KISHIELD

Security Audit

MEG Token

May 24, 2022

MOVE2E

*Audit Not Passed

Table of Contents

1 Audit Summary

2 Project Overview

2.1 Token Summary

2.2 Main Contract Assessed

3 Smart Contract Vulnerability Checks

4 Contract Ownership

4.1 Priviliged Functions

5 Important Notes To The Users

6 Findings Summary

6.1 Classification of Issues

6.1 Findings Table

01 Extra Tax

02 Check swapAndLiquify logic

03 Variable Initialization

04 Incorrect Tax Logic

05 Unprotected BNB withdrawal

06 Variables could be declared as constant

07 Public function that could be declared external

08 Missing events arithmetic

09 Too many digits

010 Unused Variable

011 Possible to gain ownership after renouncing the contract ownership

7 Statistics

7.1 Liquidity

7.2 Token Holders

7.3 Liquidity Holders

8 Liquidity Ownership

9 Disclaimer

Audit Summary

This report has been prepared for MEG Token on the Binance Chain network. KISHIELD provides both client-centered and user-centered examination of the smart contracts and their current status when applicable. This report represents the security assessment made to find issues and vulnerabilities on the source code along with the current liquidity and token holder statistics of the protocol.

A comprehensive examination has been performed, utilizing Cross Referencing, Static Analysis, In-House Security Tools, and line-by-line Manual Review.

The auditing process pays special attention to the following considerations:

- Ensuring contract logic meets the specifications and intentions of the client without exposing the user's funds to risk.
- Testing the smart contracts against both common and uncommon attack vectors.
- Inspecting liquidity and holders statistics to inform the current status to both users and client when applicable.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Verifying contract functions that allow trusted and/or untrusted actors to mint, lock, pause, and transfer assets.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Project Overview

Token Summary

Parameter	Result
Address	0x4d7EDfB32D7f5F0429b879b434925771D712fF4A
Name	MEG
Token Tracker	MEG (MEG)
Decimals	18
Supply	1,000,000,000,000
Platform	Binance Chain
compiler	v0.6.12+commit.27d51765
Optimization	Yes with 200 runs
LicenseType	None
Language	Solidity
Codebase	https://bscscan.com/address/0x4d7EDfB32D7f5F0429b879b434925771D712fF4A#code
Url	https://move2e.io

Main Contract Assessed

Name	Contract	Live
MEG	0x4d7EDfB32D7f5F0429b879b434925771D712fF4A	Yes

Smart Contract Vulnerability Checks

Vulnerability	Automatic Scan	Manual Scan	Result
Unencrypted Private Data On-Chain	Complete	Complete	✓ Low / No Risk
Code With No Effects	Complete	Complete	✓ Low / No Risk
Message call with hardcoded gas amount	Complete	Complete	✓ Low / No Risk
Hash Collisions With Multiple Variable Length Arguments	Complete	Complete	✓ Low / No Risk
Unexpected Ether balance	Complete	Complete	✓ Low / No Risk
Presence of unused variables	Complete	Complete	● Low
Right-To-Left-Override control character (U+202E)	Complete	Complete	✓ Low / No Risk
Typographical Error	Complete	Complete	✓ Low / No Risk
DoS With Block Gas Limit	Complete	Complete	✓ Low / No Risk
Arbitrary Jump with Function Type Variable	Complete	Complete	✓ Low / No Risk
Insufficient Gas Griefing	Complete	Complete	✓ Low / No Risk
Incorrect Inheritance Order	Complete	Complete	✓ Low / No Risk
Write to Arbitrary Storage Location	Complete	Complete	✓ Low / No Risk
Requirement Violation	Complete	Complete	✓ Low / No Risk
Missing Protection against Signature Replay Attacks	Complete	Complete	✓ Low / No Risk
Weak Sources of Randomness from Chain Attributes	Complete	Complete	✓ Low / No Risk

Vulnerability	Automatic Scan	Manual Scan	Result
Authorization through tx.origin	Complete	Complete	✓ Low / No Risk
Delegatecall to Untrusted Callee	Complete	Complete	✓ Low / No Risk
Use of Deprecated Solidity Functions	Complete	Complete	✓ Low / No Risk
Assert Violation	Complete	Complete	✓ Low / No Risk
Reentrancy	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
Unprotected Ether Withdrawal	Complete	Complete	● Low
Unchecked Call Return Value	Complete	Complete	✓ Low / No Risk
Outdated Compiler Version	Complete	Complete	✓ Low / No Risk
Integer Overflow and Underflow	Complete	Complete	✓ Low / No Risk
Function Default Visibility	Complete	Complete	✓ Low / No Risk

Contract Ownership

The contract ownership of MEG is not currently renounced. The ownership of the contract grants special powers to the protocol creators, making them the sole addresses that can call sensible ownable functions that may alter the state of the protocol.

