Techrate1 Techrate

TechRate

AUDIT COMPANY

Smart Contract Security Audit

TechRate

November, 2021

Audit Details

Audited project

MetaLife

Deployer address

0x4a3B5D20964E8a635f6925f7128dFc1107424539

Client contacts:

MetaLife team

Blockchain

Binance Smart Chain

Project website:

https://metalife.finance/

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 below 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 TechRate and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (TechRate) owe no duty of care towards you or any other person, nor does TechRate 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 TechRate 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, TechRate hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against TechRate, 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

TechRate was commissioned by MetaLife to perform an audit of smart contracts:
https://bscscan.com/address/0xf5295EF4AA14E1DA669eeb87f6C7df54371fbFa4#code

The purpose of the audit was to achieve the following:

- **Ensure that the smart contract functions as intended.**
- **Identify potential security issues with the smart contract.**

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Contracts Details

Token contract details for 06.11.2021

Contract name	MetaLife
Contract address	0xf5295EF4AA14E1DA669eeb87f6C7df54371fbFa4
Total supply	50,000,000
Token ticker	MetaLife
Decimals	9
Token holders	1
Transactions count	1
Top 100 holders dominance	100.00%
Liquidity fee	1
Tax fee	2
Total fees	0
Uniswap V2 pair	0x36A762a180f95D798A3e11FA440d388786526735
Contract deployer address	0x4a3B5D20964E8a635f6925f7128dFc1107424539
Contract's current owner address	0x4a3B5D20964E8a635f6925f7128dFc1107424539

MetaLife Token Distribution

The top 100 holders collectively own 100.00% (50,000,000.00 Tokens) of MetaLife

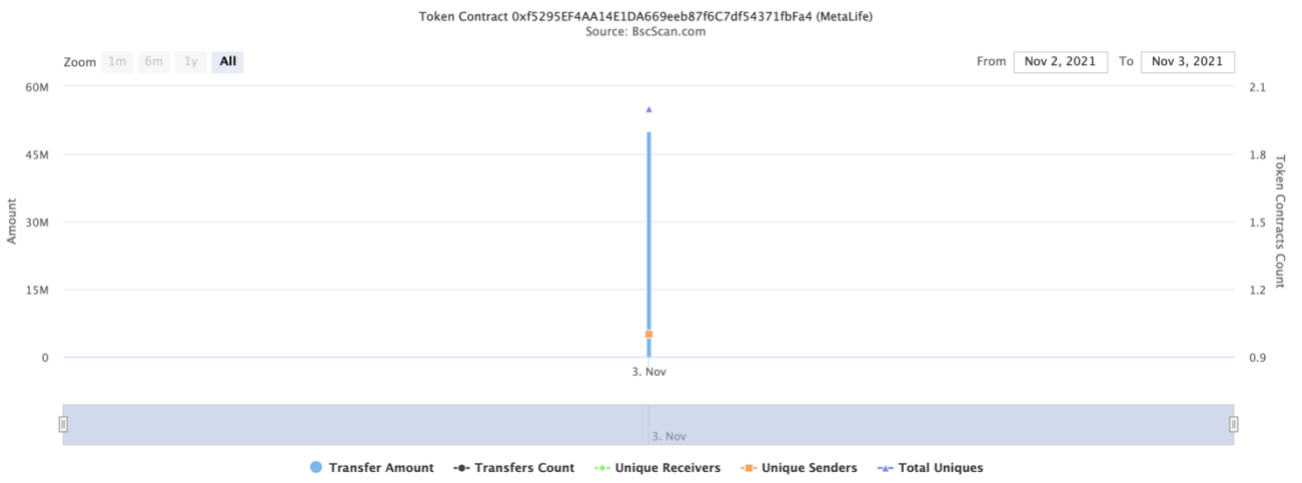
Token Total Supply: 50,000,000.00 Token | Total Token Holders: 1

(A total of 50,000,000.00 tokens held by the top 100 accounts from the total supply of 50,000,000.00 token)

MetaLife Contract Interaction Details

Time Series: Token Contract Overview

Wed 3, Nov 2021 - Wed 3, Nov 2021



MetaLife Top 10 Token Holders

Rank	Address	Quantity	Percentage
1	0x4a3b5d20964e8a635f6925f7128dfc1107424539	50,000,000	100.0000%

