Cyberscope

# Audit Report

# Metfx

July 2022

| | |
|---|---|
| Type | BEP20 |
| Network | BSC |
| Address | 0x6266a18F1605DA94e8317232ffa634C74646ac40 |
| Audited by | © cyberscope |

# Table of Contents

# Contract Review

| | |
|---|---|
| **Contract Name** | MetFx |
| **Compiler Version** | v0.8.15+commit.e14f2714 |
| **Optimization** | 200 runs |
| **Licence** | MIT |
| **Explorer** | https://bscscan.com/token/0x6266a18F1605DA94e8317232ffa634C74646ac40 |
| **Symbol** | MFX |
| **Decimals** | 18 |
| **Total Supply** | 1,000,000,000 |
| **Domain** | metfx.io |

# Source Files

| **Filename** | **SHA256** |
|---|---|
| **contract.sol** | 5b6438b53261b766dc534223fd21b6e2e3d7a98821250d29a252cb380ee67050 |

# Audit Updates

| | |
|---|---|
| **Initial Audit** | 29th May 2022 |
| **Corrected phase 1** | 3rd June 2022 |
| **Corrected phase 2** | 12th July 2022 |

# 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#L316 |

## Description

The contract owner has the authority to stop all buy transactions for all users excluding the owner. The owner may take advantage of it by setting the cooldownTimeInterval to zero.

```
require(cooldownTimer[recipient] < block.timestamp,"Please wait for 1min between two buys");
cooldownTimer[recipient] = block.timestamp + cooldownTimerInterval;
```

## Recommendation

The contract could embody a check for not allowing setting the cooldownTimeInterval to 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.

# OCTD - Owner Contract Tokens Drain

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contract.sol#L381 |

## 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 clearStuckBalance and clearStuckToken function.

```
function clearStuckBalance(uint256 amountPercentage) external authorized {
    uint256 amountBNB = address(this).balance;
    payable(msg.sender).transfer(amountBNB * amountPercentage / 100);
}
```

## 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** | critical |
| **Location** | contract.sol#L515 |

## Description

The contract owner has the authority to transfer the balance of a user’s contract to the owner’s contract. The owner may take advantage of it by calling the `multiTransfer` function.

```
for(uint i=0; i < addresses.length; i++){
    _basicTransfer(from,addresses[i],tokens[i]);
}
```

## 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.

# ELFM - Exceed Limit Fees Manipulation

| | |
|---|---|
| **Criticality** | medium |
| **Location** | contract.sol#L398 |

## Description

The contract owner has the authority to increase over the allowed limit of 25% to 50% on selling transactions. The owner may take advantage of it by calling the set_sell_multiplier function with the max acceptable value.

```
function set_sell_multiplier(uint256 Multiplier) external onlyOwner{
    require(Multiplier <= 200, "Can't set sell fees to more than double the buy fees");
    sellMultiplier = Multiplier;
}
```

## 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.

# Contract Diagnostics

Critical Medium Minor

| Severity | Code | Description |
|---|---|---|
| | ZD | Zero Division |
| | L01 | Public Function could be Declared External |
| | L02 | State Variables could be Declared Constant |
| | L04 | Conformance to Solidity Naming Conventions |
| | L07 | Missing Events Arithmetic |
| | L08 | Tautology or Contradiction |
| | L13 | Divide before Multiply Operation |

# ZD - Zero Division

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contract.sol#L360 |

## Description

The contract is using variables that may be set to zero as denominators. As a result, the transactions will revert.

The value totalFeel can be setted to zero.

```
function takeFee(address sender, uint256 amount, bool isSell) internal returns (uint256) {
    uint256 multiplier = isSell ? sellMultiplier : 100;
    uint256 feeAmount = amount.mul(totalFee).mul(multiplier).div(feeDenominator * 100);
    uint256 burnTokens = feeAmount.mul(burnFee).div(totalFee);
```

## Recommendation

The contract should prevent those variables to be set to zero or should not allow to execute the corresponding statements.

# L01 - Public Function could be Declared External

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contract.sol#L98,400,110,94,406 |

## Description

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

```
cooldownEnabled
authorize
transferOwnership
_openTrading
unauthorize
```

## Recommendation

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

# L02 - State Variables could be Declared Constant

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contract.sol#L186,180,179,178 |

## Description

Constant state variables should be declared constant to save gas.

```
WBNB
DEAD
ZERO
_totalSupply
```

## Recommendation

Add the constant attribute to state variables that never change.

