Cyberscope

Audit Report

Kakashi Inu

May 2022

Type	BEP20
Network	BSC
Address	0xd304358ebbad04b998f9b29c075c8df032fa381f
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	Kakashi
Compiler Version	v0.6.12+commit.27d51765
Optimization	200 runs
Licence	None
Explorer	https://bscscan.com/token/0xD304358EbbaD04b998F9b29C075c8DF032Fa381F
Symbol	Kakashi
Decimals	9
Total Supply	100,000,000
Domain	kakashiinu.xyz

Source Files

Filename	**SHA256**
contract.sol	3073d806d88464dbba607a386a53e53a188a72bed3a5a9fe019992672280fc2f

Audit Updates

Initial Audit	17th May 2022
Corrected	

Contract Analysis

Critical Medium Minor Pass

Severity	Code	Description
●	ST	Contract Owner is not able to stop or pause transactions
●	OCTD	Contract Owner is not able to transfer tokens from specific address
●	OTUT	Owner Transfer User's Tokens
●	ELFM	Contract Owner is not able to increase fees more than a reasonable percent (25%)
●	ULTW	Contract Owner is not able to increase the amount of liquidity taken by dev wallet more than a reasonable percent
●	MT	Contract Owner is not able to mint new tokens
●	BT	Contract Owner is not able to burn tokens from specific wallet
●	BC	Contract Owner is not able to blacklist wallets from selling

ST - Stop Transactions

Criticality	minor
Location	contract.sol#L903, 943, 949

Description

The contract owner has the authority to stop transactions for all users. The owner may take advantage of it by setting `_taxFee`, `_liquidityFee` or `_fundFeeAmount` to a higher value than 100.

```
function _getTValues(uint256 tAmount) private view returns (uint256, uint256,
uint256) {
        uint256 tFee = calculateTaxFee(tAmount);
        uint256 tLiquidity = calculateLiquidityFee(tAmount);
        uint256 tTransferAmount = tAmount.sub(tFee).sub(tLiquidity);
```

```
function calculateTaxFee(uint256 _amount) private view returns (uint256) {
        return _amount.mul(_taxFee).div(
            10**2
        );
    }
```

```
function calculateLiquidityFee(uint256 _amount) private view returns (uint256)
{
        return _amount.mul(_liquidityFee + _fundFee).div(
            10**2
        );
    }
```

Recommendation

The contract could embody a check for not allowing setting the `_taxFee`, `_liquidityFee` and `_fundFeeAmount` more than a reasonable amount. A suggested implementation could check that the maximum amount should be more than a fixed percentage of the total supply.

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

ELFM - Exceed Limit Fees Manipulation

Criticality	critical
Location	contract.sol#L989, 993, 997

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling the `setTaxFeePercent` , `setLiquidityFeePercent` and `setFundFeePercent` functions with a high percentage value.

```
function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
    _taxFee = taxFee;
}
```

```
function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner() {
    _liquidityFee = liquidityFee;
}
```

```
function setFundFeePercent(uint256 fundFee) external onlyOwner() {
    _fundFee = fundFee;
}
```

Recommendation

The contract could embody a check for the maximum acceptable value.

The team should carefully manage the private keys of the owner’s account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

BC - Blacklisted Contracts

Criticality	medium
Location	contract.sol#L1017

Description

The contract owner has the authority to stop contracts from transactions. The owner may take advantage of it by calling the `blacklistAddress` function.

```
require(!_isBlacklisted[from] && !_isBlacklisted[to], 'Blacklisted address');
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

Contract Diagnostics

Critical Medium Minor

Severity	Code	Description
	FSA	Fixed Swap Address
	MC	Missing Check
	L01	Public Function could be Declared External
	L02	State Variables could be Declared Constant
	L04	Conformance to Solidity Naming Conventions
	L07	Missing Events Arithmetic
	L09	Dead Code Elimination

FSA - Fixed Swap Address

Criticality	minor
Location	contract.sol#L742

Description

The swap address is assigned once in the constructor and it can not be changed. The decentralized swaps sometimes create a new swap version or abandon the current. A contract that cannot change the swap address may not be able to catch-up the upgrade.

