Cyberscope

Audit Report

Metfx

May 2022

Github https://github.com/METFXWORLD/metfx-token-contract

Commit 90b99d1802cae1e22200ca19061c16bd186b5ff4

Audited by © cyberscope

Table of Contents

Contract Review

Contract Name	MetFlix
Compiler Version	v0.8.13+commit.abaa5c0e
Optimization	200 runs
Licence	
Testing Deploy	https://testnet.bscscan.com/token/0xb91a5E2d505d11b5f2E2323d7fD7E5F1E0A24158
Symbol	MFLX
Decimals	18
Total Supply	1,000,000,000
Domain	metfx.io
Github	https://github.com/METFXWORLD/metfx-token-contract
Commit	90b99d1802cae1e22200ca19061c16bd186b5ff4

Source Files

Filename	**SHA256**
contract.sol	ebfa39246af8b598489c90192db7332dd9a0718254e137ed32c5fd0aed228fcf

Audit Updates

Initial Audit	29th 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	medium
Location	contract.sol#L1077

Description

The contract owner has the authority to stop transactions for all users excluding the owner. The owner may take advantage of it, if the limitsInEffect boolean is enabled. Then, the contract owner can disable the tradingActive boolean. Aslo, the owner can stop the transactions by setting the maxTransactionAmount or/and maxWallet to zero.

```
if(limitsInEffect){
            if (
                from != owner() &&
                to != owner() &&
                to != address(0) &&
                to != address(0xdead) &&
                !swapping
            ){
                if(!tradingActive){
                    require(_isExcludedFromFees[from] ||
_isExcludedFromFees[to], "Trading is not active.");
                }

                //when buy
                if (automatedMarketMakerPairs[from] &&
!_isExcludedMaxTransactionAmount[to]) {
                        require(amount <= maxTransactionAmount, "Buy transfer
amount exceeds the maxTransactionAmount.");
                        require(amount + balanceOf(to) <= maxWallet, "Max
wallet exceeded");
                }

                //when sell
                else if (automatedMarketMakerPairs[to] &&
!_isExcludedMaxTransactionAmount[from]) {
                        require(amount <= maxTransactionAmount, "Sell transfer
amount exceeds the maxTransactionAmount.");
                }
                else {
                    require(amount + balanceOf(to) <= maxWallet, "Max wallet
exceeded");
```

Recommendation

The contract could embody a check for not allowing setting the maxTransactionAmount and maxWallet less than a reasonable amount. A suggested implementation could check that the minimum 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	medium
Location	contract.sol#L961, 1135

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 enableTrading function.

```
function enableTrading() external onlyOwner {
        tradingActive = true;
        swapEnabled = true;
        tradingActiveBlock = block.number;
    }
```

```
if(takeFee){
            if(tradingActiveBlock + 2 >= block.number &&
(automatedMarketMakerPairs[to] || automatedMarketMakerPairs[from])){
                fees = amount * 99 / 100;
                tokensForLiquidity += fees * 33 / 99;
                tokensForTreasureChest += fees * 33 / 99;
                tokensForDevelopment += fees * 33 / 99;
            }
            //
```

Recommendation

The contract could embody a check for not allowing the owner to modify the tradingActiveBlock more than one time.

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
●	BLC	Business Logic Concern
●	CR	Code Repetition
●	L01	Public Function could be Declared External
●	L04	Conformance to Solidity Naming Conventions
●	L05	Unused State Variable
●	L06	Missing Events Access Control
●	L07	Missing Events Arithmetic
●	L09	Dead Code Elimination
●	L13	Divide before Multiply Operation

BLC - Business Logic Concern

Criticality	critical
Location	contract.sol#L402, 947

Description

The business logic seems peculiar. The implementation of the burn function decreases both, total amount of _balances and _totalSupply. Hence, the values of total amount of _balances and _totalSupply are equal.

```
function _burn(address account, uint256 amount) internal virtual {
        require(account != address(0), "ERC20: burn from the zero address");

        _beforeTokenTransfer(account, address(0), amount);

        _balances[account] = _balances[account].sub(amount, "ERC20: burn
amount exceeds balance");
        _totalSupply = _totalSupply.sub(amount);
        emit Transfer(account, address(0), amount);
    }
```

But in the code segment below, when the burn function is called, the contract adds the amount_ in the _balances. In that way, the _totalSupply and the total amount of the _balances, will not be aligned.

```
function burn(uint256 amount_) external {
        require(balanceOf(msg.sender) > 0, "Error: Your balance is zero!");

        if(balanceOf(msg.sender) > 0)
        _burn(msg.sender, amount_);

        _balances[deadAddress] += amount_;
        totalSupply -= amount_;
        burnedSupply += amount_;
```

Recommendation

The team is advised to carefully check if the implementation follows the expected business logic.

