auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Tuzki Ceo

DATED : 9 June 23'

AUDIT SUMMARY

Project name – Tuzki Ceo

Date: 9 June, 2023

Scope of Audit- Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

Audit Status: Passed

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	3	1	0	2
Acknowledged	0	0	0	0	0
Resolved	0	0	0	0	0

USED TOOLS

Tools:

1- Manual Review:
A line by line code review has been performed by audit ace team.

2- BSC Test Network: All tests were conducted on the BSC Test network, and each test has a corresponding transaction attached to it. These tests can be found in the "Functional Tests" section of the report.

3- Slither :
The code has undergone static analysis using Slither.

Testnet version:
The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address:
https://testnet.bscscan.com/address/0xcf0324a3eb16e9ec001eb2101dda49ce018a5db7

Token Information

Token Name : Tuzki Ceo

Token Symbol: $TZK

Decimals: 9

Token Supply: 1,000,000,000,000

Token Address:
0x1054423B0b94102c56b4be48602f4DE59B3c177A

Checksum:
10e877ea4305ed058f527f38ebff0a2b4ab987d7

Owner:
0x97ECbB8B587c514f0A06fC39bbD82A0A08f93f8e
(at time of writing the audit)

Deployer:
0x97ECbB8B587c514f0A06fC39bbD82A0A08f93f8e

TOKEN OVERVIEW

Fees:

Buy Fees: 0-11%

Sell Fees: 0-11%

Transfer Fees: 0-11%

Fees Privilege: Owner

Ownership: 0x97ECbB8B587c514f0A06fC39bbD82A0A08f93f8e

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: initial distribution of tokens

including or excluding from fees

changing swap threshold

changing fees

AUDIT METHODOLOGY

The auditing process will follow a routine as special considerations by Auditace:

- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.
- Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.
- Test coverage analysis determines whether the test cases are covering the code and how much code isexercised when we run the test cases.
- Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.
- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

VULNERABILITY CHECKLIST

- Return values of low-level calls
- Private modifier
- Multiple Sends
- Using Suicide
- Gas Limitand Loops
- Address hardcoded
- Exception Disorder
- Using inline assembly
- Divide before multiply
- Missing Zero Address Validation
- Compiler version not fixed
- **Gasless Send**
- Using block.timestamp
- Re-entrancy
- Tautology or contradiction
- Timestamp Dependence
- Revert/require functions
- Use of tx.origin
- Integer overflow/underflow
- Dangerous strict equalities
- Using SHA3
- Using throw