```
IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E);
         // Create a uniswap pair for this new token
        uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
            .createPair(address(this), _uniswapV2Router.WETH());
```

Recommendation

It could be better to allow the swap address mutation in case of future swap updates.

MC - Missing Check

Criticality	medium
Location	contract.sol#L1027, 1050

Description

The contract is processing variables that have not properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues. The contract owner has the authority to set the `_liquidityFee` and `_fundFee` to zero. Hence, the `swapAndLiquify` will always be reverted.

```
function swapAndLiquify(uint256 contractTokenBalance) private lockTheSwap {

        uint256 feeTotal = _liquidityFee + _fundFee;
        uint256 fundBalance =
contractTokenBalance.mul(_fundFee).div(feeTotal);
        uint256 liquidityBalance = contractTokenBalance - fundBalance;
```

Recommendation

The contract should properly check the variables according to the required specifications.

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L431,440,446,451,459,758,762,766,770,779,784,788,793,799,804,809,813,817,826,843,876,880,884,973,1001

Description

Public functions that are never called by the contract should be declared external to save gas.

```
sendMarketToken
isExcludedFromFee
setSwapAndLiquifyEnabled
includeInFee
excludeFromFee
excludeFromReward
reflectionFromToken
deliver
totalFees
...
```

Recommendation

Use the external attribute for functions never called from the contract.

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contract.sol#L701,720,699,700,695,718

Description

Constant state variables should be declared constant to save gas.

```
numTokensSellToAddToLiquidity
_tTotal
_symbol
_name
_lpWallet
_decimals
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L503,504,521,543,884,943,949,703,706,709,710,720,721,723

Description

Solidity defines a naming convention that should be followed. Rule exceptions:

- Allow constant variable name/symbol/decimals to be lowercase.
- Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
_isBlacklisted
_fundWallet
_lpWallet
_previousFundFee
_fundFee
_liquidityFee
_taxFee
_amount
_enabled
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions

L07 - Missing Events Arithmetic

Criticality	minor
Location	contract.sol#L989,993,997

Description

Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes.

```
_fundFee = fundFee
_liquidityFee = liquidityFee
_taxFee = taxFee
```

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L357,317,327,342,352,264,291

Description

Functions that are not used in the contract, and make the code's size bigger.

```
sendValue
isContract
functionCallWithValue
functionCall
_functionCallWithValue
```

Recommendation

Remove unused functions.

Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
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	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	

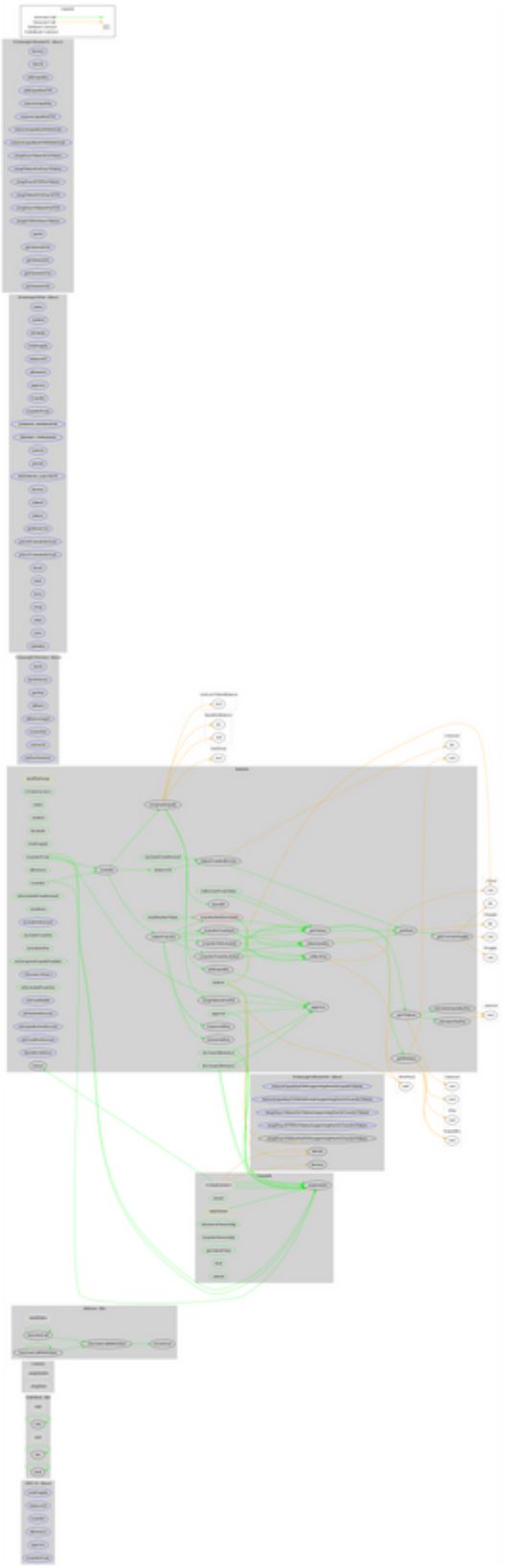
	functionCallWithValue	Internal	✓	
	_functionCallWithValue	Private	✓	
Ownable	Implementation	Context		
	<Constructor>	Internal	✓	
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	geUnlockTime	Public		-
	lock	Public	✓	onlyOwner
	unlock	Public	✓	-
IUniswapV2Factory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
IUniswapV2Pair	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-

