auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Meta Genesis

DATED : 21 JAN 23'

AUDIT SUMMARY

Project name – Meta Genesis

Date: 21 January , 2023

Scope of Audit- Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

Audit Status: Passed (Contract is developed by Pinksale safu dev)

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	0	0	0
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- Goerli:
all tests were done on Goerli network, each test has its transaction has attached to it.

3- Slither : Static Analysis

TESTNET LINKS

All tests were done using this contract, tests are done on goerli

https://goerli.etherscan.io/token/0x2dc2B668c2F9F99a907F6B35e2b3D3f6CCB68371

Token Address: Not Deployed on Chain

Checksum:
f0e4c2f76c58916ec258f246851bea091d14d4247a2fc3e18694461b1816e13b

Deployer: Not Deployed on Chain

Owner: Not Deployed on Chain

TOKEN OVERVIEW

Fees:

Buy Fees: 0%

Sell Fees: 0%

Transfer Fees: 0%

Fees Privilige: No Fees

Ownership : Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Priviliges: whitelisting wallets before launch

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

- Return values of low-level calls
- Private modifier
- Multiple Sends
- Using Suicide
- Gas Limitand Loops
- Address hardcoded
- Exception Disorder
- Using inline assembly
- Divide before multiply
- Missing Zero Address Validation
- Compiler version not fixed
- **Gasless Send**
- Using block.timestamp
- Re-entrancy
- Tautology or contradiction
- Timestamp Dependence
- Revert/require functions
- Use of tx.origin
- Integer overflow/underflow
- Dangerous strict equalities
- Using SHA3
- Using throw

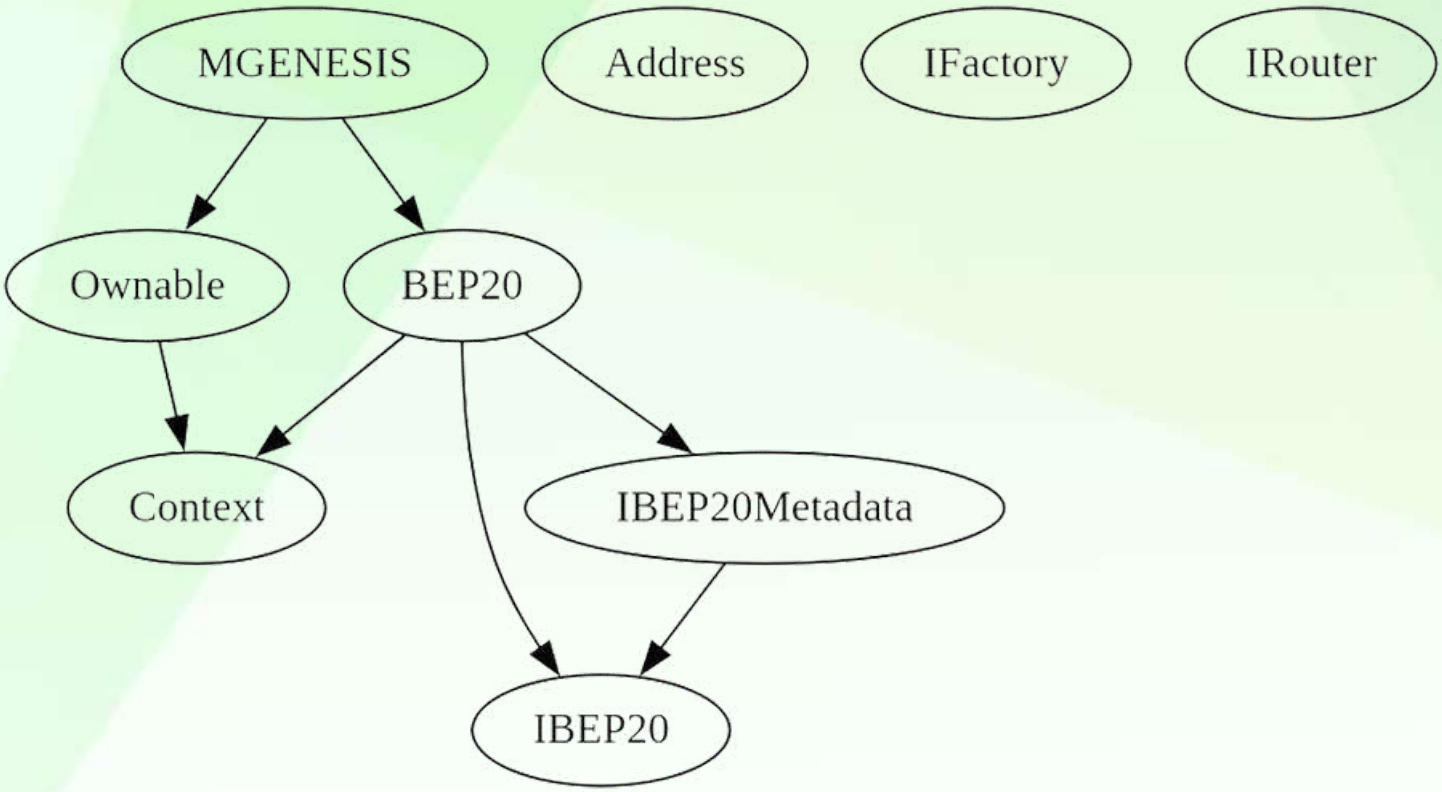
CLASSIFICATION OF RISK

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	0
Medium-Risk	0
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to set taxes (0% tax)
- Owner is not able to blacklist an arbitrary wallet
- Owner is not able to set max buy/sell/transfer amounts
- Owner is not able to disable trades
- Owner is not able to mint new tokens

