Patdit

Request your audit at Patdit.com

Advanced Manual
Smart Contract Audit

November 13, 2023

PatditAudits
t.me/Patdit_tg
Patdit.com

Audit requested by
DOGEMEME
0xc4aea3d5ddeb38151cb115e802b2403731df1ac9

Global Overview

Manual Code Review

In this audit report we will highlight the following issues:

Vulnerability Level	Total Pending	Acknowledged	Resolved
Informational	0	0	0
Low-Risk	0	0	0
Medium-Risk	0	0	0
High-Risk	0	0	0

Centralization Risks

Patdit checked the following privileges:

Contract Privilege	Description
Owner needs to enable trading?	Owner needs to manually enable trading
Owner can mint?	Owner cannot mint new tokens
Owner can blacklist?	Owner cannot blacklist addresses
Owner can set fees?	Owner can set the sell fee to 20%
Owner can exclude from fees?	Owner can exclude from fees
Can be honeypotted?	Owner cannot pause the contract
Owner can set Max TX amount?	Owner can set max transaction amount

More owner privileges are listed later in the report.

Table of Contents

1. Audit Summary
    1.1 Audit scope
    1.2 Tokenomics
    1.3 Source Code
    2. Disclaimer
3. Global Overview
    3.1 Informational issues
    3.2 Low-risk issues
    3.3 Medium-risk issues
    3.4 High-risk issues
4. Vulnerabilities Findings
5. Contract Privileges
    5.1 Maximum Fee Limit Check
    5.2 Contract Pausability Check
    5.3 Max Transaction Amount Check
    5.4 Exclude From Fees Check
    5.5 Ability to Mint Check
    5.6 Ability to Blacklist Check
    5.7 Owner Privileges Check
6. Notes
    6.1 Notes by Patdit
    6.2 Notes by DOGEMEME
7. Contract Snapshot
8. Website Review
9. Certificate of Proof

Audit Summary

Project Name	DOGEMEME
Website	https://dogememecoin.org/
Blockchain	Binance Smart Chain
Smart Contract Language	Solidity
Contract Address	0xc4aea3d5ddeb38151cb115e802b2403731df1ac9
Audit Method	Static Analysis, Manual Review
Date of Audit	13 November 2023

This audit report has been prepared by Patdit's experts at the request of the client. In this audit, the results of the static analysis and the manual code review will be presented. The purpose of the audit is to see if the functions work as intended, and to identify potential security issues within the smart contract.

The information in this report should be used to understand the risks associated with the smart contract. This report can be used as a guide for the development team on how the contract could possibly be improved by remediating the issues that were identified.

Audit Scope

Patdit was commissioned by DOGEMEME to perform an audit based on the following code:

https://bscscan.com/token/0xc4aea3d5ddeb38151cb115e802b2403731df1ac9#code

Note that we only audited the code available to us on this URL at the time of the audit. If the URL is not from any block explorer (main net), it may be subject to change. Always check the contract address on this audit report and compare it to the token you are doing research for.

Audit Method

Patdit's manual smart contract audit is an extensive methodical examination and analysis of the smart contract's code that is used to interact with the blockchain. This process is conducted to discover errors, issues and security vulnerabilities in the code in order to suggest improvements and ways to ‹x them.

Automated Vulnerability Check

Patdit uses software that checks for common vulnerability issues within smart contracts. We use automated tools that scan the contract for security vulnerabilities such as integer-overflow, integer-underflow, out-of-gas-situations, unchecked transfers, etc.

Manual Code Review

Patdit's manual code review involves a human looking at source code, line by line, to ‹nd vulnerabilities. Manual code review helps to clarify the context of coding decisions. Automated tools are faster but they cannot take the developer's intentions and general business logic into consideration.