The current owner is the address 0x4cF8B864e69D653097B687a4F799A616Ff875425 which can be viewed from:
HERE

The owner wallet has the power to call the functions displayed on the priviliged functions chart below, if the owner wallet is compromised this privileges could be exploited.

We recommend the team to renounce ownership at the right timing if possible, or gradually migrate to a timelock with governing functionalities in respect of transparency and safety considerations.

Important Notes To The Users:

- Project owners refused to renounce ownership of the contract, once the presale is done and liquidity is added they will renounce ownership over the contract. Please read the following points carefully.
- The owner cannot blacklist wallets.
- Once the owner renounces ownership of the contract, none of the following are applicable.
- Owner can pause trading by setting maxTxAmount to 0.
- Owner can change taxes fees to 100%.
- Owner can add/remove wallets from fee exemption and rewards.
- Owner can change the maxTxAmount without any constrains (can set it to 0 tokens).
- Owner can enable/disable the SwapAndLiquify mechanism and change the numTokensSellToAddToLiquidity.
- Owner can regain ownership even after renouncing to it by locking the ownership beforehand, renounce, and then unlock the ownership.
- No high-risk Exploits/Vulnerabilities Were Found in token Source Code other than owner privileges.

Audit Not Passed

✗

Technical Findings Summary

Classification of Issues

Severity	Description
🔴 High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency
🟠 Medium	Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
🟡 Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
🟣 Info	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

Findings

Severity	Found
🔴 High	1
🟠 Medium	2
🟡 Low	1
🟣 Info	7
Total	11

Findings

Extra Tax

ID	Severity	Contract	Function
01	🟠 Medium	MEG	Function _transfer(), _tokenTransfer, _transferStandard, _getValues, calculateTaxFee

Description

Extra tax for non-excluded users. If the contract owner sets the _taxFee or _liquidityFee to any value other than 0 non-fee-excluded users would pay extra or "double tax". Tax is taken in _transfer and send to external wallets; _transferStandard computes another tax "_taxFee" and "_liquidityFee" while these values remain 0 no extra fee will be charged.

Recommendation

We recommend deleting setTaxFeePercent and setLiquidityFeePercent and making both _taxFee, _liquidityFee constant. .

Check swapAndLiquify logic

ID	Severity	Contract	Function
02	🟣 Informational	MEG	Function swapAndLiquify()

Description

Current logic sends tax fees _buyFee, _sellFee to FEE_ADDRESS, FEE_ADDRESS_2 thus the contract never gains tokens; thus the swap will never be trigger unless the contract is sent tokens externally

Recommendation

We recommend sending a portion of _buyFee, _sellFee to the contract.

Variable Initialization

ID	Severity	Contract	Function
03	🟡 Low	MEG	Variables _maxTxAmount, _maxWalletAmount

Description

Variables are set to the total supply of the token. _maxWalletAmount is set to a value too high making the require statement on _transfer() meaningless. Owner cannot change _maxWalletAmount. setMaxTxPercent can set the _maxTxAmount to 0 making the contract into a honeypot.

Recommendation

We recommend adding a require statement to stop the owner of setting the _maxTxAmount lower than 0.1% and creating a function to set _maxWalletAmount.

Incorrect Tax Logic

ID	Severity	Contract	Function
04	🔴 High	MEG	Function _transfer()

Description

Contract uses the variable "WPOOL" to apply buy+sell taxes or only buy tax. We assume the correct usage of this mechanism is to check "to == pairContract". WPOOL is set to a external wallet.

Recommendation

We recommend delete the WPOOL variable an use uniswapV2Router instead. In case this is a unique feature to the protocol do nothing.

Unprotected BNB withdrawal

ID	Severity	Contract	Function
05	● Informational	MEG	Function transferWBNB()

Description

Anyone can call the function and withdraw BNB from the contract

Recommendation

We recommend adding an onlyOwner modifier to the function.

