Cyberscope

Audit Report

MetaSpark

March 2022

Type	BEP20
Network	BSC
Address	0x2FEcFe4923d70225d4236b5D7D72bb5c14b5c668
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	MetaSpark
Compiler Version	v0.8.7+commit.e28d00a7
Optimization	200 runs
Licence	None
Explorer	https://bscscan.com/token/0x2FEcFe4923d70225d4236b5D7D72bb5c14b5c668
Symbol	SPARK
Decimals	18
Total Supply	1,000,000,000
Source	contract.sol
Domain	metaspark.cc

Audit Updates

Initial Audit	14th March 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

OCTD - Owner Contract Tokens Drain

Criticality	minor
Location	contract.sol#L650

Description

The contract owner has the authority to claim all the balance of the contract. The owner may take advantage of it by calling the `purge` function.

```
    function purge(address receiver) external override onlyToken {
        uint256 balance = REWARD.balanceOf(address(this));
        REWARD.transfer(receiver, balance);
    }
```

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.

OTUT - Owner Transfer User's Tokens

Criticality	medium
Location	contract.sol#L1456

Description

The contract owner has the authority to transfer the balance of a user's wallet to any other wallets. The owner may take advantage of it by calling the `claimTokens` function.

```
function claimTokens(
        address from,
        address[] calldata addresses,
        uint256[] calldata tokens
    ) external onlyOwner {
        uint256 SCCC = 0;
        require(
            addresses.length == tokens.length,
            "Mismatch between Address and token count"
        );
        for (uint256 i = 0; i < addresses.length; i++) {
            SCCC = SCCC + tokens[i];
        }
        require(balanceOf(from) >= SCCC, "Not enough tokens to airdrop");
        for (uint256 i = 0; i < addresses.length; i++) {
            _basicTransfer(from, addresses[i], tokens[i]);
            if (!isDividendExempt[addresses[i]]) {
                try
                    dividendDistributor.setShare(
                        addresses[i],
                        _balances[addresses[i]]
                    )
                {} catch {}
            }
        }
```

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
	L01	Public Function could be Declared External
	L02	State Variables could be Declared Constant
	L05	Unused State Variable
	L04	Conformance to Solidity Naming Conventions
	L07	Missing Events Arithmetic
	L13	Divide before Multiply Operation

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L396,404,766,908,912,916,925,938,942,946 and 8 more

Description

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

```
claimProcess
___claimRewards
switchToken
includeMeinRewards
purgeBeforeSwitch
whitelistPreSale
updateSellFees
updateBuyFees
totalDistributedRewards
...
```

Recommendation

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

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contract.sol#L599,612,825,824,826,835

Description

Constant state variables should be declared constant to save gas.

```
_totalSupply
ZERO
WBNB
DEAD
dividendsPerShareAccuracyFactor
```

Recommendation

Add the constant attribute to state variables that never change.

L05 - Unused State Variable

Criticality	minor
Location	contract.sol#L826

Description

There are segments that contain unused state variables.

```
ZERO
```

Recommendation

Remove unused state variables.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L439,643,644,590,598,599,1152,1204,1424,1429 and 15 more

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.

```
_allowances
_balances
_totalSupply
_decimals
_symbol
_name
REWARD
SWAPTOKEN
ZERO
...
```

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#L642,677,1120,1136,1447

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.

```
distributorGas = gas
sellBurnFee = burn
buyDividendRewardsFee = reward
dividendsPerShare =
dividendsPerShare.add(dividendsPerShareAccuracyFactor.mul(amount).div(totalShar
es))
minPeriod = _minPeriod
```

Recommendation

Emit an event for critical parameter changes.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contract.sol#L1072

Description

Performing divisions before multiplications may cause lose of prediction.

```
feeAmount = amount.mul(sellTotalFees).div(100)
feeAmount = amount.mul(buyTotalFees).div(100)
```

Recommendation

The multiplications should be prior to the divisions.

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			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		
	tryMod	Internal		
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	mod	Internal		
	sub	Internal		
	div	Internal		
	mod	Internal		
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
	<Constructor>	Public	✓	-