CR - Code Repetition

Criticality	minor
Location	contract.sol#L947

Description

There are code segments that are repetitive in the contract. Those segments increase the code size of the contract unnecessarily. In the segment below, the contract checks the same value twice.

```
function burn(uint256 amount_) external {
        require(balanceOf(msg.sender) > 0, "Error: Your balance is zero!");

        if(balanceOf(msg.sender) > 0)
        _burn(msg.sender, amount_);
```

Recommendation

Try to remove the `if case` from the code segment.

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L341,345,353,358,362,367,373,378,485,508,541,563,1026,1058

Description

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

```
isExcludedFromFees
setAutomatedMarketMakerPair
vault
renounceOwnership
nonces
permit
decreaseAllowance
increaseAllowance
transferFrom
...
```

Recommendation

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

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L314,317,320,323,326,329,466,523,555,592,870,871,1005,1013,1240,823,861

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.

```
_isExcludedMaxTransactionAmount
deadAddress
BuyBackTokens
_treasureChestFee
_liquidityFee
_developmentFee
treasureChestWalletUpdated
developmentWalletUpdated
WETH
...
```

Recommendation

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

L05 - Unused State Variable

Criticality	minor
Location	contract.sol#L311

Description

There are segments that contain unused state variables.

```
ERC20TOKEN_ERC1820_INTERFACE_ID
```

Recommendation

Remove unused state variables.

L06 - Missing Events Access Control

Criticality	minor
Location	contract.sol#L557

Description

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

```
_vault = vault_
```

Recommendation

Emit an event for critical parameter changes.

L07 - Missing Events Arithmetic

Criticality	minor
Location	contract.sol#L983,988,992,1005,1013

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.

```
sellDevelopmentFee = _developmentFee
buyDevelopmentFee = _developmentFee
maxWallet = newNum * (10 ** 18)
maxTransactionAmount = newNum * (10 ** 18)
swapTokensAtAmount = newAmount
```

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L78,103,113,52,438,761,785,776,745,292,301,280,288,297,267,284

Description

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

```
substractPercentage
sqrrt
quadraticPricing
percentageOfTotal
percentageAmount
bondingCurve
average
safeTransferFrom
safeIncreaseAllowance
...
```

Recommendation

Remove unused functions.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contract.sol#L1064

Description

Performing divisions before multiplications may cause lose of prediction.

```
fees = amount * buyTotalFees / 100
tokensForDevelopment += fees * 33 / 99
fees = amount * sellTotalFees / 100
tokensForTreasureChest += fees * 33 / 99
fees = amount * 99 / 100
```

Recommendation

The multiplications should be prior to the divisions.

Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	_functionCallWithValue	Private	✓	
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		
	sqrrt	Internal		
	percentageAmount	Internal		