	PERMIT_TYPEHASH	External		-
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-
	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-
	mint	External	✓	-
	burn	External	✓	-
	swap	External	✓	-
	skim	External	✓	-
	sync	External	✓	-
	initialize	External	✓	-
IUniswapV2Router01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityETH	External	✓	-
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityETHWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-
	swapExactETHForTokens	External	Payable	-
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-
	swapETHForExactTokens	External	Payable	-
	quote	External		-

	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-
IUniswapV2Router02	Interface	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
Kakashi	Implementation	Context, IERC20, Ownable		
	<Constructor>	Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	isExcludedFromReward	Public		-
	totalFees	Public		-
	deliver	Public	✓	-
	reflectionFromToken	Public		-
	tokenFromReflection	Public		-

	excludeFromReward	Public	✓	onlyOwner
	includeInReward	External	✓	onlyOwner
	_transferBothExcluded	Private	✓	
	excludeFromFee	Public	✓	onlyOwner
	includeInFee	Public	✓	onlyOwner
	setSwapAndLiquifyEnabled	Public	✓	onlyOwner
	<Receive Ether>	External	Payable	-
	_reflectFee	Private	✓	
	_getValues	Private		
	_getTValues	Private		
	_getRValues	Private		
	_getRate	Private		
	_getCurrentSupply	Private		
	_takeLiquidity	Private	✓	
	calculateTaxFee	Private		
	calculateLiquidityFee	Private		
	removeAllFee	Private	✓	
	restoreAllFee	Private	✓	
	isExcludedFromFee	Public		-
	_approve	Private	✓	
	setFundWallet	External	✓	onlyOwner
	setTaxFeePercent	External	✓	onlyOwner
	setLiquidityFeePercent	External	✓	onlyOwner
	setFundFeePercent	External	✓	onlyOwner
	sendMarketToken	Public	✓	onlyOwner
	blacklistAddress	External	✓	onlyOwner
	_transfer	Private	✓	
	swapAndLiquify	Private	✓	lockTheSwap
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	_tokenTransfer	Private	✓	
	_transferStandard	Private	✓	
	_transferToExcluded	Private	✓	
	_transferFromExcluded	Private	✓	

Contract Flow



Domain Info

Domain Name	kakashiinu.xyz
Registry Domain ID	D293114309-CNIC
Creation Date	2022-04-23T09:51:39.0Z
Updated Date	2022-04-28T09:53:58.0Z
Registry Expiry Date	2023-04-23T23:59:59.0Z
Registrar WHOIS Server	whois.gname.com
Registrar URL	https://www.gname.com/
Registrar	Gname.com Pte. Ltd.
Registrar IANA ID	1923

The domain has been created 24 days before the creation of the audit. It will expire in 11 months.

There is no public billing information, the creator is protected by the privacy settings.

Summary

There are some functions that can be abused by the owner like stopping transactions, manipulating fees and blacklisting addresses. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

Disclaimer

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment.

Cyberscope team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed.

The Cyberscope team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Cyberscope receive a payment to manipulate those results or change the awarding badge that we will be adding in our website.

Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token.

The Cyberscope team disclaims any liability for the resulting losses.

About Cyberscope

Coinscope audit and K.Y.C. service has been rebranded to Cyberscope.

Coinscope is the leading early coin listing, voting and auditing authority firm. The audit process is analyzing and monitoring many aspects of the project. That way, it gives the community a good sense of security using an informative report and a generic score.

Cyberscope and Coinscope are aiming to make crypto discoverable and efficient globally. They provides all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io