AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

DOGO MOON GO

DATED : 26 May 24'

MANUAL TESTING

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 Open_Trade() external onlyOwner {
    require(!Trade_Open, "TradeOpen");
    feeProcessingEnabled = true;
    Trade_Open = 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.

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.

AUDIT SUMMARY

Project name – DOGO MOON GO

Date: 26 May 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	2	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/0x913e5c15965a0eea76747836c716d918c2e62682#code

Token Information

Token Address:
0x91F2B850e8623812D32423511d6E56c3dC050B86

Name: DOGO MOON GO

Symbol: DGM

Decimals: 18

Network: BscScan

Token Type: BEP-20

Owner: 0x0068D28c589783788090Ff94bc27949F0c3985e4

Deployer: --

Token Supply: 100000000

Checksum: Ae032c616934aeb47e6039f76b20d213

Testnet:
https://testnet.bscscan.com/address/0x913e5c15965a0eea76747836c716d918c2e62682#code

TOKEN OVERVIEW

Buy Fee: 0-15%

Sell Fee: 0-15%

Transfer Fee: 0-0%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: No

Blacklist: No

Other Privileges:

-Whitelist to transfer without enabling trades

- Enabling trades

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

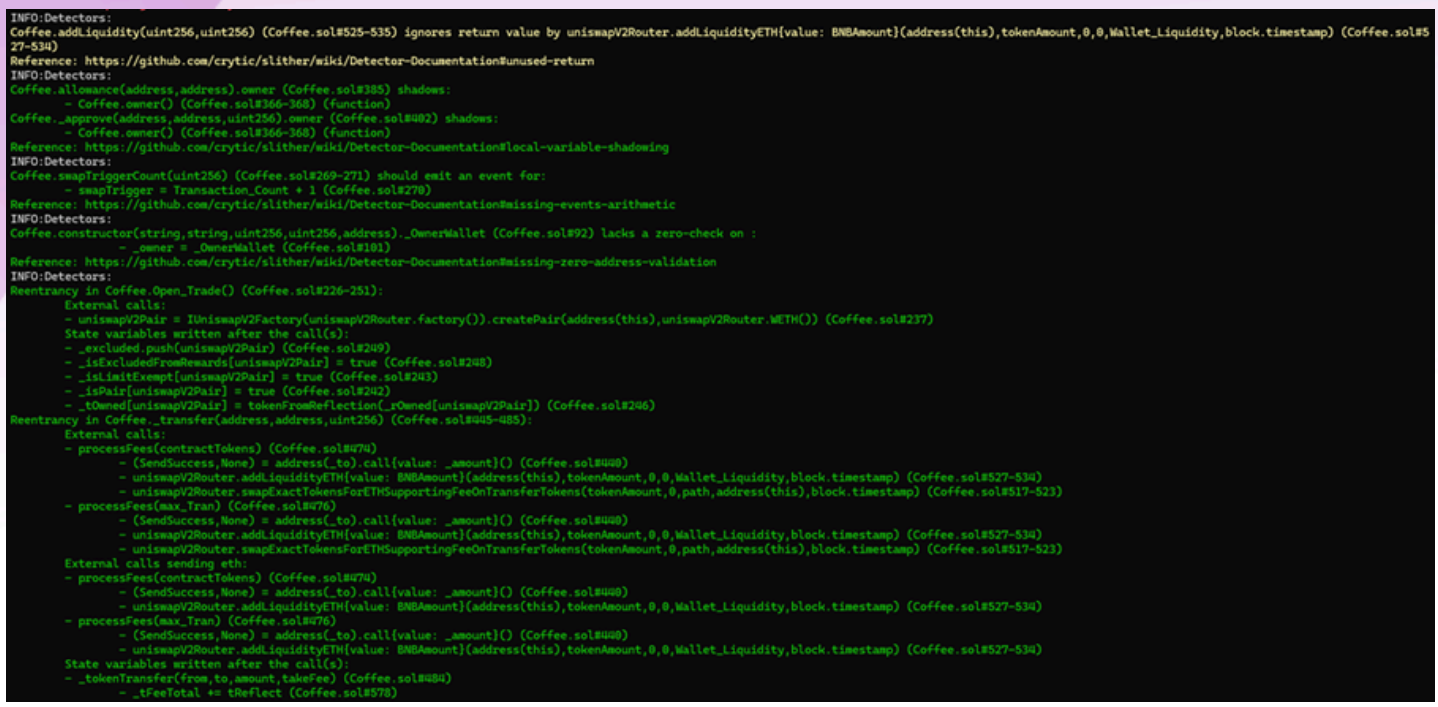
Severity	Found
Critical	0
High-Risk	1
Medium-Risk	2
Low-Risk	2
Gas Optimization / Suggestions	2