CONTRACT ASSESMENT

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation				
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
IBEP20	Interface				
└	totalSupply	External ❗	|NO ❗	
└	balanceOf	External ❗	|NO ❗	
└	transfer	External ❗	🔴 |NO ❗	
└	allowance	External ❗	|NO ❗	
└	approve	External ❗	🔴 |NO ❗	
└	transferFrom	External ❗	🔴 |NO ❗	
IBEP20Metadata	Interface	IBEP20			
└	name	External ❗	|NO ❗	
└	symbol	External ❗	|NO ❗	
└	decimals	External ❗	|NO ❗	

CONTRACT ASSESMENT

BEP20	Implementation	Context, IBEP20, IBEP20Metadata		
└	<Constructor>	Public ❗	🔴	NO ❗
└	name	Public ❗		NO ❗
└	symbol	Public ❗		NO ❗
└	decimals	Public ❗		NO ❗
└	totalSupply	Public ❗		NO ❗
└	balanceOf	Public ❗		NO ❗
└	transfer	Public ❗	🔴	NO ❗
└	allowance	Public ❗		NO ❗
└	approve	Public ❗	🔴	NO ❗
└	transferFrom	Public ❗	🔴	NO ❗
└	increaseAllowance	Public ❗	🔴	NO ❗
└	decreaseAllowance	Public ❗	🔴	NO ❗
└	_transfer	Internal 🔒	🔴	
└	_tokengeneration	Internal 🔒	🔴	
└	_approve	Internal 🔒	🔴	
Address	Library			
└	sendValue	Internal 🔒	🔴	
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🔴	NO ❗
└	owner	Public ❗		NO ❗
└	renounceOwnership	Public ❗	🔴	onlyOwner
└	transferOwnership	Public ❗	🔴	onlyOwner
└	_setOwner	Private 🔐	🔴	

CONTRACT ASSESMENT

IFactory	Interface			
└	createPair	External ❗	🔴	NO ❗
IRouter	Interface			
└	factory	External ❗		NO ❗
└	WETH	External ❗		NO ❗
MGENESIS	Implementation	BEP20, Ownable		
└	<Constructor>	Public ❗	🔴	BEP20
└	approve	Public ❗	🔴	NO ❗
└	transferFrom	Public ❗	🔴	NO ❗
└	increaseAllowance	Public ❗	🔴	NO ❗
└	decreaseAllowance	Public ❗	🔴	NO ❗
└	transfer	Public ❗	🔴	NO ❗
└	_transfer	Internal 🔒	🔴	
└	EnableTrading	External ❗	🔴	onlyOwner
└	updateWhitelist	External ❗	🔴	onlyOwner
└	bulkWhitelist	External ❗	🔴	onlyOwner
└	rescueBNB	External ❗	🔴	onlyOwner
└	rescueBSC20	External ❗	🔴	onlyOwner
└	burnBSC20	External ❗	🔴	onlyOwner
└	<Receive Ether>	External ❗	💵	NO ❗

Legend

Symbol	Meaning
🔴	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
MGENESIS.rescueBSC20(address,uint256) (contracts/token.sol#473-476) ignores return value by IBEP20(tokenAdd).transfer(owner(),amount) (contracts/token.sol#475)
MGENESIS.burnBSC20(address,uint256) (contracts/token.sol#478-480) ignores return value by IBEP20(tokenAdd).transfer(deadWallet,amount) (contracts/token.sol#479)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

Address.sendValue(address,uint256) (contracts/token.sol#314-319) is never used and should be removed
Context._msgData() (contracts/token.sol#14-17) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.17 (contracts/token.sol#7) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (contracts/token.sol#314-319):
        - (success) = recipient.call{value: amount}() (contracts/token.sol#317)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable BEP20._balances (contracts/token.sol#60) is not in mixedCase
Variable BEP20._allowances (contracts/token.sol#62) is not in mixedCase
Function IRouter.WETH() (contracts/token.sol#362) is not in mixedCase
Function MGENESIS.EnableTrading() (contracts/token.sol#454-457) is not in mixedCase
Parameter MGENESIS.updateWhitelist(address,bool)._address (contracts/token.sol#459) is not in mixedCase
Constant MGENESIS.deadWallet (contracts/token.sol#373) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/token.sol#15)" inContext (contracts/token.sol#9-18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

MGENESIS.pair (contracts/token.sol#369) should be immutable
MGENESIS.router (contracts/token.sol#368) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

Result => No issues found

FUNCTIONAL TESTING

Functionality tests for ERC20 tokens includes:
- adding liquidity
- buying / selling /transferring (for non-excluded wallets)

1- Adding Liquidity:

liquidity added on Uniswap v2:

https://goerli.etherscan.io/tx/0xfddcd2892101e7848e62ae19a24605db11e94d6c90e28e7c8525a81f6150497a

no issue were found on adding liquidity.

2- Buying from a non-excluded wallet:

https://goerli.etherscan.io/tx/0x12803cc58e657fcf02cb9707f8e000cdd806270a52bc129aee018662d1cd3495

3- Selling from a non-excluded wallet

https://goerli.etherscan.io/tx/0x18ee6b549f27a233d2782b146a00411261cfe99126e05ee10b8ca4a2877d563e

MANUAL TESTING

NO RISKS WERE FOUND IN THE CONTRACT

Social Media Overview

Here are the Social Media Accounts of

Mata Genesis

https://t.me/MetaGenesisPortal

https://twitter.com/metagenesiscoin/

https://metagenesistoken.com/

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