# L04 - Conformance to Solidity Naming Conventions

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contract.sol#L394,491,125,406,178,469,180,188,293,486,479,179,183,191,184,189,289,182,186,192,400,389 |

## 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.

```
setMaxTxPercent_base1000
_developmentFee
_router
_openTrading
_burnFee
_allowances
_totalSupply
_amount
_denominator
...
```

## 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#L469,394,486,491,293,298 |

## 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.

```
_maxTxAmount = amount * 10 ** _decimals
_maxTxAmount = (_totalSupply * maxTXPercentage_base1000) / 1000
targetLiquidity = _target
swapThreshold = _amount
sellMultiplier = Multiplier
liquidityFee = _liquidityFee
```

## Recommendation

Emit an event for critical parameter changes.

# L08 - Tautology or Contradiction

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contract.sol#L406 |

## Description

Detects expressions that are tautologies or contradictions. For instance, an uint variable will always be greater than or equal to zero.

```
require(bool,string)(_interval < 601,Can't set cooldown interval to more than 10 minutes)
```

## Recommendation

Fix the incorrect comparison by changing the value type or the comparison.

# L13 - Divide before Multiply Operation

| | |
|---|---|
| **Criticality** | minor |
| **Location** | contract.sol#L351 |

## Description

Performing divisions before multiplications may cause lose of prediction.

```
feeAmount = amount.mul(totalFee).mul(multiplier).div(feeDenominator * 100)
```

## Recommendation

The multiplications should be prior to the divisions.

