auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

DOGE NEW

DATED : 09 Apr 23'

AUDIT SUMMARY

Project name – Doge New

Date: 09 April, 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	0	0	0	1
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- Slither : The code has undergone static analysis using Slither.

Token Information

Token Name : DOGENEW

Token Symbol: DOGENEW

Decimals: 9

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

Token Address:
0xB98ddD096917F9EEF591e69731cca0508De5F470

Checksum:
9bd464a05b50f5537ecc6c166c79d70f55339618

Owner:
0x0c1ED0Be0aA2780600436e51507d45f7523fa460
(at the time of audit)

TOKEN OVERVIEW

Fees:

Buy Fees: up to 25%

Sell Fees: up to 25%

Transfer Fees: up to 25%

Fees Privilige: Owner

Ownership : Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Priviliges: Including and excluding form fee - changing swap threshold - enabling trades - modifying 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

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

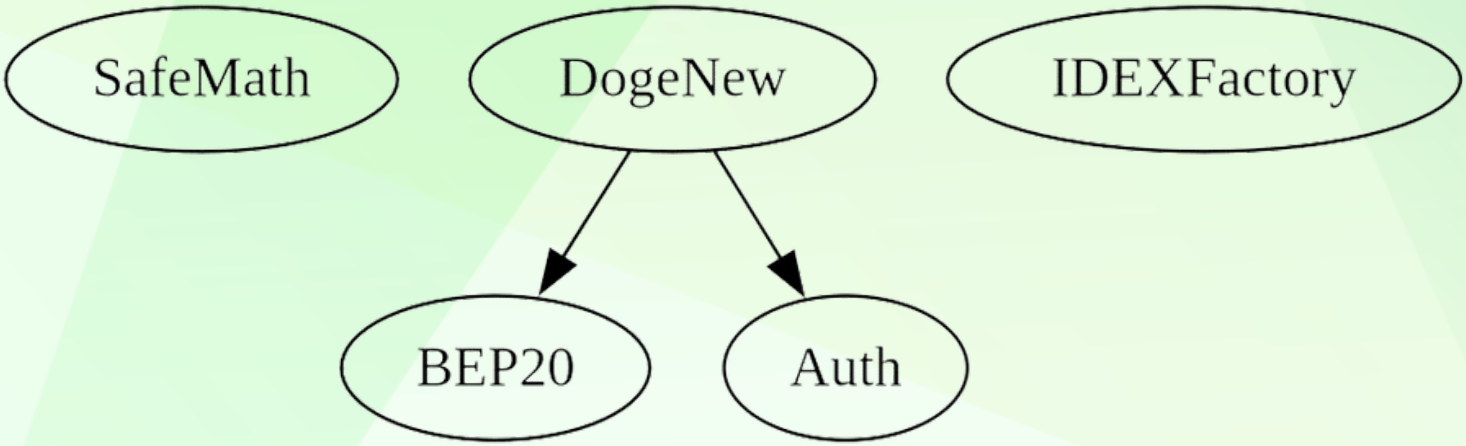
CLASSIFICATION OF RISK

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

INHERITANCE TREE



POINTS TO NOTE

- Fees can not be more than 25% for buy/sell/transfers (each one can be up to 25%)
- Owner is not able to set max buy/sell/transfer/hold amount
- Owner is not able to blacklist an arbitrary wallet
- Owner is not able to disable trades
- Owner is not able to mint new tokens

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
SafeMath	Library				
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
BEP20	Interface				
└	getOwner	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
Auth	Implementation				
└	<Constructor>	Public ❗	🛑	NO ❗
└	authorize	External ❗	🛑	onlyOwner
└	unauthorize	External ❗	🛑	onlyOwner
└	isOwner	Public ❗		NO ❗
└	isAuthorized	Public ❗		NO ❗
└	transferOwnership	External ❗	🛑	onlyOwner
└	acceptOwnership	External ❗	🛑	NO ❗
IDEXFactory	Interface				
└	createPair	External ❗	🛑	NO ❗
IDEXRouter	Interface				
└	factory	External ❗		NO ❗
└	WETH	External ❗		NO ❗
└	addLiquidityETH	External ❗	💵	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
DogeNew	Implementation	BEP20, Auth			
└	<Constructor>	Public ❗	🛑	Auth
└	<Receive Ether>	External ❗	💵	NO ❗
└	getOwner	External ❗		NO ❗
└	allowance	External ❗		NO ❗