Contract functions details

+ Context
  - [Int] _msgSender
  - [Int] _msgData
+ [Int] IERC20
  - [Ext] totalSupply
  - [Ext] balanceOf
  - [Ext] transfer #
  - [Ext] allowance
  - [Ext] approve #
  - [Ext] transferFrom #
+ [Lib] SafeMath
  - [Int] add
  - [Int] sub
  - [Int] sub
  - [Int] mul
  - [Int] div
  - [Int] div
  - [Int] mod
  - [Int] mod
+ [Lib] Address
  - [Int] isContract
  - [Int] sendValue #
  - [Int] functionCall #
  - [Int] functionCall #
  - [Int] functionCallWithValue #
  - [Int] functionCallWithValue #
  - [Prv] _functionCallWithValue #
+ Ownable (Context)
  - [Pub] <Constructor> #
  - [Pub] owner
  - [Pub] renounceOwnership #
    - modifiers: onlyOwner
  - [Pub] transferOwnership #
    - modifiers: onlyOwner
  - [Pub] getUnlockTime
  - [Pub] getTime
  - [Pub] lock #
    - modifiers: onlyOwner
  - [Pub] unlock #

+ [Int] IUniswapV2Factory
  - [Ext] feeTo
  - [Ext] feeToSetter
  - [Ext] getPair
  - [Ext] allPairs
  - [Ext] allPairsLength
  - [Ext] createPair #
  - [Ext] setFeeTo #
  - [Ext] setFeeToSetter #

+ [Int] IUniswapV2Pair

- [Ext] name
- [Ext] symbol
- [Ext] decimals
- [Ext] totalSupply
- [Ext] balanceOf
- [Ext] allowance
- [Ext] approve #
- [Ext] transfer #
- [Ext] transferFrom #
- [Ext] DOMAIN_SEPARATOR
- [Ext] PERMIT_TYPEHASH
- [Ext] nonces
- [Ext] permit #
- [Ext] MINIMUM_LIQUIDITY
- [Ext] factory
- [Ext] token0
- [Ext] token1
- [Ext] getReserves
- [Ext] price0CumulativeLast
- [Ext] price1CumulativeLast
- [Ext] kLast
- [Ext] burn #
- [Ext] swap #
- [Ext] skim #
- [Ext] sync #
- [Ext] initialize #

+ [Int] IUniswapV2Router01
- [Ext] factory
- [Ext] WETH
- [Ext] addLiquidity #
- [Ext] addLiquidityETH ($)
- [Ext] removeLiquidity #
- [Ext] removeLiquidityETH #
- [Ext] removeLiquidityWithPermit #
- [Ext] removeLiquidityETHWithPermit #
- [Ext] swapExactTokensForTokens #
- [Ext] swapTokensForExactTokens #
- [Ext] swapExactETHForTokens ($)
- [Ext] swapTokensForExactETH #
- [Ext] swapExactTokensForETH #
- [Ext] swapETHForExactTokens ($)
- [Ext] quote
- [Ext] getAmountOut
- [Ext] getAmountIn
- [Ext] getAmountsOut
- [Ext] getAmountsIn

+ [Int] IUniswapV2Router02 (IUniswapV2Router01)
- [Ext] removeLiquidityETHSupportingFeeOnTransferTokens #
- [Ext] removeLiquidityETHWithPermitSupportingFeeOnTransferTokens #
- [Ext] swapExactTokensForTokensSupportingFeeOnTransferTokens #
- [Ext] swapExactETHForTokensSupportingFeeOnTransferTokens ($)
- [Ext] swapExactTokensForETHSupportingFeeOnTransferTokens #

