



# Smart Contract Audit

FOR

## Dogu Inu

DATED : 06 June 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 enableTrading() external onlyOwner {  
    require(!tradingEnabled, "STOP! Trading is live");  
    tradingEnabled = true;  
    startTradingBlock = block.number;  
    emit TradingEnabled(startTradingBlock);  
}
```

### 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 - Dogu Inu**

**Date:** 06 June, 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	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- 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/0xde5a929ead3eb204b84212d7e5cd06f8dc74f8b9#code>

---



# Token Information

---

**Token Address:**

0x20b6cf3020F004EBF59F4Fdc20B2CB2021c00Ed9

**Name:** Dogu Inu

**Symbol:** DOGU

**Decimals:** 18

**Network:** BscScan

**Token Type:** BEP-20

**Owner:** 0x79f4BB770B9382eedd83b30e2D2A165483191E85

**Deployer:**

0x79f4BB770B9382eedd83b30e2D2A165483191E85

**Token Supply:** 1000000000

**Checksum:** A2032c616934aeb47e6039f76b20d231

**Testnet:**

<https://testnet.bscscan.com/address/0xde5a929ead3eb204b84212d7e5cd06f8dc74f8b9#code>

---



# TOKEN OVERVIEW

---

---

**Buy Fee:** 5%

---

**Sell Fee:** 5%

---

**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 is exercised 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



**Gasless Send**



Private modifier



Using block.timestamp



Multiple Sends



Re-entrancy



Using Suicide



Tautology or contradiction



Gas Limit and Loops



Timestamp Dependence



Address hardcoded



Revert/require functions



Exception Disorder



Use of tx.origin



Using inline assembly



Integer overflow/underflow



Divide before multiply



Dangerous strict equalities



Missing Zero Address Validation



Using SHA3

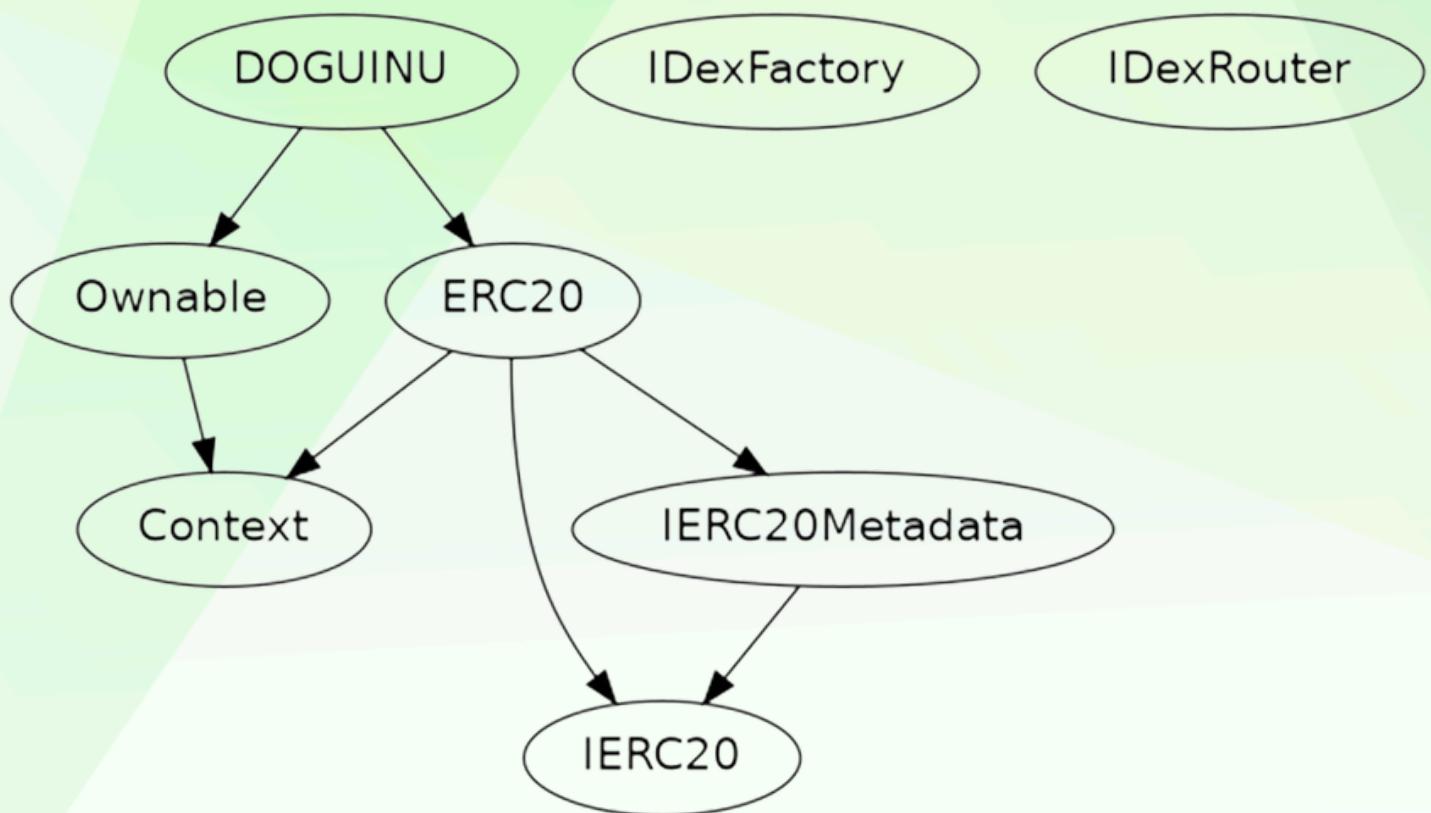


Compiler version not fixed



Using throw

# INHERITANCE TREE





## POINTS TO NOTE

---

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can enable trading.
- The owner can whitelist wallet addresses.



# STATIC ANALYSIS

