AUDIT REPORT

SecureWise

MAGIC EGG (MEGG)

Quick Result

Quick Result	Status
Owner can mint new token?	Not Detected
Owner can update tax over 25% ?	Not Detected
Owner can pause trade ?	Not Detected
Owner can enable trading ?	Not Detected
Owner can add Blacklist ?	Not Detected
Owner can set Max Tx ?	Not Detected
Owner can set Max Wallet Amount ?	Not Detected
KYC ?	No KYC

Page 6,10 for more details

Magic Egg (MEGG) as PASSED the smart contract audit.

Findings

Risk Classification	Description
High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, of the contract and its functions. Must be fixed as soon as possible.
Medium	Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Must be fxed as soon as possible.
Low	Effects are minimal in isolation and do not pose a signifcant danger to the project or its users. Issues under this classifcation are recommended to be fixed nonetheless.
Informational	A vulnerability that have informational character but is not effecting any of the code

Severity	Found	Pending	Resolved
High	0	0	0
Medium	0	0	0
Low	0	0	0
Informational	0	4	0
Total	0	4	0

Contents

Overview

Token Name: Magic Egg (**MEGG**)

Language: Solidity

Contract Address: 0x8258c4b51a73fC114d5ACF04F429E25695fd5488

Network: Binance Smart Chain

Total Supply: 42000000000000000

KYC: No KYC

Website: https://luigicoin.vip

Twitter: https://twitter.com/LuigiInuBSC

Telegram: https://t.me/LuigiInuBSC

Report Date: July 11, 2023

Testnet:

https://testnet.bscscan.com/address/0xC4666D17B68367F42EB6a27B5b832A678a83d651

Auditing Approach and Methodologies

SecureWise has performed starting with analyzing the code, issues, code quality, and libraries. Reviewed line-by-line by our team. Finding any potential issue like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.

Methodology

- Understanding the size, scope and functionality of your project's source code
- Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Testing and automated analysis of the Smart Contract to determine proper logic has been followed throughout the whole process
- Deploying the code on testnet using multiple live test
- Analyzing a program to determine the specific input that causes different parts of a program to execute its functions.
- Checking whether all the libraries used in the code are on the latest version.

Goals

Smart Contract System is secure, resilient and working according to the specifications and without any vulnerabilities.

Risk Classification

High: Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, of the contract and its functions. Must be fixed as soon as possible.

Medium: Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Must be fxed as soon as possible.

Low: Effects are minimal in isolation and do not pose a signifcant danger to the project or its users. Issues under this classifcation are recommended to be fixed nonetheless.

Informational: A vulnerability that have informational character but is not effecting any of the code

Findings Summary

SecureWise has applied the automated and manual analysis of Smart Contract and were reviewed for common contract vulnerabilities and centralized exploits

Findings

	Owner has the authority to exclude account from dividend
	Owner has the authority to exclude account from fees
	Owner has authority to change swap token amount
	Owner has authority to claim stuck token

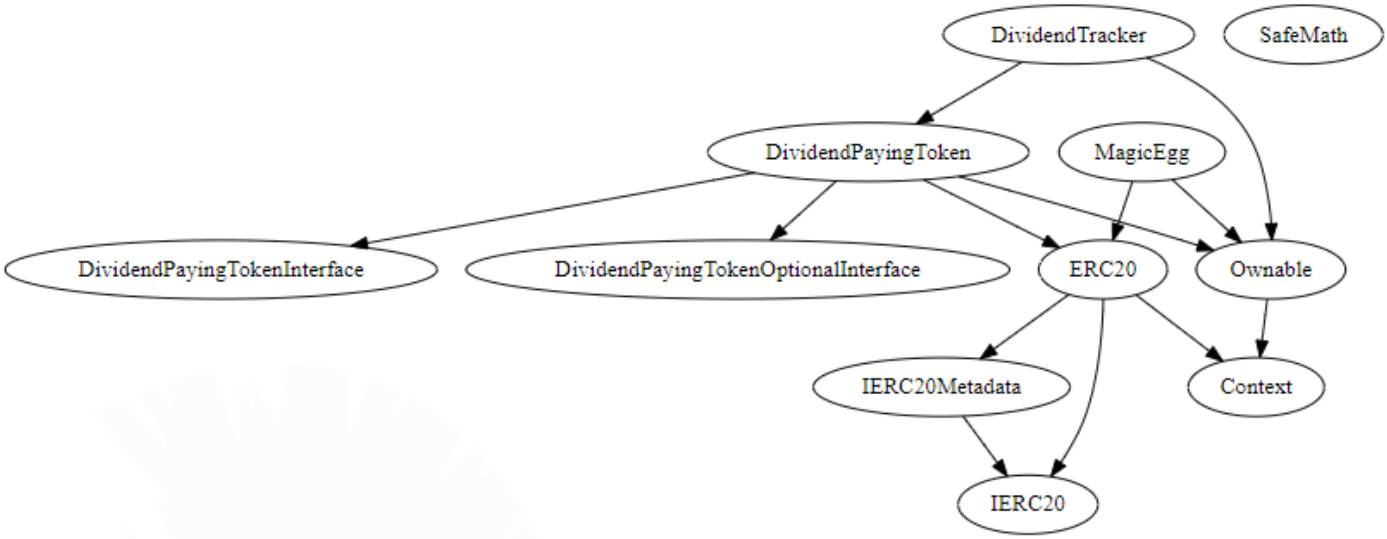
Page 10 for more details

Magic Egg (MEGG) as PASSED the smart contract audit.

