RugFreeCoins Audit

Kai Floki Token

Smart Contract Security Audit

February 07, 2022

Contents

Audit details

Audited project

KaiFloki Token

Contract Address

0xE1B4A56ea06fC52cf2E3271e70571Dbaa23744ca

Client contact

KaiFloki Team

Blockchain

Binance smart chain

Project website

https://kaifloki.com

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 Kai Floki Token to perform an audit of the smart contract.

https://bscscan.com/token/0xE1B4A56ea06fC52cf2E3271e70571Dbaa23744ca

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

Kai Floki is a token built on the Binance Smart Chain that is with an innovative investment use case the main purpose of which is to seek out constant revenue sources, and heading towards building even greater Community. Each transaction, purchase incurs 12% fee, and sale incurs a 12% fee.

Features

- The **sustainability fee of 8% when buying and selling for marketing and 2% when buying and selling for dev** is what allows Kai Floki to hold the aforementioned promise. Tokens will be swapped into BNB and will be sent to the marketing wallet and dev wallet. This way, Kai Floki will have enough funds to promote the coin and spend for future development without selling tokens as the traditional way.

- The additional component included under the sustainability section is a **liquidity fee of 2% when buying and selling**, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

Tokenomics

12% tax when buying and selling

- 8% of trade goes to the marketing wallet in BNB.
- 2% of trade goes to the dev wallet in BNB.
- 2% of trade goes to the liquidity pool.

Roadmap

Phase 1

- Build a big community on Telegram
- Setup Social Media
- Smart Contract Development
- Website Launch
- Presale
- Campaigns
- Social Media Boost
- Contract Audit
- Invest Heavily on Marketing

Phase 2

- KYC Presale on Pinksale
- 24h of marketing
- Pancakeswap launch
- Influencer Promotions
- Community contests
- Coinmarketcap
- Coingecko


Phase 3
Trust Wallet Logo
Listing on Exchanges
Merchandising
High Budget Marketing
NFTs
Launching KaiSwap Beta

Target market and the concept

Target market

- Anyone who's interested in the Crypto space with long-term investment plans.
- Anyone who's ready to earn a passive income by holding tokens.
- Anyone who's interested in trading tokens.
- Anyone who's interested in collecting NFTs or trading NFTs.
- Anyone who's interested in taking part with the future plans of the Kai Floki token.
- Anyone who's interested in making financial transactions with any other party using Kai Floki as the currency.

Core concept

Sustainable mechanism

The **sustainability fee of 8% when buying and selling for marketing and 2% when buying and selling for dev** is what allows Kai Floki to promote the token and use funds to further the development of the platform. Tokens will be swapped into BNB and will be sent to a marketing wallet and dev wallet. This way, Kai Floki will have access to the funds without selling tokens as the traditional way, which will enable them to consume funds without hurting the project.

The liquidity fee of 2% when buying and selling, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

Potential to grow with score points

1.	Project efficiency	8/10
2.	Project uniqueness	8/10
3	Information quality	6/10
4	Service quality	8/10
5	System quality	8/10
6	Impact on the community	7/10
7	Impact on the business	9/10
8	Preparing for the future	6/10
Total Points		**7.5/10**

Contract details

Token contract details for 07th February 2022

Contract name	KaiFloki
Contract address	0xE1B4A56ea06fC52cf2E3271e70571Dbaa23744ca
Token supply	100,000,000
Token ticker	MIYAGI
Decimals	9
Token holders	3
Transaction count	7
Marketing address	0xe92f0e9b001fc9fa269a47b2257ba353d7fd8e4a
Team wallet address	0x54c2ac492f676993da3d2ca941aee53a39d08a01
Contract deployer address	0xceB35Aa0ffea2c33c3B745D798D93DF4381Faa0C
Contract’s current owner address	0xceb35aa0ffea2c33c3b745d798d93df4381faa0c

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**		
└	Function Name	Visibility	Mutability	Modifiers
Context	**Implementation**			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
IERC20	**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 🔒		
Address	**Library**			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	_functionCallWithValue	Private 🔐	🛑	
Ownable	**Implementation**	**Context**		
└		Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	waiveOwnership	Public ❗	🛑	onlyOwner

└	transferOwnership	Public ❗	🛑	onlyOwner
└	getTime	Public ❗		NO❗
IUniswapV2Factory	**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❗
IUniswapV2Pair	**Interface**			
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	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	burn	External ❗	🛑	NO❗
L	swap	External ❗	🛑	NO❗
L	skim	External ❗	🛑	NO❗
L	sync	External ❗	🛑	NO❗

└	initialize	External ❗	🛑	NO❗
IUniswapV2Router01	**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❗
IUniswapV2Router02	**Interface**	**IUniswapV2Router01**		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
KaiFloki	**Implementation**	**Context, IERC20, Ownable**		
└		Public ❗	🛑	NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	decimals	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗

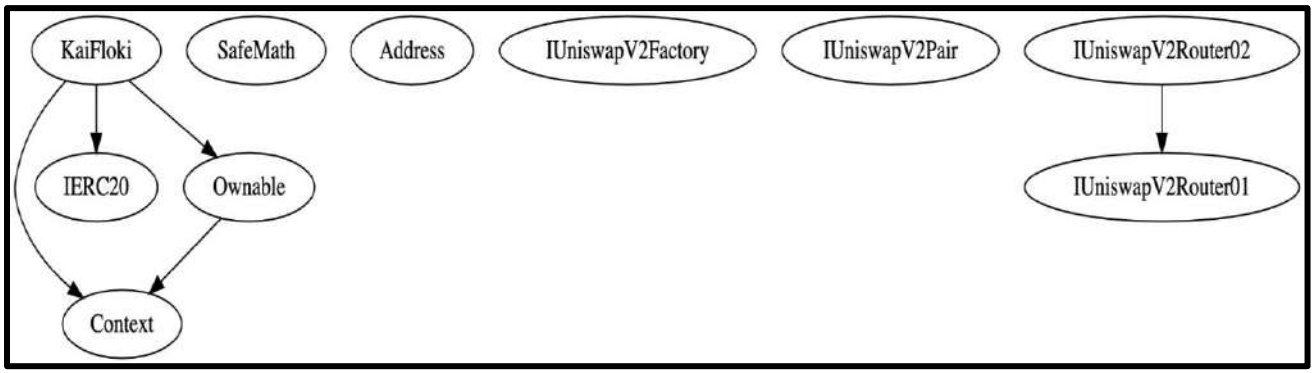
L	balanceOf	Public ❗		NO❗
L	allowance	Public ❗		NO❗
L	increaseAllowance	Public ❗	🛑	NO❗
L	decreaseAllowance	Public ❗	🛑	NO❗
L	minimumTokensBeforeSwapAmount	Public ❗		NO❗
L	approve	Public ❗	🛑	NO❗
L	_approve	Private 🔐	🛑	
L	setMarketPairStatus	Public ❗	🛑	onlyOwner
L	setIsTxLimitExempt	External ❗	🛑	onlyOwner
L	setIsExcludedFromFee	Public ❗	🛑	onlyOwner
L	activate_market	External ❗	🛑	onlyOwner
L	edit_premarket_user	External ❗	🛑	onlyOwner
L	setBuyTaxes	External ❗	🛑	onlyOwner
L	setSelTaxes	External ❗	🛑	onlyOwner
L	setDistributionSettings	External ❗	🛑	onlyOwner
L	setMaxTxAmount	External ❗	🛑	onlyOwner
L	enableDisableWalletLimit	External ❗	🛑	onlyOwner
L	setIsWalletLimitExempt	External ❗	🛑	onlyOwner
L	setWalletLimit	External ❗	🛑	onlyOwner

L	setNumTokensBeforeSwap	External ❗	🛑	onlyOwner
L	setMarketingWalletAddress	External ❗	🛑	onlyOwner
L	setTeamWalletAddress	External ❗	🛑	onlyOwner
L	setSwapAndLiquifyEnabled	Public ❗	🛑	onlyOwner
L	setSwapAndLiquifyByLimitOnly	Public ❗	🛑	onlyOwner
L	getCirculatingSupply	Public ❗		NO❗
L	transferToAddressETH	Private 🔐	🛑	
L	changeRouterVersion	Public ❗	🛑	onlyOwner
L	edit_blacklistAddress	External ❗	🛑	onlyOwner
L		External ❗	💵	NO❗
L	transfer	Public ❗	🛑	NO❗
L	transferFrom	Public ❗	🛑	NO❗
L	_transfer	Private 🔐	🛑	
L	_basicTransfer	Internal 🔒	🛑	
L	swapAndLiquify	Private 🔐	🛑	lockTheSwap
L	swapTokensForEth	Private 🔐	🛑	
L	addLiquidity	Private 🔐	🛑	
L	takeFee	Internal 🔒	🛑	

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.

