expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

Grok bnb

TOKEN OVERVIEW

Risk Findings

Severity	Found
High	2
Medium	0
Low	1
Informational	2

Centralization Risks

Owner Privileges	Description
Can Owner Set Taxes >25% ?	Not Detected
Owner needs to enable trading ?	Yes, owner needs to enable trades
Can Owner Disable Trades ?	Not Detected
Can Owner Mint ?	Not Detected
Can Owner Blacklist ?	Not Detected
Can Owner set Max Wallet amount ?	Not Detected
Can Owner Set Max TX amount ?	Not Detected

TABLE OF CONTENTS

OVERVIEW

The Expelee team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analysed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

Audit Result	**Passed with high risk**
Audit Date	**4 March 2024**

CONTRACT DETAILS

Token Address: 0xD2f95f97E165Ae33229693130ef39D55CcA35a04

Name: Grok bnb

Symbol: GBNB

Decimals: 18

Network: BscScan

Token Type: BEP-20

Owner: 0xa5BcC4085336ee2cB4e83b1a1796259989F1DD4A

Deployer: 0x6250b676F2814c43483FD1ddA4aB584DeE4e5435

Token Supply: 20000000000

Checksum: A2032c616934aeb47e6039f76b20d223

Testnet:
https://testnet.bscscan.com/address/0xa059b518f93b9f9b6ee85e1cd41384a183d617ea#code

AUDIT METHODOLOGY

Audit Details

Our comprehensive audit report provides a full overview of the audited system's architecture, smart contract codebase, and details on any vulnerabilities found within the system.

Audit Goals

The audit goal is to ensure that the project is built to protect investors and users, preventing potentially catastrophic vulnerabilities after launch , that lead to scams and rugpulls.

Code Quality

Our analysis includes both automatic tests and manual code analysis for the following aspects:

- Exploits
- Back-doors
- Vulnerability
- Accuracy
- Readability

Tools

- DE
- Open Zeppelin
- Code Analyzer
- Solidity Code
- Compiler
- Hardhat

VULNERABILITY CHECKS

Design Logic	Passed
Compiler warnings	Passed
Private user data leaks	Passed
Timestamps dependence	Passed
Integer overflow and underflow	Passed
Race conditions & reentrancy. Cross-function race conditions	Passed
Possible delays in data delivery	Passed
Oracle calls	Passed
Front Running	Passed
DoS with Revert	Passed
DoS with block gas limit	Passed
Methods execution permissions	Passed
Economy model	Passed
Impact of the exchange rate on the logic	Passed
Malicious event log	Passed
Scoping and declarations	Passed
Uninitialized storage pointers	Passed
Arithmetic accuracy	Passed
Cross-function race conditions	Passed
Safe Zepplin module	Passed

RISK CLASSIFICATION

When performing smart contract audits, our specialists look for known vulnerabilities as well as logical and acces control issues within the code. The exploitation of these issues by malicious actors may cause serious financial damage to projects that failed to get an audit in time. We categorize these vulnerabilities by the following levels:

High Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Medium Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

