auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

GROK CEO

DATED : 10 January 2024

MANUAL TESTING

Centralization - Enabling Trades
Severity: High
Function: setTradingEnable
Overview:
The **setTradingEnable** function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

```
function setTradingEnabled() external onlyOwner {
require(!tradingEnabled, "Trade is already enabled");
    tradingEnabled = true;
  }
```

Suggestion:

To reduce centralization and potential manipulation, consider one of the following approaches:

1.Automatically enable trading after a specified condition, such as the completion of a presale, is met. bad faith actions by the original owner.

MANUAL TESTING

2.If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can provide investors with more confidence in the eventual activation of trading capabilities, mitigating concerns of potential

AUDIT SUMMARY

Project name – GROK CEO

Date: 10 January, 2024

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

Audit Status: **Passed with High Risk**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	1	0	2	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/0x5dedcd8b43e335d30ddfeb968cc3c59ccb729968#code

Token Information

Token Address:
0x6294DEB10817e47a4EE6869DB59c85F3eF4BEE29

Name: GROK CEO

Symbol: GROKCEO

Decimals: 9

Network: BscScan

Token Type: BEP-20

Owner:
0xa860467495a5df4d63D86f6ccA216eBF7ab45C31

Deployer:
0xa860467495a5df4d63D86f6ccA216eBF7ab45C31

Token Supply: 420,000,000,000,000,000

Checksum: Ae1c3a4fbb6e83e8393a57617b5a5b23

Testnet:
https://testnet.bscscan.com/address/0x5dedcd8b43e335d30ddfeb968cc3c59ccb729968#code

TOKEN OVERVIEW

Transfer Fee between Wallets: 0%

Burn: 1%

Marketing: 2.4%

Developing: 1.6%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: No

Blacklist: No

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

STATIC ANALYSIS

```
INFO:Detectors:
GROKCEO.distributeTaxes(uint256) (GROKCEO.sol#971-1000) performs a multiplication on the result of a division:
        - buybackTaxRatio = (taxRates.buyback * 100) / totalTaxRate (GROKCEO.sol#978)
        - buybackTokens = (buybackTaxRatio * contractTokenBalance) / 100 (GROKCEO.sol#981)
GROKCEO.distributeTaxes(uint256) (GROKCEO.sol#971-1000) performs a multiplication on the result of a division:
        - marketingTaxRatio = (taxRates.marketing * 100) / totalTaxRate (GROKCEO.sol#976)
        - marketingEth = (remainingEth * marketingTaxRatio) / 100 (GROKCEO.sol#991)
GROKCEO.distributeTaxes(uint256) (GROKCEO.sol#971-1000) performs a multiplication on the result of a division:
        - developingTaxRatio = (taxRates.developing * 100) / totalTaxRate (GROKCEO.sol#979)
        - developingEth = (remainingEth * developingTaxRatio) / 100 (GROKCEO.sol#992)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
GROKCEO.changeSwapTokenAtAmount(uint256) (GROKCEO.sol#1064-1071) should emit an event for:
        - swapTokenAtAmount = newAmount (GROKCEO.sol#1070)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
GROKCEO.changeMarketingWallet(address)._marketing (GROKCEO.sol#1002) lacks a zero-check on :
                - marketing = _marketing (GROKCEO.sol#1003)
GROKCEO.changeDeveloperWallet(address)._developer (GROKCEO.sol#1006) lacks a zero-check on :
                - developing = _developer (GROKCEO.sol#1007)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in GROKCEO._transfer(address,address,uint256) (GROKCEO.sol#937-969):
        External calls:
        - distributeTaxes(contractTokenBalance) (GROKCEO.sol#955)
                - (success) = address(_to).call{value: amount}() (GROKCEO.sol#1017)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (GROKCEO.sol#1045-1053)
        External calls sending eth:
        - distributeTaxes(contractTokenBalance) (GROKCEO.sol#955)
                - (success) = address(_to).call{value: amount}() (GROKCEO.sol#1017)
        Event emitted after the call(s):
        - Transfer(from,to,value) (GROKCEO.sol#669)
                - super._transfer(sender,recipient,transferAmount) (GROKCEO.sol#967)
        - Transfer(from,to,value) (GROKCEO.sol#669)
                - super._transfer(sender,address(this),taxAmount) (GROKCEO.sol#966)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
INFO:Detectors:
Context._contextSuffixLength() (GROKCEO.sol#212-214) is never used and should be removed
Context._msgData() (GROKCEO.sol#208-210) is never used and should be removed
ERC20._burn(address,uint256) (GROKCEO.sol#695-700) is never used and should be removed
ReentrancyGuard._reentrancyGuardEntered() (GROKCEO.sol#855-857) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.19 (GROKCEO.sol#14) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
Pragma version^0.8.19 (GROKCEO.sol#191) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
Pragma version^0.8.19 (GROKCEO.sol#221) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
Pragma version^0.8.19 (GROKCEO.sol#324) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
Pragma version^0.8.19 (GROKCEO.sol#416) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
Pragma version^0.8.19 (GROKCEO.sol#442) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
Pragma version^0.8.19 (GROKCEO.sol#791) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in GROKCEO.sendBNB(address,uint256) (GROKCEO.sol#1010-1020):
        - (success) = address(_to).call{value: amount}() (GROKCEO.sol#1017)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IRouter.WETH() (GROKCEO.sol#802) is not in mixedCase
Parameter GROKCEO.changeMarketingWallet(address)._marketing (GROKCEO.sol#1002) is not in mixedCase
Parameter GROKCEO.changeDeveloperWallet(address)._developer (GROKCEO.sol#1006) is not in mixedCase
Parameter GROKCEO.sendBNB(address,uint256)._to (GROKCEO.sol#1010) is not in mixedCase
Function GROKCEO.AddExemptFee(address) (GROKCEO.sol#1056-1058) is not in mixedCase
Parameter GROKCEO.AddExemptFee(address)._address (GROKCEO.sol#1056) is not in mixedCase
Function GROKCEO.RemoveExemptFee(address) (GROKCEO.sol#1060-1062) is not in mixedCase
Parameter GROKCEO.RemoveExemptFee(address)._address (GROKCEO.sol#1060) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
GROKCEO.feeDenominator (GROKCEO.sol#873) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

STATIC ANALYSIS