Used tools

- Slither: Solidity static analysis framework
- Remix: IDE Developer Tool
- CWE: Common Weakness Enumeration
- SWC: Smart Contract Weakness Classi‹cation and Test Cases
- DEX: Testnet Blockchains

Risk Classification

Patdit uses certain vulnerability levels, these indicate how bad a certain issue is. The higher the risk, the more strictly it is recommended to correct the error before using the contract.

Vulnerability Level	Description
Informational	Does not compromise the functionality of the contract in any way
Low-Risk	Won't cause any problems, but can be adjusted for improvement
Medium-Risk	Will likely cause problems and it is recommended to adjust
High-Risk	Will definitely cause problems, this needs to be adjusted

Patdit has four statuses that are used for each risk level. Below we explain them briefly.

Risk Status	Description
Total	Total amount of issues within this category
Pending	Risks that have yet to be addressed by the team
Acknowledged	The team is aware of the risks but does not resolve them
Resolved	The team has resolved and remedied the risk

SWC Attack Analysis

The Smart Contract Weakness Classi‹cation Registry (SWC Registry) is an implementation of the weakness classi‹cation scheme proposed in EIP-1470. It is loosely aligned to the terminologies and structure used in the Common Weakness Enumeration (CWE) while overlaying a wide range of weakness variants that are speci‹c to smart contracts.

ID	Description	Status
SWC-100	Function Default Visibility	Passed
SWC-101	Integer Overflow and Underflow	Passed
SWC-102	Outdated Compiler Version	Passed
SWC-103	Floating Pragma	Passed
SWC-104	Unchecked Call Return Value	Passed
SWC-105	Unprotected Ether Withdrawal	Passed
SWC-106	Unprotected SELFDESTRUCT Instruction	Passed
SWC-107	Reentrancy	Passed
SWC-108	State Variable Default Visibility	Passed
SWC-109	Uninitialized Storage Pointer	Passed
SWC-110	Assert Violation	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed
SWC-112	Delegatecall to Untrusted Callee	Passed
SWC-113	DoS with Failed Call	Passed
SWC-114	Transaction Order Dependence	Passed
SWC-115	Authorization through tx.origin	Passed
SWC-116	Block values as a proxy for time	Passed
SWC-117	Signature Malleability	Passed
SWC-118	Incorrect Constructor Name	Passed
SWC-119	Shadowing State Variables	Passed
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed
SWC-121	Missing Protection against Signature Replay Attacks	Passed
SWC-122	Lack of Proper Signature Verification	Passed
SWC-123	Requirement Violation	Passed
SWC-124	Write to Arbitrary Storage Location	Passed
SWC-125	Incorrect Inheritance Order	Passed
SWC-126	Insufficient Gas Griefing	Passed
SWC-127	Arbitrary Jump with Function Type Variable	Passed
SWC-128	DoS With Block Gas Limit	Passed
SWC-129	Typographical Error	Passed
SWC-130	Right-To-Left-Override control character (U+202E)	Passed
SWC-131	Presence of unused variables	Passed
SWC-132	Unexpected Ether balance	Passed
SWC-133	Hash Collisions With Multiple Variable Length Arguments	Passed
SWC-134	Message call with hardcoded gas amount	Passed
SWC-135	Code With No Effects	Passed
SWC-136	Unencrypted Private Data On-Chain	Passed

Error Code	Description
CS: 016	Initial Supply

Low-Risk: Could be fixed, will not bring problems.

Initial Supply

When the contract is deployed, the contract deployer receives all of the initially created assets. Since the deployer and/or contract owner can distribute tokens without consulting the community, this could be a problem.

Recommendation

Private keys belonging to the employer and/or contract owner should be stored properly. The initial asset allocation procedure should involve consultation with the community.

