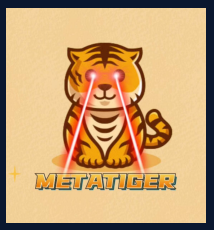
freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT



Meta Tiger

$Mtiger

25/01/2022

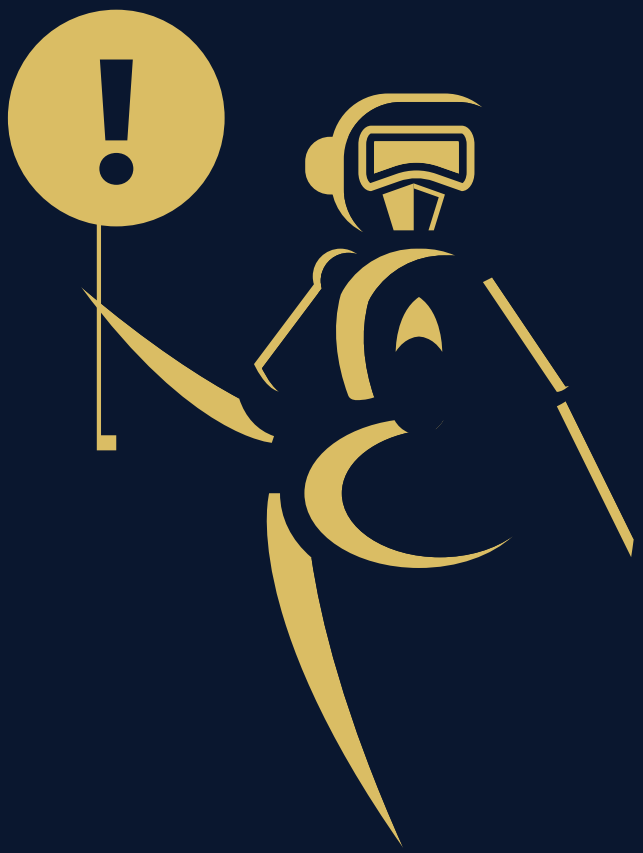
TABLE OF CONTENTS

DISCLAIMER

The information provided on this analysis document is only for general information and should not be used as a reason to invest.

FreshCoins Team will take no payment for manipulating the results of this audit.

The score and the result will stay on this project page information on our website https://freshcoins.io
FreshCoins Team does not guarantees that a project will not sell off team supply, or any other scam strategy (RUG or Honeypot etc)



INTRODUCTION

FreshCoins (Consultant) was contracted by Meta Tiger (Customer) to conduct a Smart Contract Code Review and Security Analysis.

0xfF9eEBF9E901921E5cC62D3705aA73927B877eF6

Network: Binance Smart Chain (BSC)

This report presents the findings of the security assessment of Customer's smart contract and its code review conducted on 25/01/2022

AUDIT OVERVIEW

94

Security Score

92

Static Scan
Automatic scanning for common vulnerabilities

96

ERC Scan
Automatic checks for ERC's conformance

0 High

0 Medium

0 Low

0 Optimizations

0 Informational

No.	Issue description	Checking Status
1	Compiler Errors / Warnings	Passed
2	Reentrancy and Cross-function	Passed
3	Front running	Passed
4	Timestamp dependence	Passed
5	Integer Overflow and Underflow	Passed
6	Reverted DoS	Passed
7	DoS with block gas limit	Passed
8	Methods execution permissions	Passed
9	Exchange rate impact	Passed
10	Malicious Event	Passed
11	Scoping and Declarations	Passed
12	Uninitialized storage pointers	Passed
13	Design Logic	Passed
14	Safe Zeppelin module	Passed

OWNER PRIVILEGES

Contract owner can't mint tokens after initial contract deploy.

Owner must be a contract with transparent rules for using setWalletLimit function.

Contract owner can change buy/sell taxes

```
function setBuyTaxes(uint256 newLiquidityTax, uint256 newMarketingTax, uint256 newTeamTax) external
onlyOwner() {
    _buyLiquidityFee = newLiquidityTax;
    _buyMarketingFee = newMarketingTax;
    _buyTeamFee = newTeamTax;

    _totalTaxIfBuying = _buyLiquidityFee.add(_buyMarketingFee).add(_buyTeamFee);
}

function setSellTaxes(uint256 newLiquidityTax, uint256 newMarketingTax, uint256 newTeamTax) external
onlyOwner() {
    _sellLiquidityFee = newLiquidityTax;
    _sellMarketingFee = newMarketingTax;
    _sellTeamFee = newTeamTax;

    _totalTaxIfSelling = _sellLiquidityFee.add(_sellMarketingFee).add(_sellTeamFee);
}
```

Contract owner can exclude wallet from fees

```
function setIsExcludedFromFee(address account, bool newValue) public onlyOwner {
    isExcludedFromFee[account] = newValue;
}
```