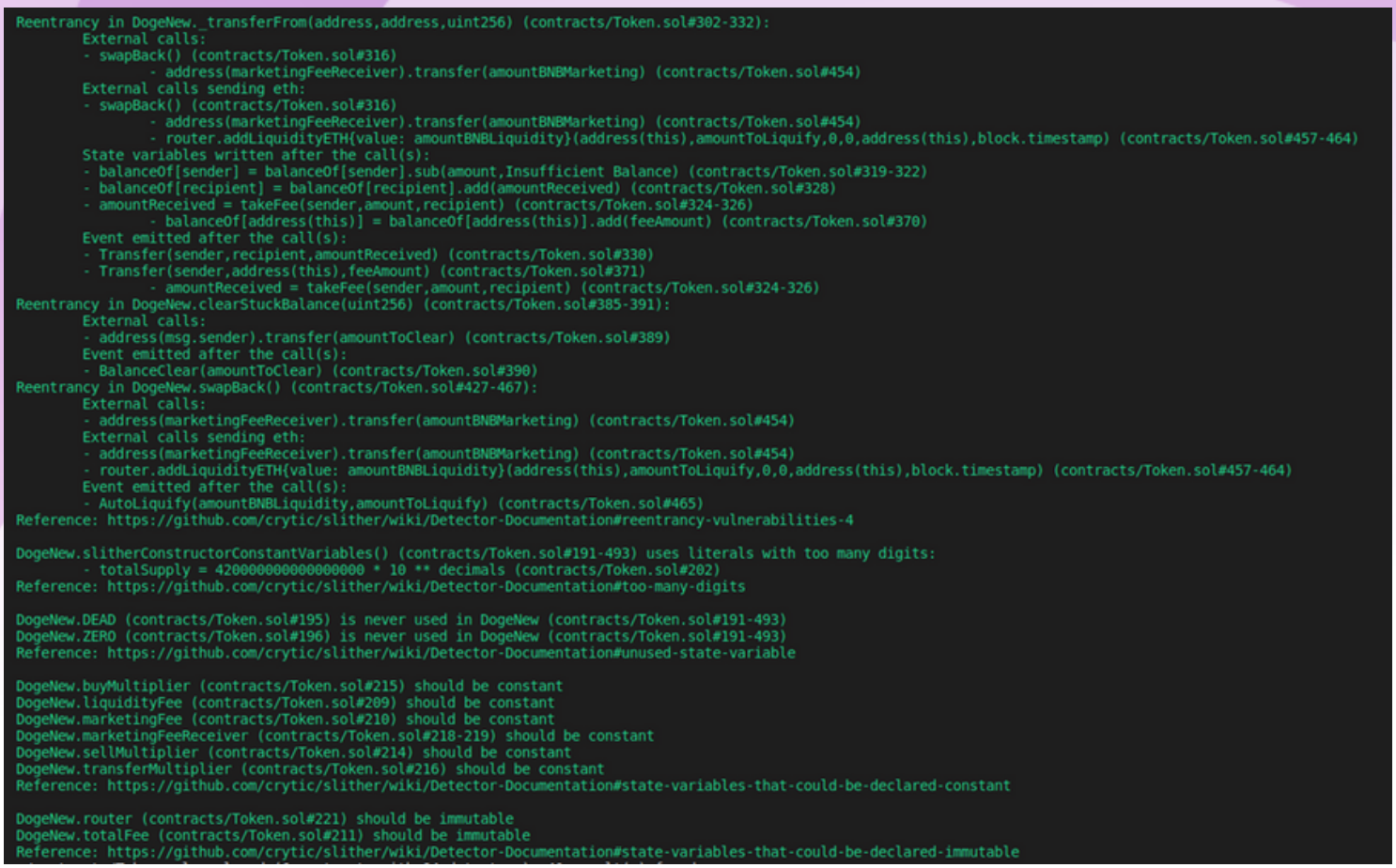
CONTRACT ASSESMENT

└	setExcludedFromFees	Public ❗	🛑	onlyOwner
└	approve	Public ❗	🛑	NO ❗
└	approveMax	External ❗	🛑	NO ❗
└	transfer	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
└	_transferFrom	Internal 🔒	🛑	
└	_basicTransfer	Internal 🔒	🛑	
└	takeFee	Internal 🔒	🛑	
└	shouldSwapBack	Internal 🔒		
└	clearStuckBalance	External ❗	🛑	onlyOwner
└	clearStuckToken	External ❗	🛑	onlyOwner
└	tradingStatus	External ❗	🛑	onlyOwner
└	tradingStatus_launchmode	External ❗	🛑	onlyOwner
└	swapBack	Internal 🔒	🛑	swapping
└	setSwapBackSettings	External ❗	🛑	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS



```
Reentrancy in DogeNew._transferFrom(address,address,uint256) (contracts/Token.sol#302-332):
        External calls:
        - swapBack() (contracts/Token.sol#316)
                - address(marketingFeeReceiver).transfer(amountBNBMarketing) (contracts/Token.sol#454)
        External calls sending eth:
        - swapBack() (contracts/Token.sol#316)
                - address(marketingFeeReceiver).transfer(amountBNBMarketing) (contracts/Token.sol#454)
                - router.addLiquidityETH{value: amountBNBLiquidity}(address(this),amountToLiquify,0,0,address(this),block.timestamp) (contracts/Token.sol#457-464)
        State variables written after the call(s):
        - balanceOf[sender] = balanceOf[sender].sub(amount,Insufficient Balance) (contracts/Token.sol#319-322)
        - balanceOf[recipient] = balanceOf[recipient].add(amountReceived) (contracts/Token.sol#328)
        - amountReceived = takeFee(sender,amount,recipient) (contracts/Token.sol#324-326)
                - balanceOf[address(this)] = balanceOf[address(this)].add(feeAmount) (contracts/Token.sol#370)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amountReceived) (contracts/Token.sol#330)
        - Transfer(sender,address(this),feeAmount) (contracts/Token.sol#371)
                - amountReceived = takeFee(sender,amount,recipient) (contracts/Token.sol#324-326)
Reentrancy in DogeNew.clearStuckBalance(uint256) (contracts/Token.sol#385-391):
        External calls:
        - address(msg.sender).transfer(amountToClear) (contracts/Token.sol#389)
        Event emitted after the call(s):
        - BalanceClear(amountToClear) (contracts/Token.sol#390)
Reentrancy in DogeNew.swapBack() (contracts/Token.sol#427-467):
        External calls:
        - address(marketingFeeReceiver).transfer(amountBNBMarketing) (contracts/Token.sol#454)
        External calls sending eth:
        - address(marketingFeeReceiver).transfer(amountBNBMarketing) (contracts/Token.sol#454)
        - router.addLiquidityETH{value: amountBNBLiquidity}(address(this),amountToLiquify,0,0,address(this),block.timestamp) (contracts/Token.sol#457-464)
        Event emitted after the call(s):