Error Code	Privilege Check
CS: 017	Reliance on third-parties

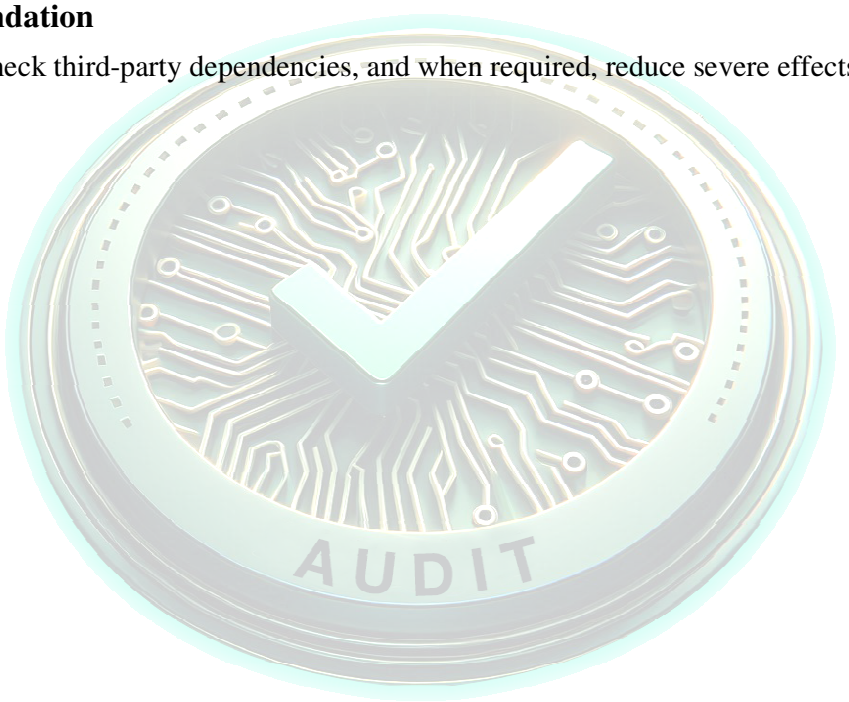
Low-Risk: Could be fixed, will not bring problems.

Reliance on third-parties

Interaction between smart contracts with third-party protocols like Uniswap and Pancake swap. The audit's scope presupposes that third party entities will perform as intended and treats them as if they were black boxes. In the real world, third parties can be hacked and used against you. Additionally, improvements made by third parties may have negative effects, such as higher transaction costs or the deprecation of older routers.

Recommendation

Regularly check third-party dependencies, and when required, reduce severe effects.



Simulated transaction

Test Code	Description
SIM-01	Testing a normal transfer

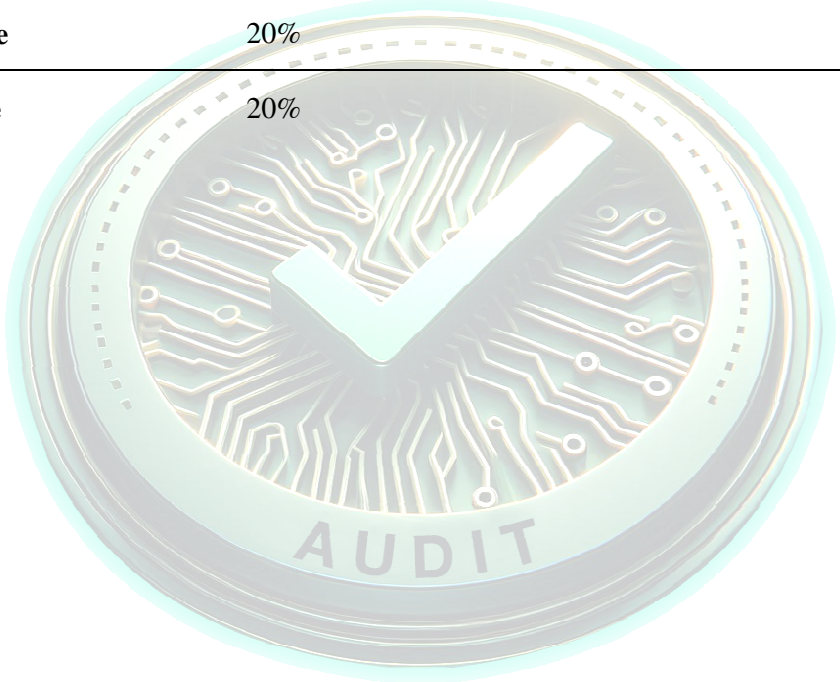
https://testnet.bscscan.com/tx/0x258c6dfd6abe9083ba05bd70ed678b7d5d25e713b8fbe223a56b29c1f2db