```
INFO:Detectors:
GROKCEO.feeDenominator (GROKCEO.sol#873) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
GROKCEO.pair (GROKCEO.sol#862) should be immutable
GROKCEO.router (GROKCEO.sol#861) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:GROKCEO.sol analyzed (12 contracts with 93 detectors), 34 result(s) found
```

FUNCTIONAL TESTING

1- **Approve** (passed):
https://testnet.bscscan.com/tx/0x93e96e05ab6a1a187b183a64c63eae1678c586aed513aaa0f5bf43c180f978db

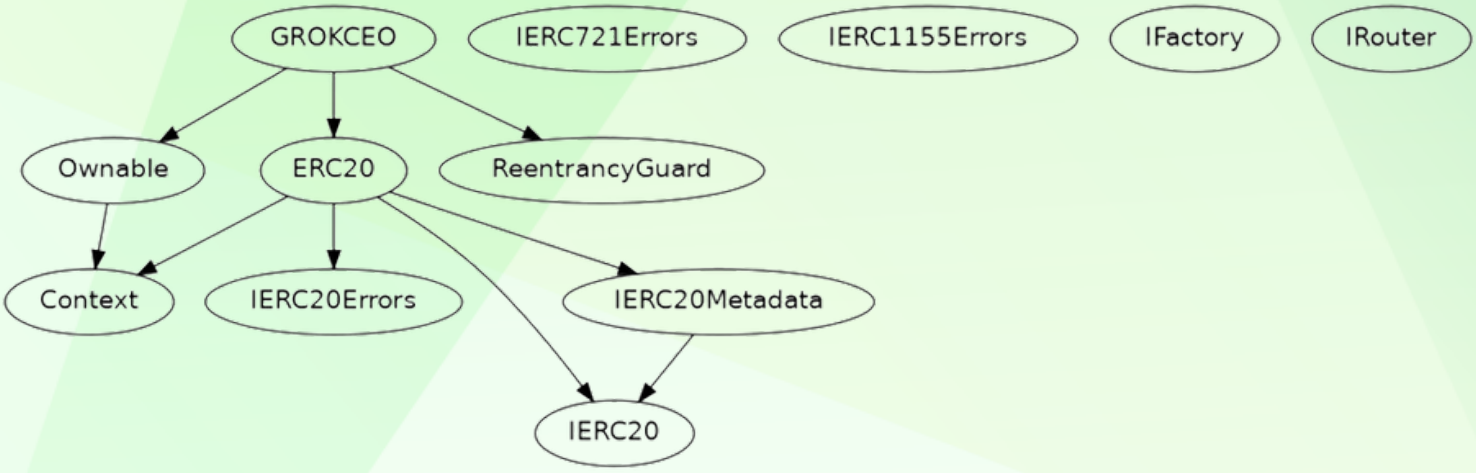
2- **Add Exempt Fee** (passed):
https://testnet.bscscan.com/tx/0x092fd2a779c561253e05ecba6ceaa51fa9542d640959ba92e8e5ab7a0f699c08

3- **Remove Exempt Fee** (passed):
https://testnet.bscscan.com/tx/0xe883e7d3277dd26d7735009539f53a1b7f28004c8406cd411daf05e2636db364

4- **Set Trading Enabled** (passed):
https://testnet.bscscan.com/tx/0xa73b977bca06665f1cf77096dd46d0340c46bd6bbf0c589a3bab6577576a3947

5- **Update Exempt Fee** (passed):
https://testnet.bscscan.com/tx/0xe3841f18d8912f7bb27fed6077f9b7e02367f4c05a67b3f1c8d2a84fbc02fdf7

INHERITANCE TREE



POINTS TO NOTE

-Whitelist to transfer without enabling trades

- Enabling trades

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	1
Medium-Risk	0
Low-Risk	2
Gas Optimization / Suggestions	2

MANUAL TESTING

Centralization - Enabling Trades
Severity: High
Function: setTradingEnable
Overview:
The **setTradingEnable** function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