- **Informational**

  - Owner can enable/disable trading anytime.

```
ftrace | funcSig
function activate_market(bool active) external onlyOwner {
    market_active = active;
}
```

  - Owner can change max transaction amount without any minimum limit.

```
ftrace | funcSig
function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner {
    _maxTxAmount = maxTxAmount;
}
```

  - Owner can change max wallet token amount without any minimum limit.

```
ftrace | funcSig
function setWalletLimit(uint256 newLimit) external onlyOwner {
    _walletMax = newLimit;
}
```

Owner privileges

- The owner can exclude wallets from transactions limit and fees.

```
ftrace | funcSig
function setIsTxLimitExempt(address holder, bool exempt)
    external
    onlyOwner
{
    isTxLimitExempt[holder] = exempt;
}

ftrace | funcSig
function setIsExcludedFromFee(address account, bool newValue)
    public
    onlyOwner
{
    isExcludedFromFee[account] = newValue;
}
```

- The owner can enable/disable trading.

```
ftrace | funcSig
function activate_market(bool active) external onlyOwner {
    market_active = active;
}
```

- The owner can add/remove authorized wallets to do transactions when trading is disabled.

```
ftrace | funcSig
function edit_premarket_user(address _address, bool active)
    external
    onlyOwner
{
    premarket_user[_address] = active;
}
```

❖ The owner can change all buy and sell fees.

```
ftrace | funcSig
function setBuyTaxes(
    uint256 newLiquidityTax,
    uint256 newMarketingTax,
    uint256 newTeamTax
) external onlyOwner {
    _buyLiquidityFee = newLiquidityTax;
    _buyMarketingFee = newMarketingTax;
    _buyTeamFee = newTeamTax;

    _totalTaxIfBuying = _buyLiquidityFee.add(_buyMarketingFee).add(
        _buyTeamFee
    );
}

ftrace | funcSig
function setSelTaxes(
    uint256 newLiquidityTax,
    uint256 newMarketingTax,
    uint256 newTeamTax
) external onlyOwner {
    _sellLiquidityFee = newLiquidityTax;
    _sellMarketingFee = newMarketingTax;
    _sellTeamFee = newTeamTax;

    _totalTaxIfSelling = _sellLiquidityFee.add(_sellMarketingFee).add(
        _sellTeamFee
    );
}
```

❖ The owner can change distribution shares.

```
ftrace | funcSig
function setDistributionSettings(
    uint256 newLiquidityShare,
    uint256 newMarketingShare,
    uint256 newTeamShare
) external onlyOwner {
    _liquidityShare = newLiquidityShare;
    _marketingShare = newMarketingShare;
    _teamShare = newTeamShare;

    _totalDistributionShares = _liquidityShare.add(_marketingShare).add(
        _teamShare
    );
}
```

- The owner can change max transaction amount.

```
ftrace | funcSig
function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner {
    _maxTxAmount = maxTxAmount;
}
```

- The owner can enable/disable checking max wallet tokens.

```
ftrace | funcSig
function enableDisableWalletLimit(bool newValue) external onlyOwner {
    checkWalletLimit = newValue;
}
```

- The owner can exclude wallets from max wallet limit.

```
ftrace | funcSig
function setIsWalletLimitExempt(address holder, bool exempt)
    external
    onlyOwner
{
    isWalletLimitExempt[holder] = exempt;
}
```

- The owner can change max wallet limit.

```
ftrace | funcSig
function setWalletLimit(uint256 newLimit) external onlyOwner {
    _walletMax = newLimit;
}
```

- The owner can change swap point.

```
ftrace | funcSig
function setNumTokensBeforeSwap(uint256 newLimit) external onlyOwner {
    minimumTokensBeforeSwap = newLimit;
}
```

- The owner can change marketing and team wallet.

```
ftrace | funcSig
function setMarketingWalletAddress(address newAddress) external onlyOwner {
    marketingWalletAddress = payable(newAddress);
}

ftrace | funcSig
function setTeamWalletAddress(address newAddress) external onlyOwner {
    teamWalletAddress = payable(newAddress);
}
```

- The Owner can enable/disable swapping.

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

- The owner can add/remove wallets from blacklist.

```
ftrace | funcSig
function edit_blacklistAddress(address account, bool value)
    external
    onlyOwner
{
    _isBlacklisted[account] = value;
}
```

❖ The owner can change router address.

```
ftrace | funcSig
function changeRouterVersion(address newRouterAddress)
    public
    onlyOwner
    returns (address newPairAddress)
{
    IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(
        newRouterAddress
    );

    newPairAddress = IUniswapV2Factory(_uniswapV2Router.factory()).getPair(
        address(this),
        _uniswapV2Router.WETH()
    );

    if (newPairAddress == address(0)) //Create If Doesnt exist
    {
        newPairAddress = IUniswapV2Factory(_uniswapV2Router.factory())
            .createPair(address(this), _uniswapV2Router.WETH());
    }

    uniswapPair = newPairAddress; //Set new pair address
    uniswapV2Router = _uniswapV2Router; //Set new router address

    isWalletLimitExempt[address(uniswapPair)] = true;
    isMarketPair[address(uniswapPair)] = true;
}
```

Audit conclusion

While conducting the audit of the Kai Floki smart contract, it was observed that there is nothing alarming with the code and it only contains informational concerns.