Maximum Fee Limit Check

Error Code	Description
CEN-01	Centralization: Operator Fee Manipulation

Patdit tests if the owner of the smart contract can set the transfer, buy or sell fee to 25% or more. It is bad practice to set the fees to 25% or more, because owners can prevent healthy trading or even stop trading when the fees are set too high.

Type of fee	Description
Max transfer fee	0%
Max buy fee	20%
Max sell fee	20%



Contract Honeypot Check

Error Code	Description
CEN-02	Centralization: Operator Pausability

Patdit tests if the owner of the smart contract has the ability to pause the contract. If this is the case, users can no longer interact with the smart contract; users can no longer trade the token.

Privilege Check	Description
Can owner pause the contract?	Owner cannot pause the contract

Max Transaction Amount Check

Error Code	Description
CEN-03	Centralization: Operator Transaction Manipulation

Patdit tests if the owner of the smart contract can set the maximum amount of a transaction. If the transaction exceeds this limit, the transaction will revert. Owners could prevent normal transactions to take place if they abuse this function.

Privilege Check	Description
Can owner set max tx amount?	Owner can set max transaction amount

Function

```
function setMaxTransactionAmounts setMaxTransactionAmounts( (uint256
_maxTransactionAmountBuy uint256 _maxTransactionAmountBuy, , uint256
_maxTransactionAmountSell uint256 _maxTransactionAmountSell) ) e e
require require( (
_maxTransactionAmountBuy > _maxTransactionAmountBuy >; ;= = ( (totalSupply totalSupply( ()
) / / ( (10 10 * ** * decimals decimals( () )) )) ) / / 1_000 1_000 & &; ;& &; ;
_maxTransactionAmountSell > _maxTransactionAmountSell >; ;= = ( (totalSupply totalSupply(
() ) / / ( (10 10 * ** * decimals decimals( () )) )) ) / / 1_000 1_000, ,
"Max Transaction limis cannot be lower than 0.1% of total supply" "Max Transaction limis
cannot be lower than 0.1% of total supply" ) ); ;
maxTransactionAmountBuy maxTransactionAmountBuy = = _maxTransactionAmountBuy
_maxTransactionAmountBuy * * ( (10 10 * ** * decimals decimals( () )) ); ;
maxTransactionAmountSell maxTransactionAmountSell = = _maxTransactionAmountSell
_maxTransactionAmountSell * * ( (10 10 * ** * decimals decimals( () )) ); ;
emit emit MaxTransactionLimitAmountChanged MaxTransactionLimitAmountChanged(
(maxTransactionAmountBuy maxTransactionAmountBuy, , maxTransactionAmountSell
maxTransactionAmountSell) ); ; } }
```

Exclude From Fees Check

Error Code	Description
CEN-04	Centralization: Operator Exclusion

Patdit tests if the owner of the smart contract can exclude addresses from paying tax fees. If the owner of the smart contract can exclude from fees, they could set high tax fees and exclude themselves from fees and bene‹t from 0% trading fees. However, some smart contracts require this function to exclude routers, dex, cex or other contracts / wallets from fees.

