimum buy amount.

```solidity
ftrace | funcSig
function setMinBuy(uint256 _minBuy↑) external onlyOwner {
    minBuy = _minBuy↑;
}
```

❖ The owner can change the Sniping time.

```solidity
ftrace | funcSig
function setTimeForSniping(uint256 _time↑) external onlyOwner {
    antiSnipingTime = _time↑;
}
```

❖ The owner can enable trading.

```
ftrace | funcSig
function startTrading() external onlyOwner {
    _tradingOpen = true;
    _launchTime = block.timestamp;

    swapAndLiquifyEnabled = true;
    // approve contract
    _approve(address(this), address(pancakeRouter), 2**256 - 1);
}
```

❖ The owner can enable swap and liquidity add.

```
ftrace | funcSig
function setSwapAndLiquifyEnabled(bool _state↑) public onlyOwner {
    swapAndLiquifyEnabled = _state↑;
    emit SwapAndLiquifyEnabledUpdated(_state↑);
}
```

❖ The owner can enable and disable the reward fee.

```
ftrace | funcSig
function setReflectionFees(bool _state↑) external onlyOwner {
    reflectionFeesdiabled = _state↑;
}
```

❖ The owner can change the marketing and router address.

```
ftrace | funcSig
function setMarketAddress(address payable _marketAddress↑)
    external
    onlyOwner
{
    marketWallet = _marketAddress↑;
}

ftrace | funcSig
function setPancakeRouter(IPancakeRouter02 _pancakeRouter↑)
    external
    onlyOwner
{
    pancakeRouter = _pancakeRouter↑;
}
```

Audit conclusion

While conducting the audit of the Joker token smart contract, it was observed that there is nothing alarming with the code, and it only contains low severity issues and informational concerns.