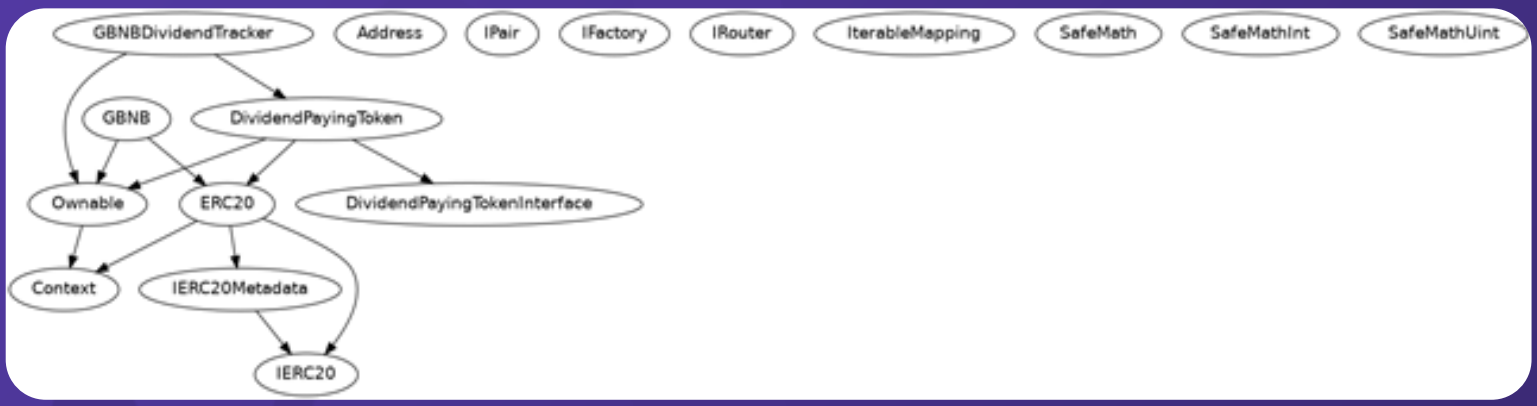
Low Risk

Issues on this level are minor details and warning that can remain unfixed.

Informational

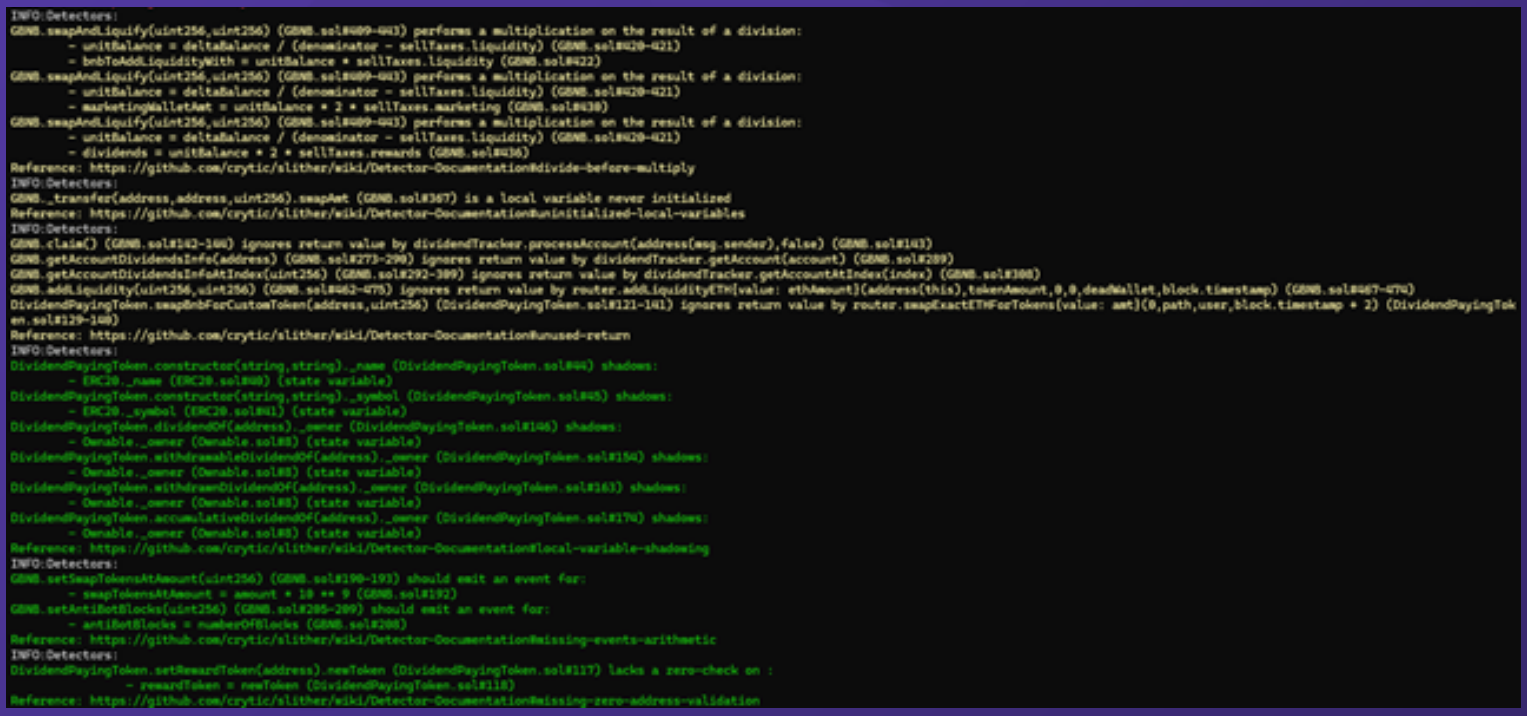
Issues on this level are minor details and warning that can remain unfixed.