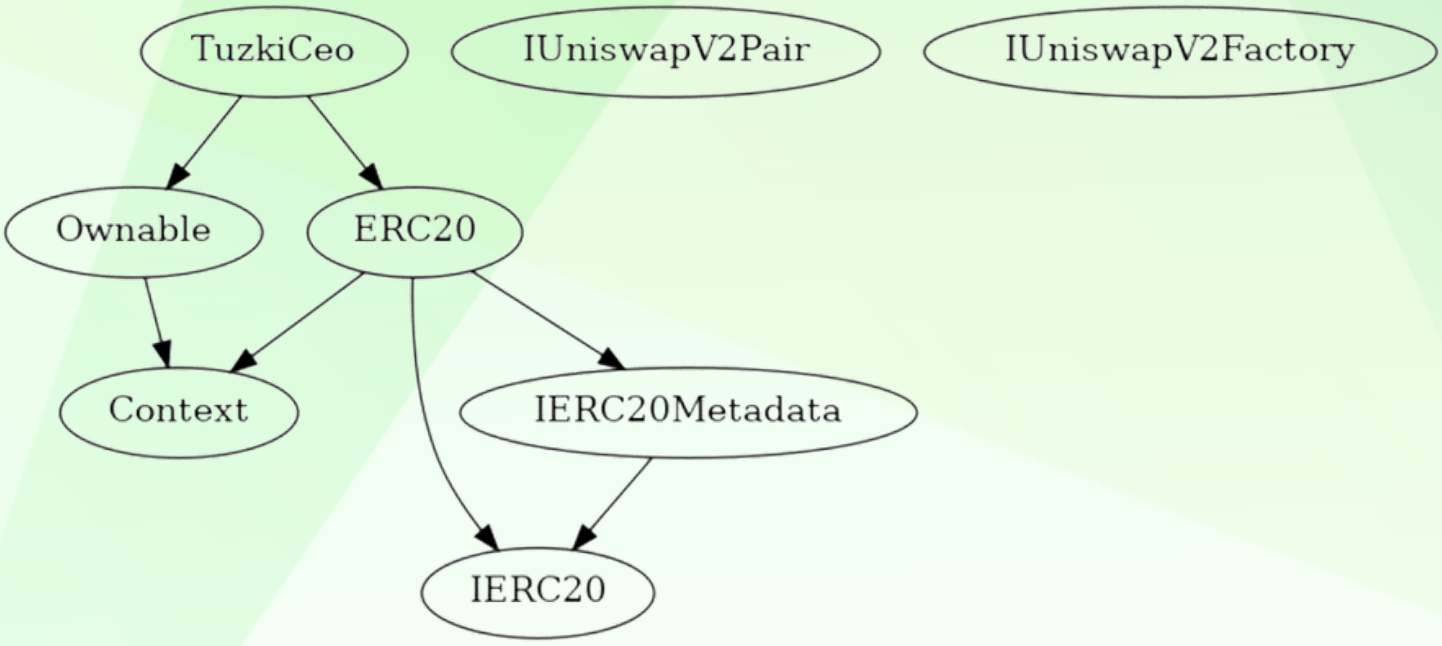
CLASSIFICATION OF RISK

Severity	Description
Critical	These vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
High-Risk	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
Medium-Risk	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
Low-Risk	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
Gas Optimization /Suggestion	A vulnerability that has an informational character but is not affecting any of the code.

Findings

Severity	Found
Critical	0
High-Risk	3
Medium-Risk	1
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- owner is able to change fees in range of 0-11% for buy/sell/transfer transactions.
- owner is not able to blacklist an arbitrary wallet
- owner is not able to set limit for buy/sell/transfer/holding amounts
- owner is not able to mint new tokens
- owner is not able to disable trades
- owner must enable trades manually