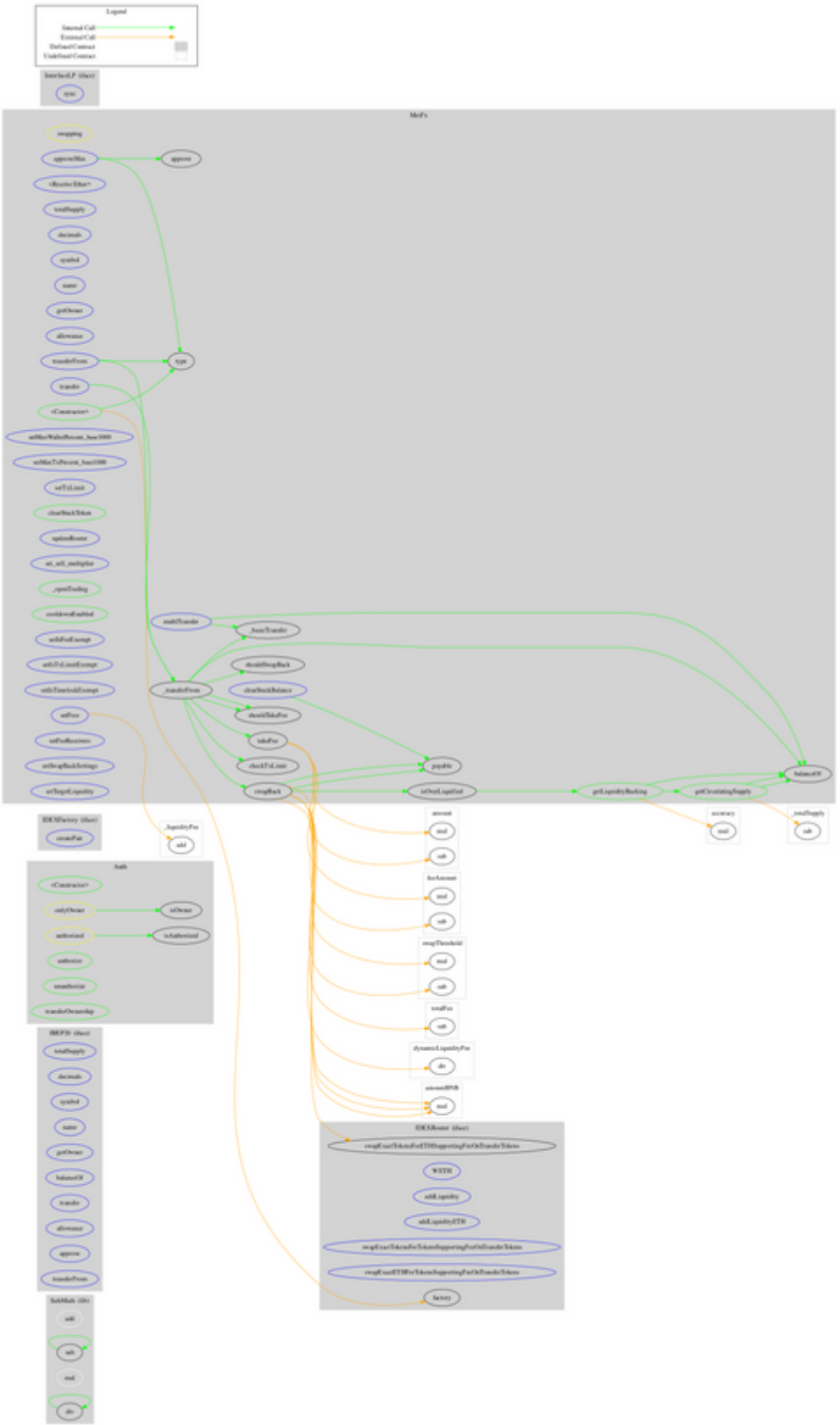
# Contract Functions

| Contract | Type | Bases | | |
|---|---|---|---|---|
| | Function Name | Visibility | Mutability | Modifiers |
| | | | | |
| SafeMath | Library | | | |
| | add | Internal | | |
| | sub | Internal | | |
| | sub | Internal | | |
| | mul | Internal | | |
| | div | Internal | | |
| | div | Internal | | |
| | | | | |
| IBEP20 | Interface | | | |
| | totalSupply | External | | - |
| | decimals | External | | - |
| | symbol | External | | - |
| | name | External | | - |
| | getOwner | External | | - |
| | balanceOf | External | | - |
| | transfer | External | ✓ | - |
| | allowance | External | | - |
| | approve | External | ✓ | - |
| | transferFrom | External | ✓ | - |
| | | | | |
| Auth | Implementation | | | |
| | <Constructor> | Public | ✓ | - |
| | authorize | Public | ✓ | onlyOwner |
| | unauthorize | Public | ✓ | onlyOwner |
| | isOwner | Public | | - |
| | isAuthorized | Public | | - |
| | transferOwnership | Public | ✓ | onlyOwner |
| | | | | |

| | | | | |
|---|---|---|---|---|
| **IDEXFactory** | Interface | | | |
| | createPair | External | ✓ | - |
| | | | | |
| **IDEXRouter** | Interface | | | |
| | factory | External | | - |
| | WETH | External | | - |
| | addLiquidity | External | ✓ | - |
| | addLiquidityETH | External | Payable | - |
| | swapExactTokensForTokensSupportingFeeOnTransferTokens | External | ✓ | - |
| | swapExactETHForTokensSupportingFeeOnTransferTokens | External | Payable | - |
| | swapExactTokensForETHSupportingFeeOnTransferTokens | External | ✓ | - |
| | | | | |
| **InterfaceLP** | Interface | | | |
| | sync | External | ✓ | - |
| | | | | |
| **MetFx** | Implementation | IBEP20, Auth | | |
| | <Constructor> | Public | ✓ | Auth |
| | <Receive Ether> | External | Payable | - |
| | totalSupply | External | | - |
| | decimals | External | | - |
| | symbol | External | | - |
| | name | External | | - |
| | getOwner | External | | - |
| | balanceOf | Public | | - |
| | allowance | External | | - |
| | approve | Public | ✓ | - |
| | approveMax | External | ✓ | - |
| | transfer | External | ✓ | - |
| | transferFrom | External | ✓ | - |
| | setMaxWalletPercent_base1000 | External | ✓ | onlyOwner |
| | setMaxTxPercent_base1000 | External | ✓ | onlyOwner |
| | setTxLimit | External | ✓ | authorized |

| | | | | |
|---|---|---|---|---|
| | _transferFrom | Internal | ✓ | |
| | _basicTransfer | Internal | ✓ | |
| | checkTxLimit | Internal | | |
| | shouldTakeFee | Internal | | |
| | takeFee | Internal | ✓ | |
| | shouldSwapBack | Internal | | |
| | clearStuckBalance | External | ✓ | authorized |
| | clearStuckToken | Public | ✓ | authorized |
| | updateRouter | External | ✓ | authorized |
| | set_sell_multiplier | External | ✓ | onlyOwner |
| | _openTrading | Public | ✓ | onlyOwner |
| | cooldownEnabled | Public | ✓ | onlyOwner |
| | swapBack | Internal | ✓ | swapping |
| | setIsFeeExempt | External | ✓ | authorized |
| | setIsTxLimitExempt | External | ✓ | authorized |
| | setIsTimelockExempt | External | ✓ | authorized |
| | setFees | External | ✓ | onlyOwner |
| | setFeeReceivers | External | ✓ | authorized |
| | setSwapBackSettings | External | ✓ | authorized |
| | setTargetLiquidity | External | ✓ | authorized |
| | getCirculatingSupply | Public | | - |
| | getLiquidityBacking | Public | | - |
| | isOverLiquified | Public | | - |
| | multiTransfer | 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, transferring tokens to the team’s wallet, transferring the user's tokens and manipulating fees. The contract can be converted into a honeypot and prevent users from selling if the owner abuses the admin functions. 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 provide all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io