INHERITANCE TREES



STATIC ANALYSIS

A static analysis of the code was performed using Slither. No issues were found.



```
INFO:Detectors:
GBNB.swapAndLiquify(uint256,uint256) (GBNB.sol#409-443) performs a multiplication on the result of a division:
        - unitBalance = deltaBalance / (denominator - sellTaxes.liquidity) (GBNB.sol#420-421)
        - bnbToAddLiquidityWith = unitBalance * sellTaxes.liquidity (GBNB.sol#422)
GBNB.swapAndLiquify(uint256,uint256) (GBNB.sol#409-443) performs a multiplication on the result of a division:
        - unitBalance = deltaBalance / (denominator - sellTaxes.liquidity) (GBNB.sol#420-421)
        - marketingWalletAmt = unitBalance * 2 * sellTaxes.marketing (GBNB.sol#430)
GBNB.swapAndLiquify(uint256,uint256) (GBNB.sol#409-443) performs a multiplication on the result of a division:
        - unitBalance = deltaBalance / (denominator - sellTaxes.liquidity) (GBNB.sol#420-421)
        - dividends = unitBalance * 2 * sellTaxes.rewards (GBNB.sol#436)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
GBNB._transfer(address,address,uint256).swapAmt (GBNB.sol#367) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
GBNB.claim() (GBNB.sol#142-146) ignores return value by dividendTracker.processAccount(address(msg.sender),false) (GBNB.sol#143)
GBNB.getAccountDividendsInfo(address) (GBNB.sol#273-290) ignores return value by dividendTracker.getAccount(account) (GBNB.sol#289)
GBNB.getAccountDividendsInfoAtIndex(uint256) (GBNB.sol#292-309) ignores return value by dividendTracker.getAccountAtIndex(index) (GBNB.sol#308)
GBNB.addLiquidity(uint256,uint256) (GBNB.sol#462-475) ignores return value by router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (GBNB.sol#467-474)
DividendPayingToken.swapBnbForCustomToken(address,uint256) (DividendPayingToken.sol#121-141) ignores return value by router.swapExactETHForTokens{value: amt}(0,path,user,block.timestamp + 2) (DividendPayingTok
en.sol#129-140)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
DividendPayingToken.constructor(string,string)._name (DividendPayingToken.sol#44) shadows:
        - ERC20._name (ERC20.sol#40) (state variable)
DividendPayingToken.constructor(string,string)._symbol (DividendPayingToken.sol#45) shadows:
        - ERC20._symbol (ERC20.sol#41) (state variable)
DividendPayingToken.dividendOf(address)._owner (DividendPayingToken.sol#146) shadows:
        - Ownable._owner (Ownable.sol#8) (state variable)
DividendPayingToken.withdrawableDividendOf(address)._owner (DividendPayingToken.sol#154) shadows:
        - Ownable._owner (Ownable.sol#8) (state variable)
DividendPayingToken.withdrawnDividendOf(address)._owner (DividendPayingToken.sol#163) shadows:
        - Ownable._owner (Ownable.sol#8) (state variable)
DividendPayingToken.accumulativeDividendOf(address)._owner (DividendPayingToken.sol#174) shadows:
        - Ownable._owner (Ownable.sol#8) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
GBNB.setSwapTokensAtAmount(uint256) (GBNB.sol#190-193) should emit an event for:
        - swapTokensAtAmount = amount * 10 ** 9 (GBNB.sol#192)
GBNB.setAntiBotBlocks(uint256) (GBNB.sol#205-209) should emit an event for:
        - antiBotBlocks = numberOfBlocks (GBNB.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
DividendPayingToken.setRewardToken(address).newToken (DividendPayingToken.sol#117) lacks a zero-check on :
        - rewardToken = newToken (DividendPayingToken.sol#118)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

expelee

STATIC ANALYSIS