Privilege Check	Description
Can owner exclude from fees?	Owner can exclude from fees

Function

```
function function excludeFromFees excludeFromFees( (address account address account, , bool
excluded bool excluded) ) external external onlyOwner onlyOwner{ {
require require( (_isExcludedFromFees _isExcludedFromFees[ [account account] ] != !=
excluded excluded, ,"Account is already the value of 'excluded'" "Account is already the
value of 'excluded'") ); ;
_isExcludedFromFees _isExcludedFromFees[ [account account] ] = = excluded excluded; ;
emit emit ExcludeFromFees ExcludeFromFees( (account account, , excluded excluded) ); ; } }
```

Ability To Mint Check

Error Code	Description
CEN-05	Centralization: Operator Increase Supply

Patdit tests if the owner of the smart contract can mint new tokens. If the contract contains a mint function, we refer to the token's total supply as non-‹xed, allowing the token owner to "mint" more tokens whenever they want.

A mint function in the smart contract allows minting tokens at a later stage. A method to disable minting can also be added to stop the minting process irreversibly.

Minting tokens is done by sending a transaction that creates new tokens inside of the token smart contract. With the help of the smart contract function, an unlimited number of tokens can be created without spending additional energy or money.

Privilege Check	Description
Can owner mint?	Owner cannot mint new tokens

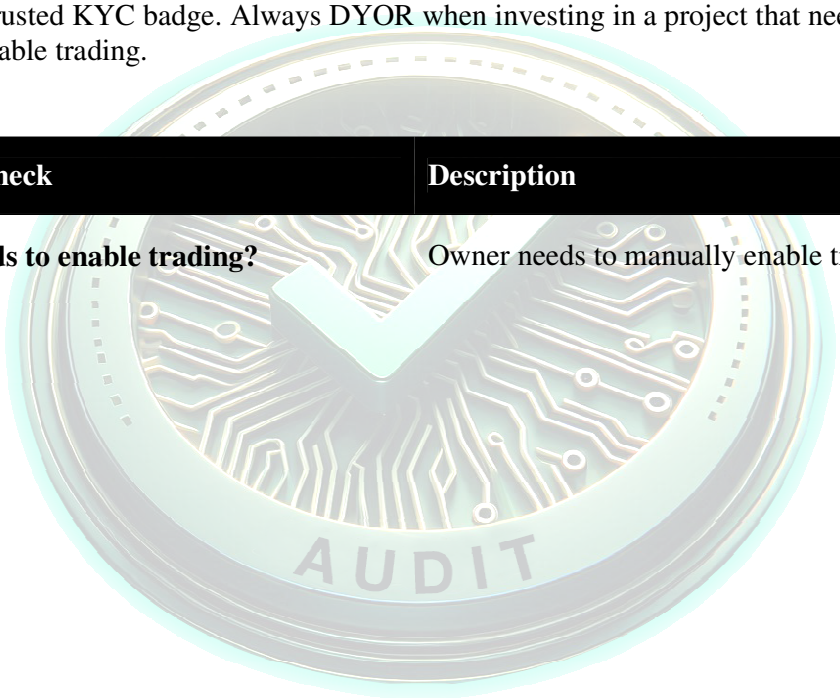
Enable Trading

Error Code	Description
CEN-06	Centralization: Operator enable trading

Patdit tests if the owner of the smart contract needs to manually enable trading before everyone can buy & sell. If the owner needs to manually enable trading, this poses a high centralization risk.

If the owner needs to manually enable trading, make sure to check if the project has a SAFU badge or a trusted KYC badge. Always DYOR when investing in a project that needs to manually enable trading.