+ MetaLife (Context, IERC20, Ownable)
  - [Pub] <Constructor> #
  - [Pub] name
  - [Pub] symbol
  - [Pub] decimals
  - [Pub] totalSupply
  - [Pub] balanceOf
  - [Pub] transfer #
  - [Pub] allowance
  - [Pub] approve #
  - [Pub] transferFrom #
  - [Pub] increaseAllowance #
  - [Pub] decreaseAllowance #
  - [Pub] isExcludedFromReward
  - [Pub] totalFees
  - [Pub] minimumTokensBeforeSwapAmount
  - [Pub] buyBackSellLimitAmount
  - [Pub] deliver #
  - [Pub] reflectionFromToken
  - [Pub] tokenFromReflection
  - [Pub] excludeFromReward #
    - modifiers: onlyOwner
  - [Ext] includeInReward #
    - modifiers: onlyOwner
  - [Prv] _approve #
  - [Prv] _transfer #
  - [Prv] swapTokens #
    - modifiers: lockTheSwap
  - [Prv] buyBackTokens #
    - modifiers: lockTheSwap
  - [Prv] swapTokensForEth #
  - [Prv] swapETHForTokens #
  - [Prv] addLiquidity #
  - [Prv] _tokenTransfer #
  - [Prv] _transferStandard #
  - [Prv] _transferToExcluded #
  - [Prv] _transferFromExcluded #
  - [Prv] _transferBothExcluded #
  - [Prv] _reflectFee #
  - [Prv] _getValues
  - [Prv] _getTValues
  - [Prv] _getRValues
  - [Prv] _getRate
  - [Prv] _getCurrentSupply
  - [Prv] _takeLiquidity #
  - [Prv] calculateTaxFee
  - [Prv] calculateLiquidityFee
  - [Prv] removeAllFee #
  - [Prv] restoreAllFee #
  - [Pub] isExcludedFromFee
  - [Pub] excludeFromFee #
    - modifiers: onlyOwner
  - [Pub] includeInFee #
    - modifiers: onlyOwner
  - [Prv] _getSellBnBAmount

- [Prv] _removeOldSellHistories #
- [Ext] SetBuyBackMaxTimeForHistories #
  - modifiers: onlyOwner
- [Ext] SetBuyBackDivisor #
  - modifiers: onlyOwner
- [Pub] GetBuyBackTimeInterval
- [Ext] SetBuyBackTimeInterval #
  - modifiers: onlyOwner
- [Ext] SetBuyBackRangeRate #
  - modifiers: onlyOwner
- [Pub] GetSwapMinutes
- [Ext] SetSwapMinutes #
  - modifiers: onlyOwner
- [Ext] setTaxFeePercent #
  - modifiers: onlyOwner
- [Ext] setBuyFee #
  - modifiers: onlyOwner
- [Ext] setSellFee #
  - modifiers: onlyOwner
- [Ext] setLiquidityFeePercent #
  - modifiers: onlyOwner
- [Ext] setBuyBackSellLimit #
  - modifiers: onlyOwner
- [Ext] setMaxTxAmount #
  - modifiers: onlyOwner
- [Ext] setMarketingDivisor #
  - modifiers: onlyOwner
- [Ext] setNumTokensSellToAddToBuyBack #
  - modifiers: onlyOwner
- [Ext] setMarketingAddress #
  - modifiers: onlyOwner
- [Pub] setSwapAndLiquifyEnabled #
  - modifiers: onlyOwner
- [Pub] setBuyBackEnabled #
  - modifiers: onlyOwner
- [Pub] setAutoBuyBackEnabled #
  - modifiers: onlyOwner
- [Ext] prepareForPreSale #
  - modifiers: onlyOwner
- [Ext] afterPreSale #
  - modifiers: onlyOwner
- [Prv] transferToAddressETH #
- [Pub] changeRouterVersion #
  - modifiers: onlyOwner
- [Ext] <Fallback> ($)
- [Pub] transferForeignToken #
  - modifiers: onlyOwner
- [Ext] Sweep #
  - modifiers: onlyOwner
- [Ext] setAddressFee #
  - modifiers: onlyOwner
- [Ext] setBuyAddressFee #
  - modifiers: onlyOwner
- [Ext] setSellAddressFee #
  - modifiers: onlyOwner

Issues Checking Status

	Issue description	Checking status
1.	Compiler errors.	Passed
2.	Race conditions and Reentrancy. Cross-function race conditions.	Passed
3.	Possible delays in data delivery.	Passed
4.	Oracle calls.	Passed
5.	Front running.	Passed
6.	Timestamp dependence.	Passed
7.	Integer Overflow and Underflow.	Passed
8.	DoS with Revert.	Passed
9.	DoS with block gas limit.	Low issues
10.	Methods execution permissions.	Passed
11.	Economy model of the contract.	Passed
12.	The impact of the exchange rate on the logic.	Passed
13.	Private user data leaks.	Passed
14.	Malicious Event log.	Passed
15.	Scoping and Declarations.	Passed
16.	Uninitialized storage pointers.	Passed
17.	Arithmetic accuracy.	Passed
18.	Design Logic.	Passed
19.	Cross-function race conditions.	Passed
20.	Safe Open Zeppelin contracts implementation and usage.	Passed
21.	Fallback function security.	Passed

Security Issues

High Severity Issues

No high severity issues found.