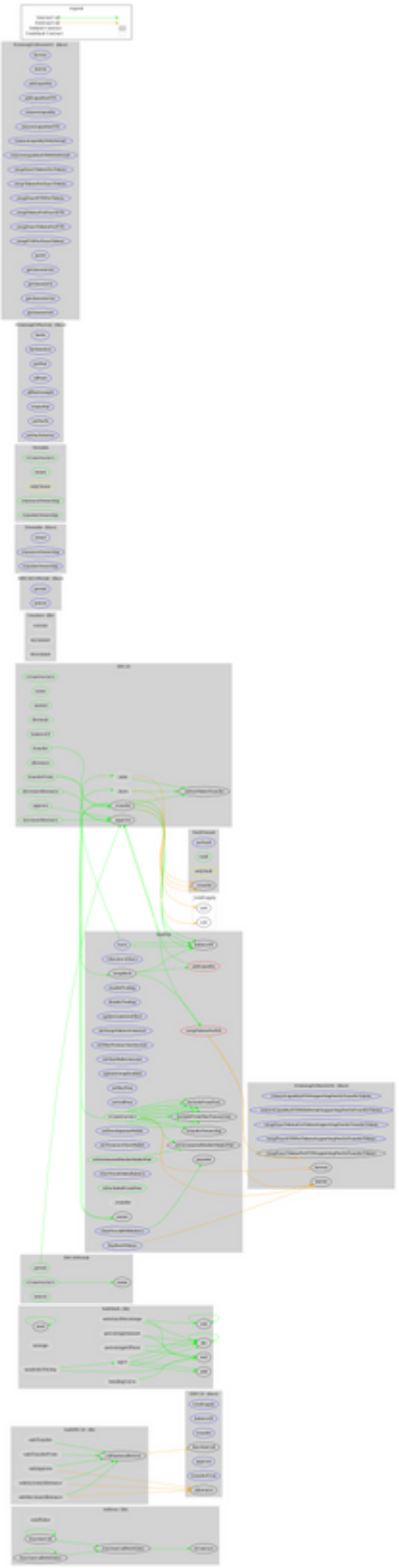
	substractPercentage	Internal		
	percentageOfTotal	Internal		
	average	Internal		
	quadraticPricing	Internal		
	bondingCurve	Internal		
ERC20	Implementation	IERC20		
	<Constructor>	Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
Counters	Library			
	current	Internal		
	increment	Internal	✓	
	decrement	Internal	✓	
IERC2612Permit	Interface			
	permit	External	✓	-
	nonces	External		-

ERC20Permit	Implementation	ERC20, IERC2612Permit		
	<Constructor>	Public	✓	-
	permit	Public	✓	-
	nonces	Public		-
IOwnable	Interface			
	owner	External		-
	renounceOwnership	External	✓	-
	transferOwnership	External	✓	-
Ownable	Implementation	IOwnable		
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
VaultOwned	Implementation	Ownable		
	setVault	External	✓	onlyOwner
	vault	Public		-
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	✓	-
SafeERC20	Library			
	safeTransfer	Internal	✓	
	safeTransferFrom	Internal	✓	
	safeApprove	Internal	✓	
	safeIncreaseAllowance	Internal	✓	

	safeDecreaseAllowance	Internal	✓	
	_callOptionalReturn	Private	✓	
MetFlix	Implementation	ERC20, VaultOwned		
	<Constructor>	Public	✓	ERC20
	<Receive Ether>	External	Payable	-
	burn	External	✓	-
	enableTrading	External	✓	onlyOwner
	disableTrading	External	✓	onlyOwner
	updateLimitsInEffect	External	✓	onlyOwner
	setSwapTokensAtAmount	External	✓	onlyOwner
	setMaxTransactionAmount	External	✓	onlyOwner
	setMaxWalletAmount	External	✓	onlyOwner
	excludeFromMaxTransaction	Public	✓	onlyOwner
	updateSwapEnabled	External	✓	onlyOwner
	setBuyFees	External	✓	onlyOwner
	setSellFees	External	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	setAutomatedMarketMakerPair	Public	✓	onlyOwner
	_setAutomatedMarketMakerPair	Private	✓	
	setDevelopmentWallet	External	✓	onlyOwner
	setTreasureChestWallet	External	✓	onlyOwner
	clearStuckBNBBalance	External	✓	onlyOwner
	clearStuckTokenBalance	External	✓	onlyOwner
	isExcludedFromFees	Public		-
	_transfer	Internal	✓	
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	swapBack	Private	✓	
	BuyBackTokens	External	✓	onlyOwner

Contract Flow



Domain Info

Domain Name	metfx.io
Registry Domain ID	d2cf44ba7edb4a1baff3af71c6ed8b25-DONUTS
Creation Date	2022-01-11T23:47:57Z
Updated Date	2022-02-24T08:15:24Z
Registry Expiry Date	2023-01-11T23:47:57Z
Registrar WHOIS Server	whois.namesilo.com
Registrar URL	http://www.namesilo.com
Registrar	NameSilo, LLC
Registrar IANA ID	1479

The domain has been created 5 months before the creation of the audit. It will expire in 8 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 and manipulating fees. The maximum fee percentage that can be set is 10%. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

The contract diverses the balances from the total supply. This issue cannot be resolved by the ownership settings.

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 provide all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io