Privilege Check	Description
Owner needs to enable trading?	Owner needs to manually enable trading



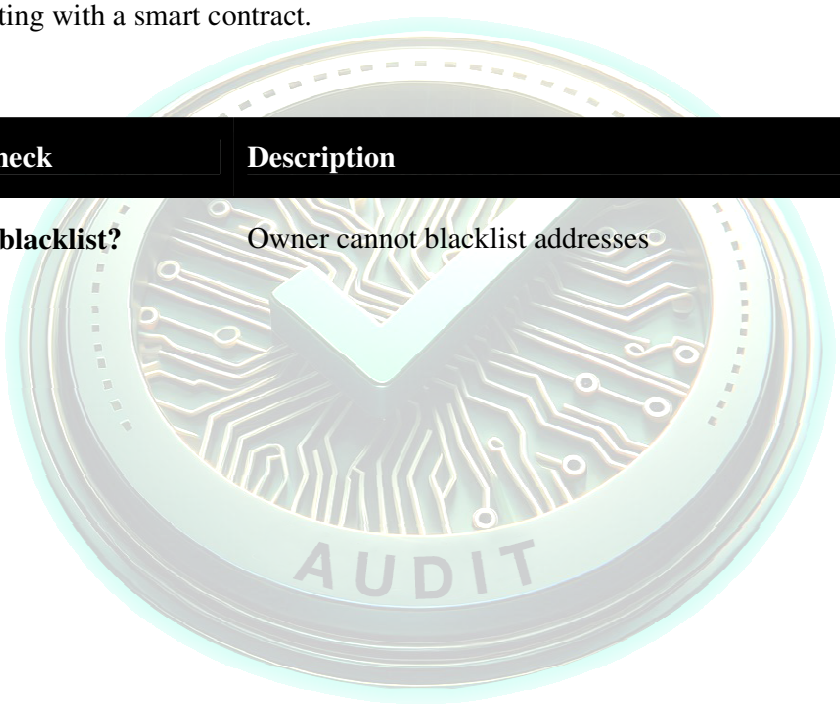
Ability To Blacklist Check

Error Code	Description
CEN-07	Centralization: Operator Dissalows Wallets

Patdit tests if the owner of the smart contract can blacklist accounts from interacting with the smart contract. Blacklisting methods allow the contract owner to enter wallet addresses which are not allowed to interact with the smart contract.

This method can be abused by token owners to prevent certain / all holders from trading the token. However, blacklists might be good for tokens that want to rule out certain addresses from interacting with a smart contract.

Privilege Check	Description
Can owner blacklist?	Owner cannot blacklist addresses



Other Owner Privileges Check

Error Code	Description
CEN-100	Centralization: Operator Privileges

Patdit lists all important contract methods which the owner can interact with.

Owner can withdraw tokens from the contract address

Owner can exclude addresses from max wallet amount

Owner can exclude addresses from max transaction amount

Notes

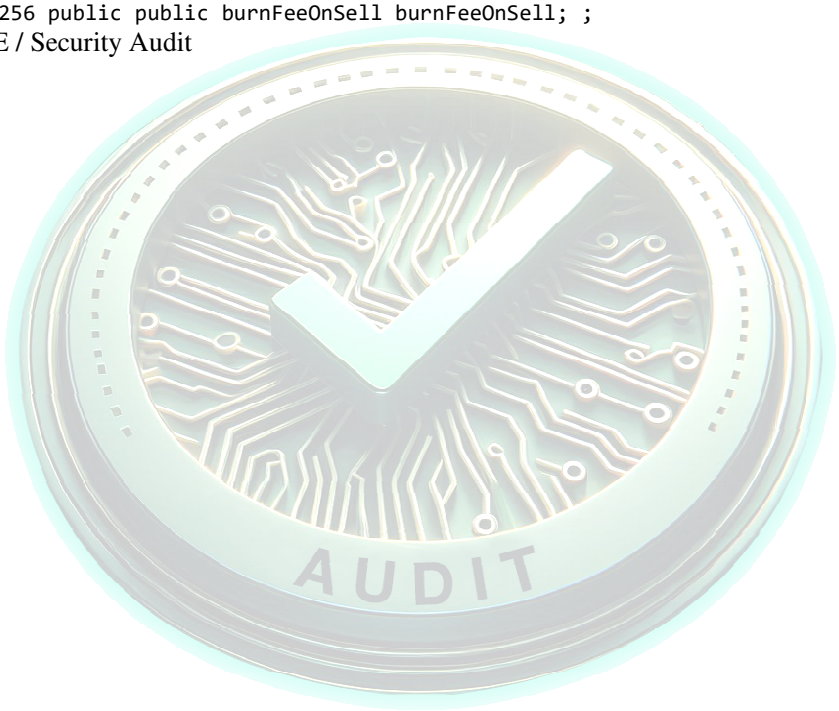
Notes by KittyGodZii
No notes provided by the team.