        - AutoLiquify(amountBNBLiquidity,amountToLiquify) (contracts/Token.sol#465)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

DogeNew.slitherConstructorConstantVariables() (contracts/Token.sol#191-493) uses literals with too many digits:
        - totalSupply = 420000000000000000 * 10 ** decimals (contracts/Token.sol#202)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

DogeNew.DEAD (contracts/Token.sol#195) is never used in DogeNew (contracts/Token.sol#191-493)
DogeNew.ZERO (contracts/Token.sol#196) is never used in DogeNew (contracts/Token.sol#191-493)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

DogeNew.buyMultiplier (contracts/Token.sol#215) should be constant
DogeNew.liquidityFee (contracts/Token.sol#209) should be constant
DogeNew.marketingFee (contracts/Token.sol#210) should be constant
DogeNew.marketingFeeReceiver (contracts/Token.sol#218-219) should be constant
DogeNew.sellMultiplier (contracts/Token.sol#214) should be constant
DogeNew.transferMultiplier (contracts/Token.sol#216) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

DogeNew.router (contracts/Token.sol#221) should be immutable
DogeNew.totalFee (contracts/Token.sol#211) 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

The contract is a reliable and rigorously pre-audited token, produced by the PinkSale platform. Given that it is a standard token, we have not conducted any additional tests, as the generation process through PinkSale ensures its security and proper functioning.

MANUAL TESTING

Centralization – Excessive fees

Severity: Informational

Function: setLiquidityFeePercent – setCharityFeePercent - setTaxFeePercent

Lines: 1249 – 1260 - 1241

Status: Not Resolved

The owner has the capability to set buy, sell, and transfer fees up to 25% each, resulting in a potential combined tax of 50% for buy and sell transactions if the maximum limit is set for each fee.

```
function setLiquidityFeePercent(
    uint256 liquidityFeeBps
) external onlyOwner {
    _liquidityFee = liquidityFeeBps;
    require(
        _taxFee + _liquidityFee + _charityFee <= MAX_FEE,
        "Total fee is over 25%"
    );
}

function setCharityFeePercent(uint256 charityFeeBps) external onlyOwner {
    _charityFee = charityFeeBps;
    require(
        _taxFee + _liquidityFee + _charityFee <= MAX_FEE,
        "Total fee is over 25%"
    );
}

function setTaxFeePercent(uint256 taxFeeBps) external onlyOwner {
    _taxFee = taxFeeBps;
    require(
        _taxFee + _liquidityFee + _charityFee <= MAX_FEE,
        "Total fee is over 25%"
    );
}
```

MANUAL TESTING

Recommendation:

order to be compliant with PinkSale's SAFU criteria, the total sum of buy and sell fees should not exceed 25%. It is recommended to modify the code to ensure that this requirement is met.

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