```
INFO:Detectors:
GBNB.setSwapTokensAtAmount(uint256) (GBNB.sol#190-193) should emit an event for:
        - swapTokensAtAmount = amount * 10 ** 9 (GBNB.sol#192)
GBNB.setAntiBotBlocks(uint256) (GBNB.sol#205-209) should emit an event for:
        - antiBotBlocks = numberOfBlocks (GBNB.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
DividendPayingToken.setRewardToken(address).newToken (DividendPayingToken.sol#117) lacks a zero-check on :
                - rewardToken = newToken (DividendPayingToken.sol#118)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
DividendPayingToken._withdrawDividendOfUser(address) (DividendPayingToken.sol#74-115) has external calls inside a loop: rewardToken != router.WETH() (DividendPayingToken.sol#83)
DividendPayingToken.swapBnbForCustomToken(address,uint256) (DividendPayingToken.sol#121-141) has external calls inside a loop: path[0] = router.WETH() (DividendPayingToken.sol#126)
DividendPayingToken.swapBnbForCustomToken(address,uint256) (DividendPayingToken.sol#121-141) has external calls inside a loop: router.swapExactETHForTokens{value: amt}(0,path,user,block.timestamp + 2) (Dividen
dPayingToken.sol#129-140)
DividendPayingToken._withdrawDividendOfUser(address) (DividendPayingToken.sol#74-115) has external calls inside a loop: (secondSuccess) = user.call{gas: 3000,value: _withdrawableDividend}() (DividendPayingToke
n.sol#89-92)
DividendPayingToken._withdrawDividendOfUser(address) (DividendPayingToken.sol#74-115) has external calls inside a loop: (success_scope_0) = user.call{gas: 3000,value: _withdrawableDividend}() (DividendPayingTo
ken.sol#101-104)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in GBNBDividendTracker.processAccount(address,bool) (GBNB.sol#714-727):
        External calls:
        - amount = _withdrawDividendOfUser(account) (GBNB.sol#718)
                - router.swapExactETHForTokens{value: amt}(0,path,user,block.timestamp + 2) (DividendPayingToken.sol#129-140)
                - (secondSuccess) = user.call{gas: 3000,value: _withdrawableDividend}() (DividendPayingToken.sol#89-92)
                - (success_scope_0) = user.call{gas: 3000,value: _withdrawableDividend}() (DividendPayingToken.sol#101-104)
        State variables written after the call(s):
        - lastClaimTimes[account] = block.timestamp (GBNB.sol#721)
Reentrancy in GBNB.swapAndLiquify(uint256,uint256) (GBNB.sol#409-443):
        External calls:
        - swapTokensForBNB(toSwap) (GBNB.sol#417)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (GBNB.sol#453-459)
        - addLiquidity(tokensToAddLiquidityWith,bnbToAddLiquidityWith) (GBNB.sol#426)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (GBNB.sol#467-474)
        External calls sending eth:
        - addLiquidity(tokensToAddLiquidityWith,bnbToAddLiquidityWith) (GBNB.sol#426)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (GBNB.sol#467-474)
        State variables written after the call(s):
        - addLiquidity(tokensToAddLiquidityWith,bnbToAddLiquidityWith) (GBNB.sol#426)
                - _allowances[owner][spender] = amount (ERC20.sol#323)
Reentrancy in GBNB.updateDividendTracker(address) (GBNB.sol#111-124):
        External calls:
        - newDividendTracker.excludeFromDividends(address(newDividendTracker),true) (GBNB.sol#116-119)
        - newDividendTracker.excludeFromDividends(address(this),true) (GBNB.sol#120)
        - newDividendTracker.excludeFromDividends(owner(),true) (GBNB.sol#121)
        - newDividendTracker.excludeFromDividends(address(router),true) (GBNB.sol#122)
        State variables written after the call(s):
        - dividendTracker = newDividendTracker (GBNB.sol#123)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
INFO:Detectors:
Low level call in DividendPayingToken._withdrawDividendOfUser(address) (DividendPayingToken.sol#74-115):
        - (secondSuccess) = user.call{gas: 3000,value: _withdrawableDividend}() (DividendPayingToken.sol#89-92)
        - (success_scope_0) = user.call{gas: 3000,value: _withdrawableDividend}() (DividendPayingToken.sol#101-104)
Low level call in Address.sendValue(address,uint256) (GBNB.sol#11-22):
        - (success) = recipient.call{value: amount}() (GBNB.sol#17)
Low level call in GBNB.swapAndLiquify(uint256,uint256) (GBNB.sol#409-443):
        - (success) = address(dividendTracker).call{value: dividends}() (GBNB.sol#438-440)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter DividendPayingToken.dividendOf(address)._owner (DividendPayingToken.sol#146) is not in mixedCase