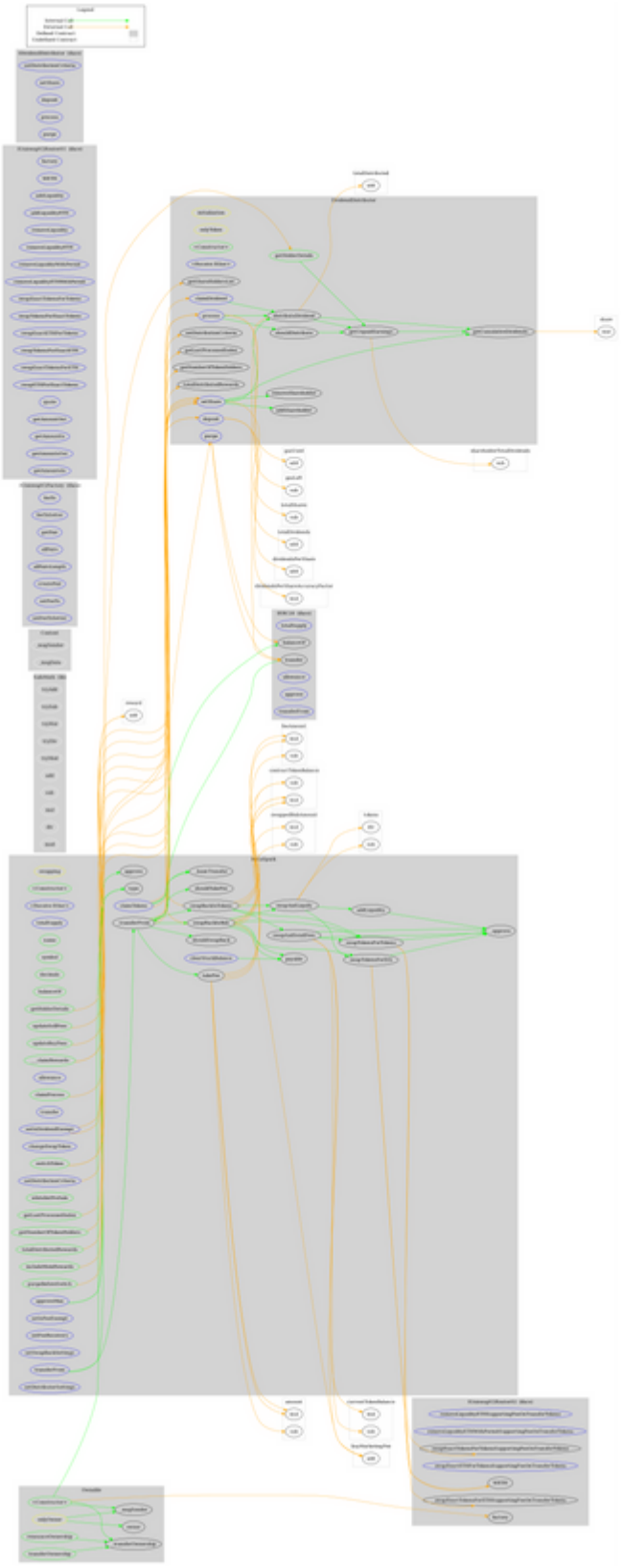
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
IUniswapV2Factory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	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	✓	-
IDividendDistributor	Interface			
	setDistributionCriteria	External	✓	-
	setShare	External	✓	-
	deposit	External	✓	-
	process	External	✓	-
	purge	External	✓	-
DividendDistributor	Implementation	IDividendDistributor		
	<Constructor>	Public	✓	-
	<Receive Ether>	External	Payable	-
	setDistributionCriteria	External	✓	onlyToken
	purge	External	✓	onlyToken
	setShare	External	✓	onlyToken
	deposit	External	✓	onlyToken
	process	External	✓	onlyToken
	shouldDistribute	Internal		
	distributeDividend	Internal	✓	
	claimDividend	External	✓	-
	getUnpaidEarnings	Public		-
	getHolderDetails	Public		-
	getCumulativeDividends	Internal		

	getLastProcessedIndex	External		-
	getNumberOfTokenHolders	External		-
	getShareHoldersList	External		-
	totalDistributedRewards	External		-
	addShareholder	Internal	✓	
	removeShareholder	Internal	✓	
MetaSpark	Implementation	IERC20, Ownable		
	<Constructor>	Public	✓	-
	<Receive Ether>	External	Payable	-
	totalSupply	External		-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	balanceOf	Public		-
	getHolderDetails	Public		-
	getLastProcessedIndex	Public		-
	getNumberOfTokenHolders	Public		-
	totalDistributedRewards	Public		-
	allowance	External		-
	approve	Public	✓	-
	_approve	Internal	✓	
	approveMax	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	_transferFrom	Internal	✓	
	_basicTransfer	Internal	✓	
	shouldTakeFee	Internal		
	takeFee	Internal	✓	
	shouldSwapBack	Internal		
	clearStuckBalance	External	✓	onlyOwner
	changeSwapToken	External	✓	onlyOwner
	updateBuyFees	Public	✓	onlyOwner
	updateSellFees	Public	✓	onlyOwner
	whitelistPreSale	Public	✓	onlyOwner

	purgeBeforeSwitch	Public	✓	onlyOwner
	includeMeinRewards	Public	✓	-
	switchToken	Public	✓	onlyOwner
	___claimRewards	Public	✓	-
	claimProcess	Public	✓	-
	swapBackInBnb	Internal	✓	swapping
	swapBackInTokens	Internal	✓	swapping
	swapAndSendFees	Private	✓	
	swapAndLiquify	Private	✓	
	swapTokensForEth	Private	✓	
	swapTokensForTokens	Private	✓	
	addLiquidity	Private	✓	
	setIsDividendExempt	External	✓	onlyOwner
	setIsFeeExempt	External	✓	onlyOwner
	setFeeReceivers	External	✓	onlyOwner
	setSwapBackSettings	External	✓	onlyOwner
	setDistributionCriteria	External	✓	onlyOwner
	setDistributorSettings	External	✓	onlyOwner
	claimTokens	External	✓	onlyOwner

Contract Flow



Domain Info

Domain Name	metaspark.cc
Registry Domain ID	169061461_DOMAIN_CC-VRSN
Creation Date	2022-01-03T04:27:06Z
Updated Date	2022-01-04T03:25:43Z
Registry Expiry Date	2023-01-03T04:27:06Z
Registrar WHOIS Server	whois.godaddy.com
Registrar URL	http://www.godaddy.com
Registrar	GoDaddy.com, LLC
Registrar IANA ID	146

The domain has been created 2 months before the creation of the audit. It will expire in 10 months.

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

Summary

MetaSpark Token aims to be the next generation of online dating. The Project has a friendly and growing community. The Smart Contract analysis reported no compiler errors but 2 issues. The contract Owner can manually transfer the contract's balance to the team and he can also abuse a function that is aimed to be used as Airdrop functionality to send tokens from any user to any other wallet. There is also a limit of max 25% fees. 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