INHERITANCE TREE

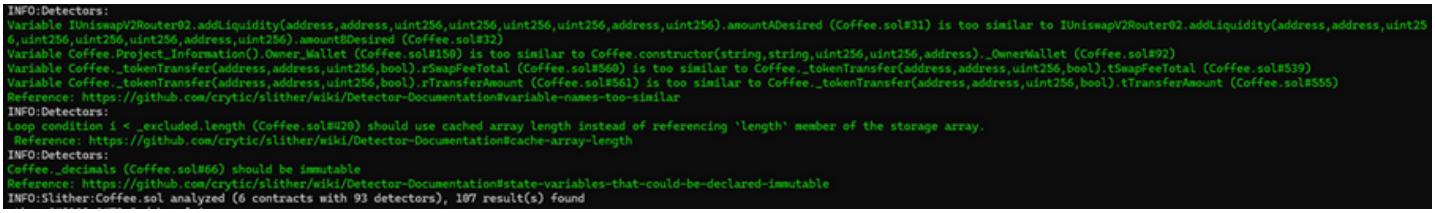
POINTS TO NOTE

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can Enable trading.
- The owner can set the fees not more than 15%.
- The owner can set wallet limits.
- The owner can rescue trapped tokens.
- The owner can update telegram group/Lp locks URL/ Website URL.
- The owner can manually swap tokens.
- The owner can include/exclude wallets from rewards.

STATIC ANALYSIS



```
INFO:Detectors:
Coffee.addLiquidity(uint256,uint256) (Coffee.sol#525-535) ignores return value by uniswapV2Router.addLiquidityETH{value: BNBAmount}(address(this),tokenAmount,0,0,Wallet_Liquidity,block.timestamp) (Coffee.sol#5
27-534)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Coffee.allowance(address,address).owner (Coffee.sol#385) shadows:
        - Coffee.owner() (Coffee.sol#366-368) (function)
Coffee._approve(address,address,uint256).owner (Coffee.sol#402) shadows:
        - Coffee.owner() (Coffee.sol#366-368) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Coffee.swapTriggerCount(uint256) (Coffee.sol#269-271) should emit an event for:
        - swapTrigger = Transaction_Count + 1 (Coffee.sol#270)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
Coffee.constructor(string,string,uint256,uint256,address)._OwnerWallet (Coffee.sol#92) lacks a zero-check on :
                - _owner = _OwnerWallet (Coffee.sol#101)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Coffee.Open_Trade() (Coffee.sol#226-251):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(uniswapV2Router.factory()).createPair(address(this),uniswapV2Router.WETH()) (Coffee.sol#237)
        State variables written after the call(s):
        - _excluded.push(uniswapV2Pair) (Coffee.sol#249)
        - _isExcludedFromRewards[uniswapV2Pair] = true (Coffee.sol#248)
        - _isLimitExempt[uniswapV2Pair] = true (Coffee.sol#243)
        - _isPair[uniswapV2Pair] = true (Coffee.sol#242)
        - _tOwned[uniswapV2Pair] = tokenFromReflection(_rOwned[uniswapV2Pair]) (Coffee.sol#246)
Reentrancy in Coffee._transfer(address,address,uint256) (Coffee.sol#445-485):
        External calls:
        - processFees(contractTokens) (Coffee.sol#474)
                - (SendSuccess,None) = address(_to).call{value: _amount}() (Coffee.sol#440)
                - uniswapV2Router.addLiquidityETH{value: BNBAmount}(address(this),tokenAmount,0,0,Wallet_Liquidity,block.timestamp) (Coffee.sol#527-534)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Coffee.sol#517-523)
        - processFees(max_Tran) (Coffee.sol#476)
                - (SendSuccess,None) = address(_to).call{value: _amount}() (Coffee.sol#440)
                - uniswapV2Router.addLiquidityETH{value: BNBAmount}(address(this),tokenAmount,0,0,Wallet_Liquidity,block.timestamp) (Coffee.sol#527-534)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Coffee.sol#517-523)
        External calls sending eth:
        - processFees(contractTokens) (Coffee.sol#474)
                - (SendSuccess,None) = address(_to).call{value: _amount}() (Coffee.sol#440)
                - uniswapV2Router.addLiquidityETH{value: BNBAmount}(address(this),tokenAmount,0,0,Wallet_Liquidity,block.timestamp) (Coffee.sol#527-534)
        - processFees(max_Tran) (Coffee.sol#476)
                - (SendSuccess,None) = address(_to).call{value: _amount}() (Coffee.sol#440)
                - uniswapV2Router.addLiquidityETH{value: BNBAmount}(address(this),tokenAmount,0,0,Wallet_Liquidity,block.timestamp) (Coffee.sol#527-534)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount,takeFee) (Coffee.sol#484)
                - _tFeeTotal += tReflect (Coffee.sol#578)
```



```
INFO:Detectors:
Variable IUniswapV2Router02.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Coffee.sol#31) is too similar to IUniswapV2Router02.addLiquidity(address,address,uint25
6,uint256,uint256,uint256,address,uint256).amountBDesired (Coffee.sol#32)
Variable Coffee.Project_Information().Owner_Wallet (Coffee.sol#150) is too similar to Coffee.constructor(string,string,uint256,uint256,address)._OwnerWallet (Coffee.sol#92)
Variable Coffee._tokenTransfer(address,address,uint256,bool).rSwapFeeTotal (Coffee.sol#560) is too similar to Coffee._tokenTransfer(address,address,uint256,bool).tSwapFeeTotal (Coffee.sol#539)
Variable Coffee._tokenTransfer(address,address,uint256,bool).rTransferAmount (Coffee.sol#561) is too similar to Coffee._tokenTransfer(address,address,uint256,bool).tTransferAmount (Coffee.sol#555)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
Loop condition i < _excluded.length (Coffee.sol#420) should use cached array length instead of referencing `length` member of the storage array.
 Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cache-array-length
INFO:Detectors:
Coffee._decimals (Coffee.sol#66) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:Coffee.sol analyzed (6 contracts with 93 detectors), 107 result(s) found
```