Contract owner can change swap settings

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

Contract owner can change max tx amount and wallet limit

```
function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner() {
    _maxTxAmount = maxTxAmount;
}
.
.
.
function setIsTxLimitExempt(address holder, bool exempt) external onlyOwner {
    isTxLimitExempt[holder] = exempt;
}
```

```
function setWalletLimit(uint256 newLimit) external onlyOwner {
    _walletMax  = newLimit;
}
.
.
.
function setIsWalletLimitExempt(address holder, bool exempt) external onlyOwner {
    isWalletLimitExempt[holder] = exempt;
}
```

Contract owner can renounce ownership

```
function waiveOwnership() public virtual onlyOwner {
    emit OwnershipTransferred(_owner, address(0));
    _owner = address(0);
}
```

Contract owner can transfer ownership

```
function transferOwnership(address newOwner) public virtual onlyOwner {
    require(newOwner != address(0), "Ownable: new owner is the zero address");
    emit OwnershipTransferred(_owner, newOwner);
    _owner = newOwner;
}
```

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. The risk can be prevented by temporarily locking the contract or renouncing ownership.

CONCLUSION AND ANALYSIS

Smart Contracts within the scope were manually reviewed and analyzed with static tools.

Audit report overview contains all found security vulnerabilities and other issues in the reviewed code.

Found no issue during the first review.

TOKEN DETAILS

Details

Buy fees:	9%
Sell fees:	9%
Max TX:	1,000,000,000,000
Max Sell:	N/A

Honeypot Risk

Ownership:	Owned
Blacklist:	Not detected
Modify Max TX:	Detected
Modify Max Sell:	Not detected
Disable Trading:	Not detected

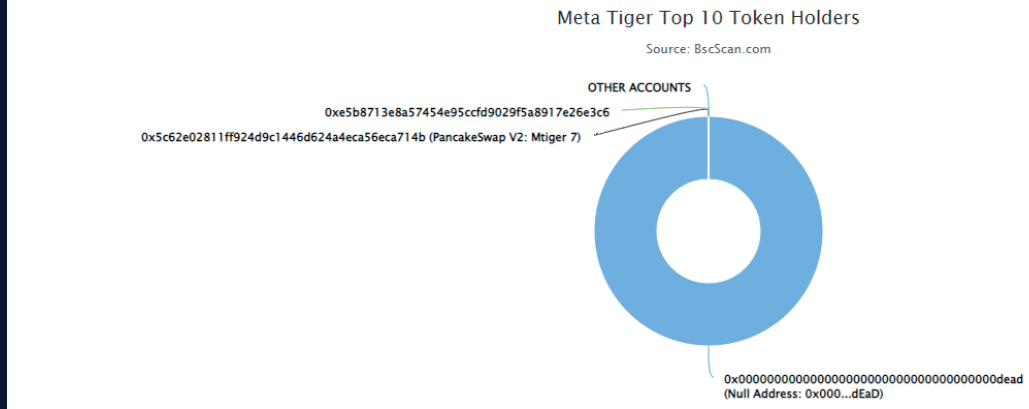
Rug Pull Risk

Liquidity:	N/A
Holders:	Clean

META TIGER COIN TOKEN DISTRIBUTION & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (999,999,960,998,533.00 Tokens) of Meta Tiger

Token Total Supply: 1,000,000,000,000,000.00 Token | Total Token Holders: 111



(A total of 999,999,960,998,533.00 tokens held by the top 10 accounts from the total supply of 1,000,000,000,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	Null Address: 0x000...dEaD	999,999,900,000,000	100.0000%
2	PancakeSwap V2: Mtiger 7	38,512,301.158589373	0.0000%
3	0xe5b8713e8a57454e95ccfd9029f5a8917e26e3c6	4,676,462.167374265	0.0000%
4	0xc2cde24df5331fd7f3f9215ff5119e7eb2a6d98a	3,569,883.709116248	0.0000%
5	0x87edc43c1da8294627acc99759b27d2fbf46c52c	3,000,000	0.0000%
6	0x139163bbda73b9be4941930d56db226b3c25a2f9	2,941,820.782752691	0.0000%
7	0x5a377ebac2cecf65b5f4591b056a0756876534ee	2,430,516.401027513	0.0000%
8	0x1583f7a3d433d87c34ce52e1f695f42f29b81b48	2,052,654.704527413	0.0000%
9	0x194f876e1b2571de246b29ff69fe3aede4fcbd90	1,967,049.757312945	0.0000%
10	0xb7f08c976d4dde0629378796a17adfac649b9f41	1,847,844.178124955	0.0000%

TECHNICAL DISCLAIMER

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. The audit can't guarantee the explicit security of the audited project / smart contract.