Notes by Patdit
No notes provided by Patdit

Contract Snapshot

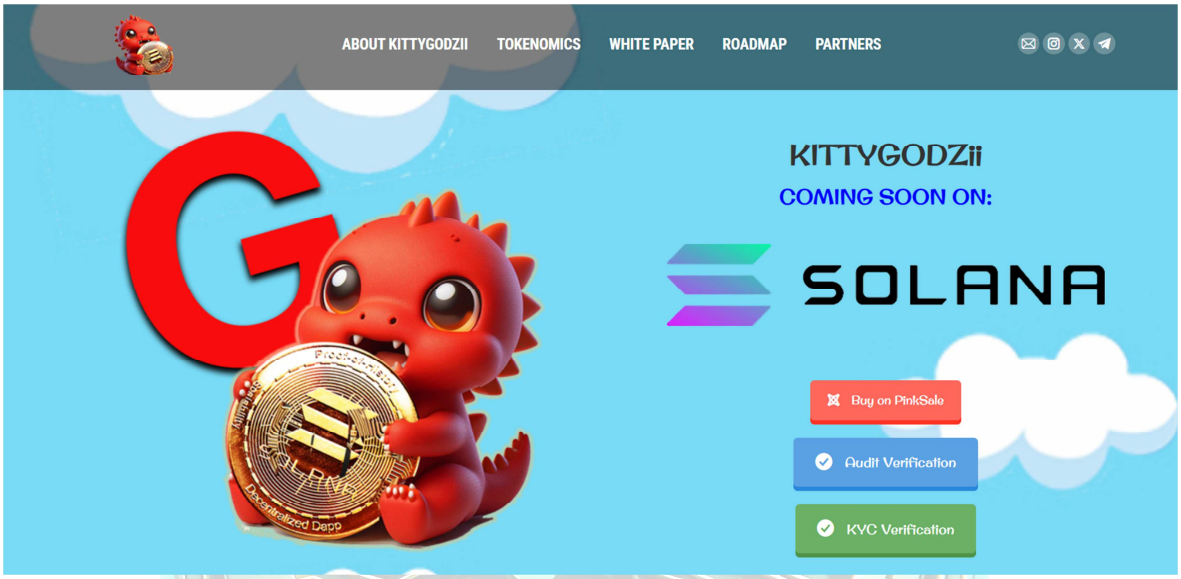
This is how the constructor of the contract looked at the time of auditing the smart contract.

```
contract DOGEMEME contract DOGEMEME is is ERC20 ERC20, , Ownable Ownable { {
using Address using Address for for address payable address payable; ;
IUniswapV2Router02 IUniswapV2Router02 public public uniswapV2Router uniswapV2Router; ;
address address public public uniswapV2Pair uniswapV2Pair; ;
mapping mapping ( (address address = => >; ; bool bool) ) private private
_isExcludedFromFees _isExcludedFromFees; ; uint256 uint256 public public liquidityFeeOnBuy
liquidityFeeOnBuy; ;
uint256 uint256 public public liquidityFeeOnSell liquidityFeeOnSell; ;
uint256 uint256 public public marketingFeeOnBuy marketingFeeOnBuy; ;
uint256 uint256 public public marketingFeeOnSell marketingFeeOnSell; ;
uint256 uint256 public public developerFeeOnBuy developerFeeOnBuy; ;
uint256 uint256 public public developerFeeOnSell developerFeeOnSell; ;
uint256 uint256 public public burnFeeOnBuy burnFeeOnBuy; ;
uint256 uint256 public public burnFeeOnSell burnFeeOnSell; ;
```

Website Review

Patdit checks the website completely manually and looks for visual, technical and textual errors. We also look at the security, speed and accessibility of the website. In short, a complete check to see if the website meets the current standard of the web development industry.