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- Approve (passed):

https://testnet.bscscan.com/tx/0x85915645313d132199873484c3ae6cf056a9c13bf273d01a3cf18e08123f01be

2- Increase Allowance (passed):

https://testnet.bscscan.com/tx/0x8b8186e1caec2015ea3d8afdf835a35829637c1e31635b5b446c6a8fa62b1b4f

3- Decrease Allowance (passed):

https://testnet.bscscan.com/tx/0xd9675fddde3f01de612915694f83e72d9d97d13590d4f1b06a438bf83e5f38d0

4- Set Fees (passed):

https://testnet.bscscan.com/tx/0x66d18adfbf6bb50762b288a9ec2dfbc4658b082af8630e850e6ff937b9ce295e

5- Update Wallet Liquidity (passed):

https://testnet.bscscan.com/tx/0x72b9045c799c343775c3739c91b4d188c551a9969c8af482033ff5ee3a913ce6

6- Update Wallet Marketing (passed):

https://testnet.bscscan.com/tx/0x75573729ce4195f9ba0e9f0e95ba1d4c7d63c015befb08e03bd6f023c5574315

MANUAL TESTING

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 Open_Trade() external onlyOwner {
    require(!Trade_Open, "TradeOpen");
    feeProcessingEnabled = true;
    Trade_Open = 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.

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.

MANUAL TESTING

Centralization – Liquidity is added to EOA.

Severity: Medium

function: addLiquidity

Status: Open

Overview:

Liquidity is added to EOA. It may be drained by the Wallet_Liquidity.

```
function addLiquidity(uint256 tokenAmount, uint256 BNBAmount)
private {
    _approve(address(this), address(uniswapV2Router), tokenAmount);
    uniswapV2Router.addLiquidityETH{value: BNBAmount}(
      address(this),
      tokenAmount,
      0,
      0,
      Wallet_Liquidity,
      block.timestamp
    );
  }
```

Suggestion:

It is suggested that the address should be a contract address or a dead address.

MANUAL TESTING

Centralization – Missing Require Check.

Severity: Medium

Function:
Update_Wallet_Marketing/Update_Wallet_Liquidity

Status: Open

Overview:
The owner can set any arbitrary address excluding zero address as this is not recommended because if the owner will set 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 Update_Wallet_Liquidity(
   address Liquidity_Collection_Wallet
   ) external onlyOwner {
   require(Liquidity_Collection_Wallet != address(0), "E07");
   Wallet_Liquidity = Liquidity_Collection_Wallet;
 }
function Update_Wallet_Marketing(
   address payable Marketing_Wallet
   ) external onlyOwner {
   require(Marketing_Wallet != address(0), "E08");
   Wallet_Marketing = payable(Marketing_Wallet);
 }
```

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

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 swapTriggerCount(uint256 Transaction_Count) external
onlyOwner {
    swapTrigger = Transaction_Count + 1;
  }
function burnFromTotalSupply(bool true_or_false) external onlyOwner {
    burnFromSupply = true_or_false;
  }
function Update_Wallet_Liquidity(
address Liquidity_Collection_Wallet
    ) external onlyOwner {
require(Liquidity_Collection_Wallet != address(0), "E07");
    Wallet_Liquidity = Liquidity_Collection_Wallet;
  }
function Update_Wallet_Marketing(
address payable Marketing_Wallet
    ) external onlyOwner {
require(Marketing_Wallet != address(0), "E08");
    Wallet_Marketing = payable(Marketing_Wallet);
  }
function Set_Presale_CA(address Presale_CA) external onlyOwner {
    _isExcludedFromFee[Presale_CA] = true;
    _isLimitExempt[Presale_CA] = true;
    _isWhiteListed[Presale_CA] = true;
  }
```

MANUAL TESTING

Centralization – Missing Zero Address

Severity: Low

Subject: Zero Check

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 addLiquidityPair(
  address Wallet_Address,
  bool true_or_false)
  external onlyOwner {
  _isPair[Wallet_Address] = true_or_false;
  _isLimitExempt[Wallet_Address] = true_or_false;
}
```

MANUAL TESTING

Optimization

Severity: Informational

Subject: Floating Pragma.

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 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.

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.

```
event LiquidityAdded(uint256 Tokens_Amount, uint256 BNB_Amount);
```

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