STATIC ANALYSIS

```
Context._msgData() (contracts/Token.sol#14-17) is never used and should be removed
ERC20._burn(address,uint256) (contracts/Token.sol#399-407) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/Token.sol#568-570) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/Token.sol#584-587) is never used and should be removed
SafeMathInt.abs(int256) (contracts/Token.sol#690-693) is never used and should be removed
SafeMathInt.add(int256,int256) (contracts/Token.sol#681-685) is never used and should be removed
SafeMathInt.div(int256,int256) (contracts/Token.sol#661-667) is never used and should be removed
SafeMathInt.mul(int256,int256) (contracts/Token.sol#649-656) is never used and should be removed
SafeMathInt.sub(int256,int256) (contracts/Token.sol#672-676) is never used and should be removed
SafeMathInt.toUint256Safe(int256) (contracts/Token.sol#695-698) is never used and should be removed
SafeMathUint.toInt256Safe(uint256) (contracts/Token.sol#702-706) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.9 (contracts/Token.sol#7) allows old versions
solc-0.8.20 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in TuzkiCeo.swapBack() (contracts/Token.sol#1186-1225):
        - (success,None) = address(marketingWallet).call{value: address(this).balance}() (contracts/Token.sol#1224)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IUniswapV2Pair.DOMAIN_SEPARATOR() (contracts/Token.sol#35) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (contracts/Token.sol#36) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (contracts/Token.sol#54) is not in mixedCase
Function IUniswapV2Router01.WETH() (contracts/Token.sol#711) is not in mixedCase
Event TuzkiCeomarketingWalletUpdated(address,address) (contracts/Token.sol#922) is not in CapWords
Parameter TuzkiCeo.updateBuyFees(uint256,uint256)._marketingFee (contracts/Token.sol#1033) is not in mixedCase
Parameter TuzkiCeo.updateBuyFees(uint256,uint256)._liquidityFee (contracts/Token.sol#1033) is not in mixedCase
Parameter TuzkiCeo.updateSellFees(uint256,uint256)._marketingFee (contracts/Token.sol#1040) is not in mixedCase
Parameter TuzkiCeo.updateSellFees(uint256,uint256)._liquidityFee (contracts/Token.sol#1040) is not in mixedCase
Parameter TuzkiCeo.updateTransferFees(uint256,uint256)._marketingFee (contracts/Token.sol#1047) is not in mixedCase
Parameter TuzkiCeo.updateTransferFees(uint256,uint256)._liquidityFee (contracts/Token.sol#1047) is not in mixedCase
Constant TuzkiCeo.deadAddress (contracts/Token.sol#878) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/Token.sol#15)" inContext (contracts/Token.sol#9-18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Token.sol#716) is too similar to IUniswapV2Router01.
addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Token.sol#717)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

TuzkiCeo.constructor() (contracts/Token.sol#930-993) uses literals with too many digits:
        - totalSupply = 1000000000000 * (10 ** _decimals) (contracts/Token.sol#935)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

SafeMathInt.MAX_INT256 (contracts/Token.sol#644) is never used in SafeMathInt (contracts/Token.sol#642-699)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

TuzkiCeo._decimals (contracts/Token.sol#884) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

Result => A static analysis of contract's source code has been performed using slither,

No major issues were found in the output

FUNCTIONAL TESTING

1- Adding liquidity (passed):

https://testnet.bscscan.com/tx/0xe268a57cde24d2f783707573250009b494498f138f8a02a8d06678f9da46de515

2- Buying when excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0xe9a219003eab1e640e48f10f9c8c4082629cf9b64588f07f15aac8b88d6d6318

3- Selling when excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0xf421b0b5519b18b63231efe4ad61d57d409cb6ae14c0c8b3946c7464e40575d2

4- Transferring when excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0x0dee8b9de63bc7ebb5e15c411344ad600b32ef413c4cab77f7af7ad9716a0ac0

5- Buying when not excluded from fees (0-11% tax) (passed):

https://testnet.bscscan.com/tx/0x0e2a844133255e3a3efa3bc2057f88728caf4c0603f7aeaeef38e45f32641930

6- Selling when not excluded from fees (0-11% tax) (passed):

https://testnet.bscscan.com/tx/0x42125580de945e4bd749810d93ac3267a8a63cc448e81bf58944fc9846666ead

FUNCTIONAL TESTING

7- Transferring when not excluded from fees (0-11% tax) (passed):
https://testnet.bscscan.com/tx/0xc2e4dc3660dff288a2ae9f18215b72ec2c95420cd1b85580510cd360c40ff330

8- Internal swap (passed):
- Tax converted to BNB and sent to marketing wallet
- Tax converted to BNB and added to liquidity pool along with an equivalent (in value) number of tokens
https://testnet.bscscan.com/tx/0x42125580de945e4bd749810d93ac3267a8a63cc448e81bf58944fc9846666ead

9- Airdropping before launch (passed):
airdrop gets disabled after trades are enabled.
https://testnet.bscscan.com/tx/0x3c51e0a0bd87c0e38c99ff03822f44f7a35de016561e713548279f90c741b735

10- Rescue tax (passed):
a feature that directly sends all collected fees to marketing wallet instead of swapping those tokens for BNB.
https://testnet.bscscan.com/tx/0xcf25588cc4d69484439e85e42c17d0f419295e6f570ce56735f361752e8a06e4

MANUAL TESTING

Centralization – Trades must be enabled

Severity: High

function: enableTrading

Status: Open

Overview:

The smart contract owner must enable trades for holders. If trading remain disabled, no one would be able to buy/sell/transfer tokens.

```
function enableTrading() external onlyOwner {
    tradingActive = true;
    swapEnabled = true;
}
```

Suggestion

To mitigate this centralization issue, we propose the following options:

1. Renounce Ownership: Consider relinquishing control of the smart contract by renouncing ownership. This would remove the ability for a single entity to manipulate the router, reducing centralization risks.
2. Multi-signature Wallet: Transfer ownership to a multi-signature wallet. This would require multiple approvals for any changes to the mainRouter, adding an additional layer of security and reducing the centralization risk.
3. Transfer ownership to a trusted and valid 3rd party in order to guarantee enabling of the trade

MANUAL TESTING

Logical – Setting marketing wallet to 0x0000000000000000000000000000000000000000

Severity: High

function: updateMarketingWallet

Status: Open

Overview:

Setting the marketing wallet to the null address (0x0000000000000000000000000000000000000000) may lead to the disruption of sale transactions when the "rescueSwap" feature is active. Specifically, accumulated fees, which should be transferred to the marketing wallet, would fail to be processed since the null address is not a valid recipient. Consequently, the entire sale transaction would be reverted due to this internal swap failure.