Parameter DividendPayingToken.withdrawableDividendOf(address)._owner (DividendPayingToken.sol#154) is not in mixedCase
Parameter DividendPayingToken.withdrawnDividendOf(address)._owner (DividendPayingToken.sol#163) is not in mixedCase
Parameter DividendPayingToken.accumulativeDividendOf(address)._owner (DividendPayingToken.sol#174) is not in mixedCase
Constant DividendPayingToken.magnitude (DividendPayingToken.sol#20) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter GBNB.setSwapEnabled(bool)._enabled (GBNB.sol#195) is not in mixedCase
Parameter GBNBDividendTracker.getAccount(address)._account (GBNB.sol#550) is not in mixedCase
Function IRouter.WETH() (IDex.sol#16) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (Context.sol#21)" inContext (Context.sol#15-25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable DividendPayingToken._withdrawDividendOfUser(address)._withdrawableDividend (DividendPayingToken.sol#77) is too similar to GBNBDividendTracker.getAccount(address).withdrawableDividends (GBNB.sol#567)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
GBNB.setGasForProcessing(uint256) (GBNB.sol#228-239) uses literals with too many digits:
        - require(bool,string)(newValue >= 200000 && newValue <= 500000,GBNB: gasForProcessing must be between 200,000 and 500,000) (GBNB.sol#229-232)
GBNB.slitherConstructorVariables() (GBNB.sol#25-476) uses literals with too many digits:
        - gasForProcessing = 300000 (GBNB.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
Ownable._previousOwner (Ownable.sol#9) is never used in GBNB (GBNB.sol#25-476)
GBNB.currentRewardToken (GBNB.sol#44) is never used in GBNB (GBNB.sol#25-476)
Ownable._previousOwner (Ownable.sol#9) is never used in GBNBDividendTracker (GBNB.sol#478-729)
SafeMathInt.MAX_INT256 (SafeMath.sol#166) is never used in SafeMathInt (SafeMath.sol#160-221)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
GBNB.currentRewardToken (GBNB.sol#44) should be constant
GBNB.launchTax (GBNB.sol#58) should be constant
Ownable._previousOwner (Ownable.sol#9) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
DividendPayingToken.router (DividendPayingToken.sol#22) should be immutable
GBNB.pair (GBNB.sol#29) should be immutable
GBNB.router (GBNB.sol#28) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:GBNB.sol analyzed (17 contracts with 93 detectors), 89 result(s) found
```

TESTNET VERSION

1- Approve (passed):
https://testnet.bscscan.com/tx/0x4d54c757f3a2415e9bda8392959f2297fe13ed3cf6d0baaa494ea363f24615c6

2- Increase Allowance (passed):
https://testnet.bscscan.com/tx/0x645c7304bb3b60ceee41aef5d3835c373fa0a791241fa99ce02ca43226cc6cf5

3- Decrease Allowance (passed):
https://testnet.bscscan.com/tx/0x9d38fe6416735a24c6b89a6004d55d51314ab34de049b7863fe4ed85ede0c3b0

4- Enable Trading (passed):
https://testnet.bscscan.com/tx/0x25b0c6141f74c20dfddb501705bbc50bbf2c22ae781e7a9376b370022623d22a

5- Exclude From Dividends (passed):
https://testnet.bscscan.com/tx/0xc1b05dcdbdb63cb26edb36fa61d87ac541481cd0b70ebd22bd3eeb2ea9293b9e

6- Exclude From Fees (passed):
https://testnet.bscscan.com/tx/0x8c1f6685d412da44ca420b475f95583a24732b84d761b8e0eec1e90af64fdaec

7- Transfer (passed):
https://testnet.bscscan.com/tx/0x86c1144c2070827a90b5c14ebb43dc89bf58264d4dfa78a446a0a6aac75710de

8- Set marketing Wallet (passed):
https://testnet.bscscan.com/tx/0xb1e4ae3b9c77549261282e57a085f14ccd2a1f08fcc794f04004974ab639e823

MANUAL REVIEW

Severity Criteria

Expelee assesses the severity of disclosed vulnerabilities according to methodology based on OWASP standarts.

Vulnerabilities are dividend into three primary risk categroies:
High
Medium
Low

High-level considerations for vulnerabilities span the following key areas when conducting assessments:

- Malicious input handling
- Escalation of privileges
- Arithmetic
- Gas use

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

HIGH RISK FINDING

Centralization – Missing Require Check

Severity: High

function: SetMarketingWallet

Status: Open

Overview:

The owner can set any arbitrary address excluding zero address as this is not recommended because if the owner sets the address to the contract address, then the ETH will not be sent to that address and the transaction will fail and this will lead to a potential honeypot in the contract.