Medium Severity Issues

No medium severity issues found.

Low Severity Issues

1. Out of gas

Issue:

- The function `includeInReward()` uses the loop to find and remove addresses from the `_excluded` list. Function will be aborted with `OUT_OF_GAS` exception if there will be a long excluded addresses list.

```
function includeInReward(address account) external onlyOwner() {
    require(_isExcluded[account], "Account is not excluded");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

- The function `_getCurrentSupply` also uses the loop for evaluating total supply. It also could be aborted with `OUT_OF_GAS` exception if there will be a long excluded addresses list.

```
function _getCurrentSupply() private view returns(uint256, uint256) {
    uint256 rSupply = _rTotal;
    uint256 tSupply = _tTotal;
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal, _tTotal);
        rSupply = rSupply.sub(_rOwned[_excluded[i]]);
        tSupply = tSupply.sub(_tOwned[_excluded[i]]);
    }
    if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
    return (rSupply, tSupply);
}
```

Recommendation:

Check that the excluded array length is not too big.

Notes:

- addLiquidity function is unused.

Owner privileges (In the period when the owner is not renounced)

- Owner can lock and unlock. By the way, using these functions the owner could retake privileges even after the ownership was renounced.

```
function lock(uint256 time) public virtual onlyOwner {
    _previousOwner = _owner;
    _owner = address(0);
    _lockTime = block.timestamp + time;
    emit OwnershipTransferred(_owner, address(0));
}

function unlock() public virtual {
    require(_previousOwner == msg.sender, "You don't have permission to unlock");
    require(block.timestamp > _lockTime , "Contract is locked until 7 days");
    emit OwnershipTransferred(_owner, _previousOwner);
    _owner = _previousOwner;
}
```

- Owner can include in and exclude from reward.

```
function excludeFromReward(address account) public onlyOwner() {

    require(!_isExcluded[account], "Account is already excluded");
    if(_rOwned[account] > 0) {
        _tOwned[account] = tokenFromReflection(_rOwned[account]);
    }
    _isExcluded[account] = true;
    _excluded.push(account);
}

function includeInReward(address account) external onlyOwner() {
    require(_isExcluded[account], "Account is not excluded");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

- Owner can include in and exclude from fee.

```
function excludeFromFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = true;
}

function includeInFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = false;
}
```

- Owner can change _buyBackMaxTimeForHistories.

```
function SetBuyBackMaxTimeForHistories(uint256 newMinutes) external onlyOwner {
    _buyBackMaxTimeForHistories = newMinutes * 1 minutes;
}
```

- Owner can change buyback divisor.

```
function SetBuyBackDivisor(uint256 newDivisor) external onlyOwner {
    _buyBackDivisor = newDivisor;
}
```

- Owner can change buyback time interval and range rate.

```
function SetBuyBackTimeInterval(uint256 newMinutes) external onlyOwner {
    _buyBackTimeInterval = newMinutes * 1 minutes;
}

function SetBuyBackRangeRate(uint256 newPercent) external onlyOwner {
    require(newPercent <= 100, "The value must not be larger than 100.");
    _buyBackRangeRate = newPercent;
}
```

- Owner can can change _intervalMinutesForSwap.

```
function SetSwapMinutes(uint256 newMinutes) external onlyOwner {
    _intervalMinutesForSwap = newMinutes * 1 minutes;
}
```

- Owner can change tax, liquidity, buy and sell fees.

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

function setBuyFee(uint256 buyTaxFee, uint256 buyLiquidityFee) external onlyOwner {
    _buyTaxFee = buyTaxFee;
    _buyLiquidityFee = buyLiquidityFee;
}

function setSellFee(uint256 sellTaxFee, uint256 sellLiquidityFee) external onlyOwner {
    _sellTaxFee = sellTaxFee;
    _sellLiquidityFee = sellLiquidityFee;
}

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

- Owner can change buyBackSellLimit.

```
function setBuyBackSellLimit(uint256 buyBackSellSetLimit) external onlyOwner {
    buyBackSellLimit = buyBackSellSetLimit;
}
```

- Owner can change maximum transaction amount.