Type of check	Description
Mobile friendly?	The website is mobile friendly
Contains jQuery errors?	The website does not contain jQuery errors
Is SSL secured?	The website is SSL secured
Contains spelling errors?	The website does not contain spelling errors

Certificate of Proof

Not KYC verified by Patdit

GodZii

Audited by Patdit.com

Date: 13 November 2023

Advanced Manual Smart Contract Audit

Disclaimer

This audit report has been prepared by Patdit's experts at the request of the client. In this audit, the results of the static analysis and the manual code review will be presented. The purpose of the audit is to see if the functions work as intended, and to identify potential security issues within the smart contract.

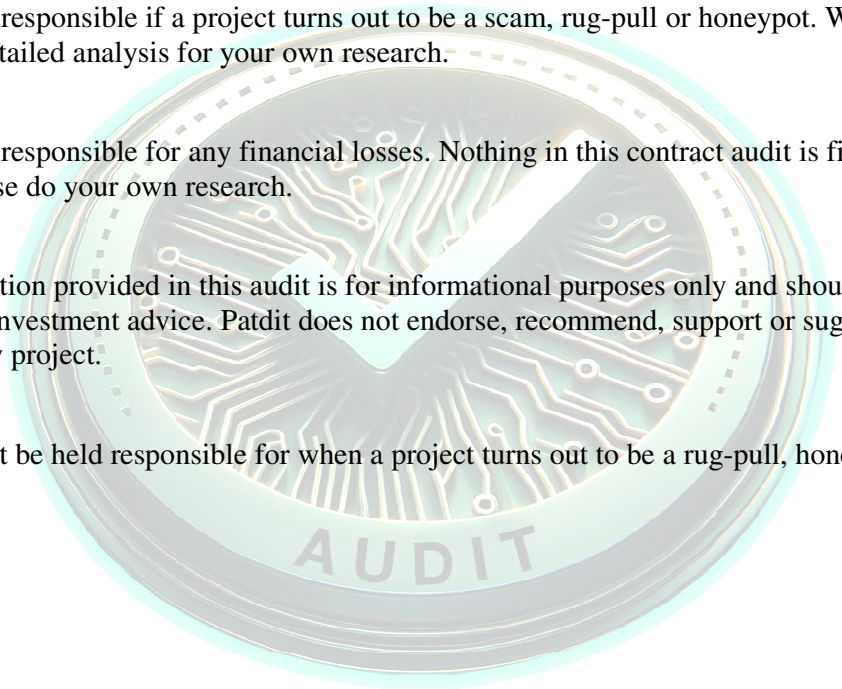
The information in this report should be used to understand the risks associated with the smart contract. This report can be used as a guide for the development team on how the contract could possibly be improved by remediating the issues that were identified.

Patdit is not responsible if a project turns out to be a scam, rug-pull or honeypot. We only provide a detailed analysis for your own research.

Patdit is not responsible for any financial losses. Nothing in this contract audit is financial advice, please do your own research.

The information provided in this audit is for informational purposes only and should not be considered investment advice. Patdit does not endorse, recommend, support or suggest to invest in any project.

Patdit cannot be held responsible for when a project turns out to be a rug-pull, honeypot or scam.



Patdit

Patdit.com

End of report
Smart Contract Audit

Patdit Audits
◊ info@Patdit.com
← Patdit.com
Request your smart contract audit / KYC
t.me/Patdit_tg