```
  function setMarketingWallet(address newWallet) external
onlyOwner {
require(newWallet != address(0), "Fee Address cannot be zero
address");
    marketingWallet = newWallet;
  }
function setRewardToken(address newToken) external
onlyOwner {
    rewardToken = newToken;
  }
```

Suggestion:

It is recommended that the address should not be able to set as a contract address.

HIGH RISK FINDING

Centralization – Enabling Trades

Severity: High

function: EnableTrading

Status: Open

Overview:

The EnableTrading 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 enableTrading() external onlyOwner {
 require(!tradingEnabled, "Trading is already enabled");
 tradingEnabled = true;
 startTradingBlock = block.number;

}
```

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.

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 give investors more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

LOW RISK FINDING

Centralization – Missing Events

Severity: Low

function: 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 setAntiBotBlocks(uint256 numberOfBlocks) external
onlyOwner {
require(!tradingEnabled, "Can't change when trading has started");
require(numberOfBlocks < 3, "Deadline should be less than 3 Blocks");
  antiBotBlocks = numberOfBlocks;
 }
function setBalance(address account, uint256 newBalance) public
onlyOwner {
if (excludedFromDividends[account]) {
return;
  }
if (newBalance >= minimumTokenBalanceForDividends) {
    _setBalance(account, newBalance);
    tokenHoldersMap.set(account, newBalance);
  } else {
    _setBalance(account, 0);
    tokenHoldersMap.remove(account);
  }
  processAccount(payable(account), true);
 }
```

Suggestion

Emit an event for critical changes.

INFORMATIONAL & OPTIMIZATIONS

Optimization
Severity: Informational
subject: Remove Safe Math
Status: Open
Line: 144-297

Overview:
compiler version above 0.8.0 can control arithmetic overflow/underflow, it is recommended to remove the unwanted code to avoid high gas fees.

INFORMATIONAL & OPTIMIZATIONS

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
calldata) {
   this; // silence state mutability warning without generating
bytecode - see
https://github.com/ethereum/solidity/issues/2691
return msg.data;
  }
}
interface IPair {
  function sync() external;
}
function get(Map storage map, address key) internal view
returns (uint) {
return map.values[key];
  }
```

ABOUT EXPELEE

Expelee is a product-based aspirational Web3 start-up. Coping up with numerous solutions for blockchain security and constructing a Web3 ecosystem from deal making platform to developer hosting open platform, while also developing our own commercial and sustainable blockchain.

www.expelee.com

expeleeofficial

expelee

Expelee

expelee

expelee_official

expelee-co

expelee

Building the Futuristic Blockchain Ecosystem

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 guarantess against the sale of team tokens or the removal of liquidity by the project audited in this document.

Always do your own research and project yourselves from being scammed. The Expelee 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 Expelee receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Alway 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 Expelee team disclaims any liability for the resulting losses.

expelee

Building the Futuristic Blockchain Ecosystem