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

- Owner can change marketing divisor.

```
function setMarketingDivisor(uint256 divisor) external onlyOwner {
    marketingDivisor = divisor;
}
```

- Owner can change minimum number of tokens to add to liquidity.

```
function setNumTokensSellToAddToBuyBack(uint256 _minimumTokensBeforeSwap) external onlyOwner {
    minimumTokensBeforeSwap = _minimumTokensBeforeSwap;
}
```

- Owner can change marketing address.

```
function setMarketingAddress(address _marketingAddress) external onlyOwner {
    marketingAddress = payable(_marketingAddress);
}
```

- Owner can enable / disable swap and liquify.

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

- Owner can enable / disable buyback and auto buyback.

```
function setBuyBackEnabled(bool _enabled) public onlyOwner {
    buyBackEnabled = _enabled;
    emit BuyBackEnabledUpdated(_enabled);
}

function setAutoBuyBackEnabled(bool _enabled) public onlyOwner {
    _isAutoBuyBack = _enabled;
    emit AutoBuyBackEnabledUpdated(_enabled);
}
```

- Owner can enable before and after presale modes.

```
function prepareForPreSale() external onlyOwner {
    setSwapAndLiquifyEnabled(false);
    _taxFee = 0;
    _liquidityFee = 0;
    _maxTxAmount = 50 * 10**6 * 10**9;
}

function afterPreSale() external onlyOwner {
    setSwapAndLiquifyEnabled(true);
    _taxFee = 2;
    _liquidityFee = 10;
    _maxTxAmount = 3000000 * 10**6 * 10**9;
}
```

- Owner can withdraw BNBs.

```
function Sweep() external onlyOwner {
    uint256 balance = address(this).balance;
    payable(owner()).transfer(balance);
}
```

- **Owner can withdraw tokens.**

```
function transferForeignToken(address _token, address _to) public onlyOwner returns(bool _sent){
    require(_token != address(this), "Can't let you take all native token");
    uint256 _contractBalance = IERC20(_token).balanceOf(address(this));
    _sent = IERC20(_token).transfer(_to, _contractBalance);
}
```

- **Owner can Uniswap router address.**

```
function changeRouterVersion(address _router) public onlyOwner returns(address _pair) {
    IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(_router);

    _pair = IUniswapV2Factory(_uniswapV2Router.factory()).getPair(address(this), _uniswapV2Router.WETH());
    if(_pair == address(0)){
        // Pair doesn't exist
        _pair = IUniswapV2Factory(_uniswapV2Router.factory())
        .createPair(address(this), _uniswapV2Router.WETH());
    }
    uniswapV2Pair = _pair;

    // Set the router of the contract variables
    uniswapV2Router = _uniswapV2Router;
}
```

- **Owner can set addresses fees.**

```
function setAddressFee(address _address, bool _enable, uint256 _addressTaxFee, uint256 _addressLiquidityFee) external onlyOwner {
    _addressFees[_address].enable = _enable;
    _addressFees[_address]._taxFee = _addressTaxFee;
    _addressFees[_address]._liquidityFee = _addressLiquidityFee;
}

function setBuyAddressFee(address _address, bool _enable, uint256 _addressTaxFee, uint256 _addressLiquidityFee) external onlyOwner {
    _addressFees[_address].enable = _enable;
    _addressFees[_address]._buyTaxFee = _addressTaxFee;
    _addressFees[_address]._buyLiquidityFee = _addressLiquidityFee;
}

function setSellAddressFee(address _address, bool _enable, uint256 _addressTaxFee, uint256 _addressLiquidityFee) external onlyOwner {
    _addressFees[_address].enable = _enable;
    _addressFees[_address]._sellTaxFee = _addressTaxFee;
    _addressFees[_address]._sellLiquidityFee = _addressLiquidityFee;
}
```

Conclusion

Smart contracts contain low severity issues and owner privileges! Liquidity pair contract's security is not checked due to out of scope. 4% of the liquidity goes to the marketing address. The further transfers and operations with the funds raise are not related to this particular contract.

Liquidity locking details are NOT provided by the team.

TechRate note:

Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.