```
function updateMarketingWallet(address newMarketingWallet) external onlyOwner {
    emit marketingWalletUpdated(newMarketingWallet, marketingWallet);
    marketingWallet = newMarketingWallet;
}

function swapBack() private {
    uint256 contractBalance = balanceOf(address(this));

    if (rescueSwap) {
        if (contractBalance > 0) {
            super._transfer(address(this), marketingWallet, contractBalance);
        }
        return;
    }
    ///rest of the code
}
```

Suggestion

Prevent the marketing wallet from being set to the null address by introducing a requirement check within the updateMarketingWallet function. A modified version of the function would look like this:

```
function updateMarketingWallet(address newMarketingWallet) external onlyOwner {
    emit marketingWalletUpdated(newMarketingWallet, marketingWallet);
    require(newMarketingWallet != address(0), "can not set marketing wallet to address 0");
    marketingWallet = newMarketingWallet;
}
```

MANUAL TESTING

Logical – fees are not resseted

Severity: High

function: swapBack

Status: Open

Overview:

Currently, the variables **tokensForLiquidity** and **tokensForMarketing** represent the amount of tokens accumulated for each type of tax. However, the swapBack function fails to reset these two variables to zero after sending the contract token balance to the marketing wallet.

While the owner can manually reset these variables to zero using the resetTaxAmount function, a failure to perform this action could cause an internal swap to revert. This would occur when the rescueSwap flag is set to false and an attempt is made to swap more tokens to BNB than what the contract has.

```
function resetTaxAmount() public onlyOwner {
    tokensForLiquidity = 0;
    tokensForMarketing = 0;
}

function swapBack() private {
    uint256 contractBalance = balanceOf(address(this));

    if (rescueSwap) {
        if (contractBalance > 0) {
            super._transfer(address(this), marketingWallet, contractBalance);
        }
        return;
    }
    ///rest of the code
}
```

Suggestion

Ensure that tokensForLiquidity and tokensForMarketing are reset to zero in the swapBack function.

A modified version of the function is as follows:

```
function swapBack() private {
    uint256 contractBalance = balanceOf(address(this));

    if (rescueSwap) {
        if (contractBalance > 0) {
            super._transfer(address(this), marketingWallet, contractBalance);
        }
        tokensForLiquidity = 0;
        tokensForMarketing = 0;
        return;
    }
    ///rest of the code
}
```

MANUAL TESTING

Logical – Stuck ETH and Tokens

Severity: Medium

Status: Open

Overview:

This contract can receive both ETH and all types of ERC20 tokens. However, there are currently no functions available to withdraw these stuck tokens or ETH. This could result in assets being permanently locked within the contract.

Suggestion

Recommendation: To resolve this issue, implement withdrawal functions for both ETH and ERC20 tokens. This will allow the contract owner to recover any assets mistakenly sent to the contract address. Here is an example of how such functions might look:

javascript

```
// For ERC20 Tokens
function withdrawTokens(address _tokenContract) external onlyOwner {
   require(_tokenContract != address(this), “can not withdraw native tokens”);
   ERC20 token = ERC20(_tokenContract);
   uint256 balance = token.balanceOf(address(this));
   token.transfer(owner, balance);
   emit WithdrawalTokens(owner, balance);
}

// For ETH
function withdrawETH(uint256 amount) external onlyOwner {
   require(amount <= address(this).balance, "Not enough ETH balance");
   payable(owner).transfer(amount);
   emit WithdrawalETH(owner, amount);
}
```

In these functions, onlyOwner is a modifier to ensure only the contract owner can execute these functions, preventing unauthorized withdrawals. The WithdrawalTokens and WithdrawalETH events are emitted after the successful transfer of tokens or ETH to provide transparency and trackability.

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. 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 Auditace 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 Auditace 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 Auditace team disclaims any liability for the resulting losses.

ABOUT AUDITACE

We specializes in providing thorough and reliable audits for Web3 projects. With a team of experienced professionals, we use cutting-edge technology and rigorous methodologies to evaluate the security and integrity of blockchain systems. We are committed to helping our clients ensure the safety and transparency of their digital assets and transactions.

https://auditace.tech/

https://t.me/Audit_Ace

https://twitter.com/auditace_

https://github.com/Audit-Ace