Variables could be declared as constant

ID	Severity	Contract	Function
06	● Informational	MEG	Variables FEE_ADDRESS, FEE_ADDRESS_2, ROUTER_ADDRES,SWBNB_ADDRESS, WETH, _buyFee, _decimals, _feeToDiv, _name, _sellFee, _symbol, _tokenToBNBFee

Description

Gas Optimization. Variables that are never changed could be declared as constant.

Recommendation

We recommend declaring those variables as constant.

Public function that could be declared external

ID	Severity	Contract	Function
07	● Informational	MEG	Functions externalrenounceOwnership, transferOwnership, geUnlockTime, lock, unlock, addresses, isExcludedFromReward, totalFees, deliver, reflectionFromToken, excludeFromReward, excludeFromFee, includeInFee, setSwapAndLiquifyEnabled, isExcludedFromFee.

Description

Gas Optimization. Public function that could be declared external

Recommendation

Public functions that are never called by the contract should be declared external to save gas.

Missing events arithmetic

ID	Severity	Contract	Function
08	● Informational	MEG	Missing events for setTaxFeePercent, setLiquidityFeePercent, setMaxTxPercent, changeNumTokensSellToAddToLiquidity

Description

Functions that change critical arithmetic parameters should emit an event.

Recommendation

Emit corresponding events for critical parameter changes.

Too many digits

ID	Severity	Contract	Function
09	● Informational	MEG	Variable _tTotal, numTokensSellToAddToLiquidity

Description

Literals with many digits are difficult to read and review.

Recommendation

Make use of scientific notation, use underscores, and/or use ether suffix.

Unused Variable

ID	Severity	Contract	Function
010	● Informational	MEG	Variable _feeToDiv, _MKTshare

Description

Variables are never used in the contract logic in a meaningful way.

Recommendation

We recommend deleting this variable.

Possible to gain ownership after renouncing the contract ownership

ID	Severity	Contract	Function
011	🟠 Medium	MEG	function lock(uint256 time) public virtual onlyOwner && function unlock()

Description

Logical Issue, Privilege. An owner can regain ownership even after renouncing to it. If an owner calls the lock function his address is saved in the _previousOwner variable. Then, if after renouncing ownership the _previousOwner calls the unlock function the owner of the contract is set to address of _previousOwner.

Recommendation

We advise updating/removing lock and unlock functions in the contract as this functions logic voids the point of renouncing ownership.

Priviliged Functions (onlyOwner & Others)

Function Name	Parameters	Visibility
renounceOwnership	none	public
transferOwnership	address newOwner	public
lock	uint256 time	public
excludeFromReward	address account	public
includeInReward	address account	external
excludeFromFee	address account	public
includeInFee	address account	public
setTaxFeePercent	uint256 taxFee	external
setLiquidityFeePercent	uint256 liquidityFee	external
setMaxTxPercent	uint256 maxTxPercent	external
setMarketingFeePercent	uint256 marketingFee	external
setMarketingWallet	address _add	external
setPoolWallet	address _add	external
setSwapAndLiquifyEnabled	bool _enabled	public
changeNumTokensSellToAddToLiquidity	uint256 _numTokensSellToAddToLiquidity	external

Statistics

Liquidity Info

Parameter	Result
Pair Address	0x410b11d9B9b332D5F14b793abD22124977D160f2
MEG Reserves	0.00 MEG
FGD Reserves	0.00 FGD
Liquidity Value	$0 USD

Token (MEG) Holders Info

Parameter	Result
MEG Percentage Burnt	0.00%
MEG Amount Burnt	0 MEG
Top 10 Percentage Own	100.00%
Top 10 Amount Owned	1,000,000,000,000 MEG
Top 10 Aprox Value	$NaN USD

LP (MEG/FGD) Holders Info

Parameter	Result
MEG/FGD % Burnt	0.00%
MEG/FGD Amount Burnt	0 MEG/FGD
Top 10 Percentage Owned	0.00%
Top 10 Amount Owned	0 MEG/FGD
Locked Tokens Percentage	0.00%
Locked Tokens Amount	0 MEG/FGD

* All the data diplayed above was taken on-chain at block 18060368
* The tokens on industry-standard burn wallets are not included on the top 10 wallets calculations

Liquidity Ownership

The token does not have liquidity at the moment of the audit, block 18060368

KISHIΞLD

Disclaimer

KISHIELD has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided ‘as is’, and KISHIELD is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will KISHIELD or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

The assessment services provided by KISHIELD is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.