```
INFO:Detectors:  
Context._msgData() (DOGUINU.sol#6) is never used and should be removed  
ERC20._burn(address,uint256) (DOGUINU.sol#94-105) is never used and should be removed  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code  
INFO:Detectors:  
Pragma version0.8.17 (DOGUINU.sol#2) allows old versions  
solc-0.8.17 is not recommended for deployment  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity  
INFO:Detectors:  
Low level call in DOGUINU.internalSwap() (DOGUINU.sol#247-261):  
    - (success) = marketingWallet.call{value: newBalance}() (DOGUINU.sol#257)  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls  
INFO:Detectors:  
Function IDexRouter.WETH() (DOGUINU.sol#128) is not in mixedCase  
Parameter DOGUINU.setWhitelistStatus(address,bool)._wallet (DOGUINU.sol#179) is not in mixedCase  
Parameter DOGUINU.setWhitelistStatus(address,bool)._status (DOGUINU.sol#179) is not in mixedCase  
Parameter DOGUINU.clearTokens(address)._tokenAddress (DOGUINU.sol#199) is not in mixedCase  
Parameter DOGUINU.checkWhitelist(address)._wallet (DOGUINU.sol#288) is not in mixedCase  
Parameter DOGUINU.swapToBNB(uint256)._amount (DOGUINU.sol#263) is not in mixedCase  
Constant DOGUINU._totalSupply (DOGUINU.sol#134) is not in UPPER_CASE_WITH_UNDERSCORES  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions  
INFO:Detectors:  
Redundant expression "this (DOGUINU.sol#6)" inContext (DOGUINU.sol#4-7)  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements  
INFO:Detectors:  
Reentrancy in DOGUINU.clearBNB() (DOGUINU.sol#190-196):  
    External calls:  
        - address(msg.sender).transfer(amount) (DOGUINU.sol#194)  
    Event emitted after the call(s):  
        - BNBCleared(msg.sender,amount) (DOGUINU.sol#195)  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4  
INFO:Detectors:  
DOGUINU.slitherConstructorVariables() (DOGUINU.sol#133-282) uses literals with too many digits:  
    - swapTokensAtAmount = _totalSupply / 100000 (DOGUINU.sol#138)  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits  
INFO:Detectors:  
DOGUINU.marketingWallet (DOGUINU.sol#137) should be constant  
DOGUINU.swapTokensAtAmount (DOGUINU.sol#138) should be constant  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant  
INFO:Detectors:  
DOGUINU.pairAddress (DOGUINU.sol#136) should be immutable  
DOGUINU.uniswapRouter (DOGUINU.sol#135) should be immutable  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable  
INFO:Slither:DOGUINU.sol analyzed (8 contracts with 93 detectors), 27 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/0x1c85de6fb7f8185051b91255fc4f0c8ddd1f68bfa500c14e88f79936317df9e9>

## 2- Increase Allowance (**passed**):

<https://testnet.bscscan.com/tx/0xcb2a42120956b584b21f2330e6c97954dfa687d839f5ce42bd2053e7413f4b1>

## 3- Decrease Allowance (**passed**):

<https://testnet.bscscan.com/tx/0x58f32506ad524d6f964105153a0f6a3fc97281aae6267b801c920aceebac8b7d>

## 4- Set Whitelist Status (**passed**):

<https://testnet.bscscan.com/tx/0xb2945673e0fdf6cf9edfa30e7d33f940ba44613e22f08901b83cf8ad767d63e>

## 5- Enable Trading (**passed**):

<https://testnet.bscscan.com/tx/0xd9e301aacb22dfc7fb48f1ab16df4636f2b74eafa7ebdbc33e10722eba01560e>

---



# 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	0
◆ Gas Optimization / Suggestions	1



# 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 enableTrading() external onlyOwner {  
    require(!tradingEnabled, "STOP! Trading is live");  
    tradingEnabled = true;  
    startTradingBlock = block.number;  
    emit TradingEnabled(startTradingBlock);  
}
```

**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

---

## 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;}
}
function _burn(address account, uint256 amount) internal virtual {
    require(account != address(0), "STOP: burn from the zero address");
    _beforeTokenTransfer(account, address(0), amount);
    uint256 accountBalance = _balances[account];
    require(accountBalance >= amount, "STOP: burn amount exceeds
balance");
    unchecked {
        _balances[account] = accountBalance - amount;
        _totalSupply -= amount;
    }
    emit Transfer(account, address(0), amount);
    _afterTokenTransfer(account, address(0), 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 specialize 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://t.me/Audit_Ace)**



**[https://twitter.com/auditace\\_](https://twitter.com/auditace_)**



**<https://github.com/Audit-Ace>**

---