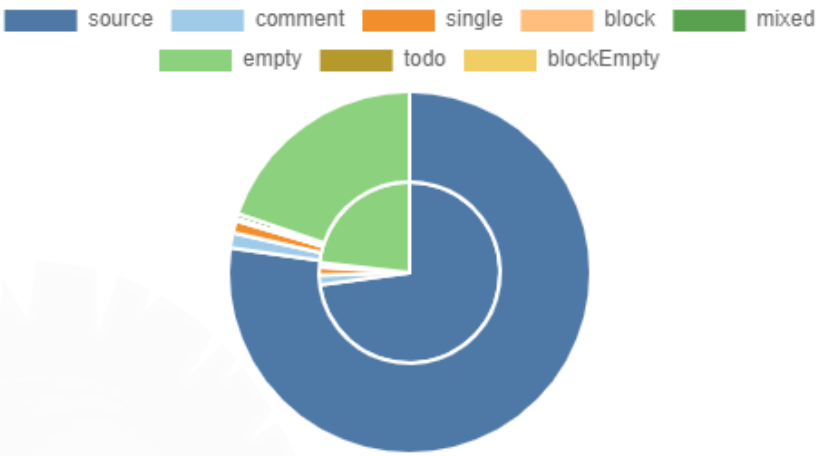
Function Privileges

DividendTracker	Implementation	Ownable, DividendPayingToken		
└	<Constructor>	Public ❗	🛑	DividendPayingToken
└	_transfer	Internal 🔒		
└	withdrawDividend	Public ❗		NO ❗
└	updateMinimumTokenBalanceForDividends	External ❗	🛑	onlyOwner
└	excludeFromDividends	External ❗	🛑	onlyOwner
└	updateClaimWait	External ❗	🛑	onlyOwner
└	setLastProcessedIndex	External ❗	🛑	onlyOwner
└	getLastProcessedIndex	External ❗		NO ❗
└	getNumberOfTokenHolders	External ❗		NO ❗
└	getAccount	Public ❗		NO ❗
└	getAccountAtIndex	Public ❗		NO ❗
└	canAutoClaim	Private 🔐		
└	setBalance	External ❗	🛑	onlyOwner
└	process	Public ❗	🛑	NO ❗
└	processAccount	Public ❗	🛑	onlyOwner
MagicEgg	Implementation	ERC20, Ownable		
└	<Constructor>	Public ❗	💸	ERC20
└	<Receive Ether>	External ❗	💸	NO ❗
└	claimStuckTokens	External ❗	🛑	onlyOwner
└	isContract	Internal 🔒		
└	sendBNB	Internal 🔒	🛑	
└	_setAutomatedMarketMakerPair	Private 🔐	🛑	
└	excludeFromFees	External ❗	🛑	onlyOwner
└	isExcludedFromFees	Public ❗		NO ❗
└	_transfer	Internal 🔒	🛑	
└	swapAndSendDividends	Private 🔐	🛑	
└	setSwapTokensAtAmount	External ❗	🛑	onlyOwner
└	updateClaimWait	External ❗	🛑	onlyOwner
└	getClaimWait	External ❗		NO ❗
└	getTotalDividendsDistributed	External ❗		NO ❗
└	withdrawableDividendOf	Public ❗		NO ❗
└	dividendTokenBalanceOf	Public ❗		NO ❗
└	totalRewardsEarned	Public ❗		NO ❗
└	excludeFromDividends	External ❗	🛑	onlyOwner
└	getAccountDividendsInfo	External ❗		NO ❗
└	getAccountDividendsInfoAtIndex	External ❗		NO ❗
└	processDividendTracker	External ❗	🛑	NO ❗
└	claim	External ❗	🛑	NO ❗
└	claimAddress	External ❗	🛑	onlyOwner
└	getLastProcessedIndex	External ❗		NO ❗
└	setLastProcessedIndex	External ❗	🛑	onlyOwner
└	getNumberOfDividendTokenHolders	External ❗		NO ❗