```
function setTradingEnabled() external onlyOwner {
require(!tradingEnabled, "Trade is already enabled");
    tradingEnabled = true;
  }
```

Suggestion:

To reduce centralization and potential manipulation, consider one of the following approaches:

1.Automatically enable trading after a specified condition, such as the completion of a presale, is met. bad faith actions by the original owner.

MANUAL TESTING

2.If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can provide investors with more confidence in the eventual activation of trading capabilities, mitigating concerns of potential

MANUAL TESTING

Centralization - Missing Events
Severity: Low
Subject: Missing Events
Status: Open

Overview:
They serve as a mechanism for emitting and recording data onto the blockchain, making it transparent and easily accessible.

```
function changeMarketingWallet(address _marketing)
external onlyOwner {
    marketing = _marketing;
  }
function changeDeveloperWallet(address _developer)
external onlyOwner {
    developing = _developer;
  }
```

MANUAL TESTING

```
 function setAutomatedMarketMakerPair(address
lpPair, bool value)
external
 onlyOwner
 {
require(
 pair != lpPair,
"The pair cannot be removed from
automatedMarketMakerPairs"
 );

 _setAutomatedMarketMakerPair(pair, value);
 }
function changeSwapTokenAtAmount(uint256
newAmount) external onlyOwner {
require(
 newAmount > totalSupply() / 100_000 &&
 newAmount < (totalSupply() / 100),
"Amount should be greater than 1 and less than 1% of
total supply"
 );
 swapTokenAtAmount = newAmount;
 }
```

MANUAL TESTING

Centralization - Missing Zero Address

Severity: Low

Status: Open

Overview:
Functions can take a zero address as a parameter (0x00000...). If a function parameter of address type is not properly validated by checking for zero addresses, there could be serious consequences for the contract's functionality.

```
function changeMarketingWallet(address _marketing)
external onlyOwner {
   marketing = _marketing;
 }
function changeDeveloperWallet(address _developer)
external onlyOwner {
   developing = _developer;
 }
```

Suggestion:
It is suggested that the address should not be zero or dead.

MANUAL TESTING

Optimization

Severity: Optimization

Subject: Remove unused code.

Status: Open

Overview:

Unused variables are allowed in Solidity, and they do. not pose a direct security issue. It is the best practice. though to avoid them.

```
function _msgData() internal view virtual returns (bytes memory) {
    this;
return msg.data;
  }
```

MANUAL TESTING

Optimization

Severity: Informational

Subject: Floating Pragma Solidity version

Status: Open

Overview:
It is considered best practice to pick one compiler version and stick with it. With a floating pragma, contracts may accidentally be deployed using an outdated.

pragma solidity ^0.8.19;

Suggestion:
Adding the latest constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

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