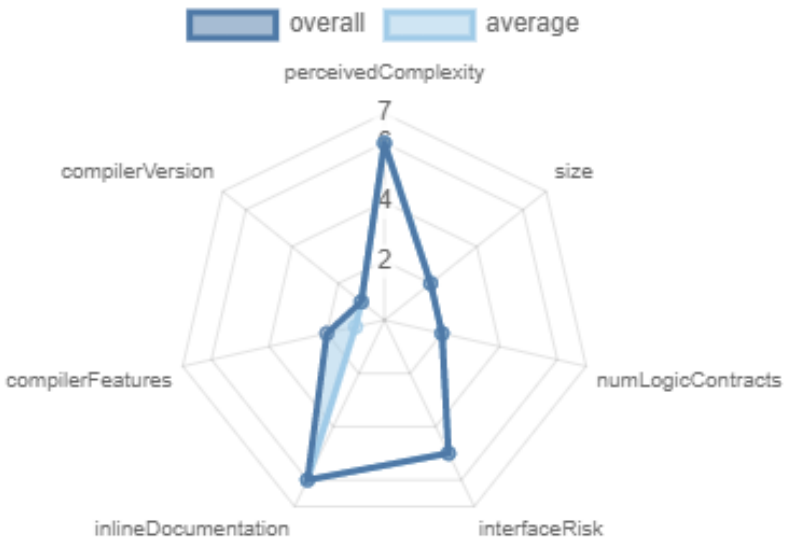
Inheritance Graph



Source Lines



Risk



Manual Review

Informational

Owner has the authority to exclude account from dividend

```
function excludeFromDividends(address account) external onlyOwner{
    dividendTracker.excludeFromDividends(account);
}
```

Description

excludeFromDividends *function, designed to exclude a specified account from receiving dividends.*

Recommendation

While assuming input validation occurs before invoking this function, it is advisable to explicitly validate the account parameter to ensure its conformity as a valid address. Implementing error-handling mechanisms to gracefully manage potential exceptions is also recommended. Ensure that appropriate access control mechanisms are in place to restrict the excludeFromDividends function to only be called by the contract owner

Manual Review

Informational

Owner can exclude specific accounts from paying fees

```
function excludeFromFees(address account, bool excluded) external onlyOwner {
    require(_isExcludedFromFees[account] != excluded, "Account is already set to that state");
    _isExcludedFromFees[account] = excluded;

    emit ExcludeFromFees(account, excluded);
}
```

Description

excludeFromFees *allows the contract owner to modify the exclusion status of an account from fees by updating the _isExcludedFromFees mapping.*

Recommendation

No specific recommendation is necessary for the ***excludeFromFees*** *function at this time. However, it is important to ensure that the function is being used appropriately and that the owner's ability to exclude or include accounts from fees is clearly documented and understood.*

Manual Review

Informational

Owner has the authority to change swap settings.

```
function setSwapTokensAtAmount(uint256 newAmount) external onlyOwner{
    require(newAmount > totalSupply() / 100_000, "SwapTokensAtAmount must be greater than 0.001% of total supply");
    swapTokensAtAmount = newAmount;
}
```

Description

setSwapTokensAtAmount *function updates the* ***swapTokensAtAmount*** *variable to the provided value, which represents the minimum amount of tokens required for a swap to occur.*

Recommendation

Validate the input values provided to ensure they conform to any specific constraints or requirements. For example, ensure that the ***newAmount*** *provided in* ***setSwapTokensAtAmount*** *is within acceptable ranges and aligned with the tokenomics of the project. Verify that appropriate access control mechanisms are in place to restrict these functions to only be called by the contract owner*

Manual Review

Informational

Owner has authority to claim stuck token

```
function claimStuckTokens(address token) external onlyOwner {
    require(token != address(this), "Owner cannot claim native tokens");
    if (token == address(0x0)) {
        payable(msg.sender).transfer(address(this).balance);
        return;
    }
    IERC20 ERC20token = IERC20(token);
    uint256 balance = ERC20token.balanceOf(address(this));
    ERC20token.transfer(msg.sender, balance);
}
```

Description

claimStuckTokens *that allows the contract owner to claim tokens that may have become stuck in the contract. For native tokens, if the provided token address is the same as the contract address (address(this)), an error message is returned since the* ***owner cannot claim native tokens.***

Recommendation

Verify that appropriate access control mechanisms are in place to restrict this function to only be called by the contract owner. Consider adding additional error handling mechanisms to handle exceptional cases, such as when the provided token address is invalid or the transfer of tokens fails. This will provide better feedback and help identify any issues during the token claiming process.

Disclaimer

SecureWise provides the smart contract audit of solidity. Audit and report are for informational purposes only and not, nor should be considered, as an endorsement to engage with, invest in, participate, provide an incentive, or disapprove, criticise, discourage, or purport to provide an opinion on any particular project or team.

This audit report doesn't provide any warranty or guarantee regarding the nature of the technology analysed. These reports, in no way, provide investment advice, nor should be used as investment advice of any sort. Investors must always do their own research and manage their risk.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and SecureWise and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) SecureWise owe no duty of care towards you or any other person, nor does SecureWise make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and SecureWise hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, SecureWise hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against SecureWise, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

AUDIT REPORT

SecureWise

securewise.org

t.me/securewisehub

twitter.com/securewiseAudit