AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

MitsukiAI

DATED : 26 October 23'

AUDIT SUMMARY

Project name – MitsukiAI

Date: 26 October 2023

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

Audit Status: Passed

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- 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/token/0xA68d8310102754895686C9ea627c05022f5e0a90

Token Information

Token Address :
0x28eBf570ED22BD4dea2301BD8c23aBb3b742aA07

Name: MitsukiAI

Symbol: MAI

Decimals: 18

Network: Binance smart chain

Token Type: BEP20

Owner: 0x31bd26c05f176Bc295a7Fe75E616a10FC4A4B48E

Deployer: 0x31bd26c05f176Bc295a7Fe75E616a10FC4A4B48E

Token Supply: 1,000,000,000

Checksum:
1666029b29a5f1ae543a23971ebc1e066fc0f1b5

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/token/0xA68d83101027548956868C9ea627c05022f5e0a90

TOKEN OVERVIEW

buy fee: 0-30%

Sell fee: 0-30%

transfer fee: 0%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: No

Blacklist: No

Other Privileges:

- Initial distribution of the tokens
- Modifying fees

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

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

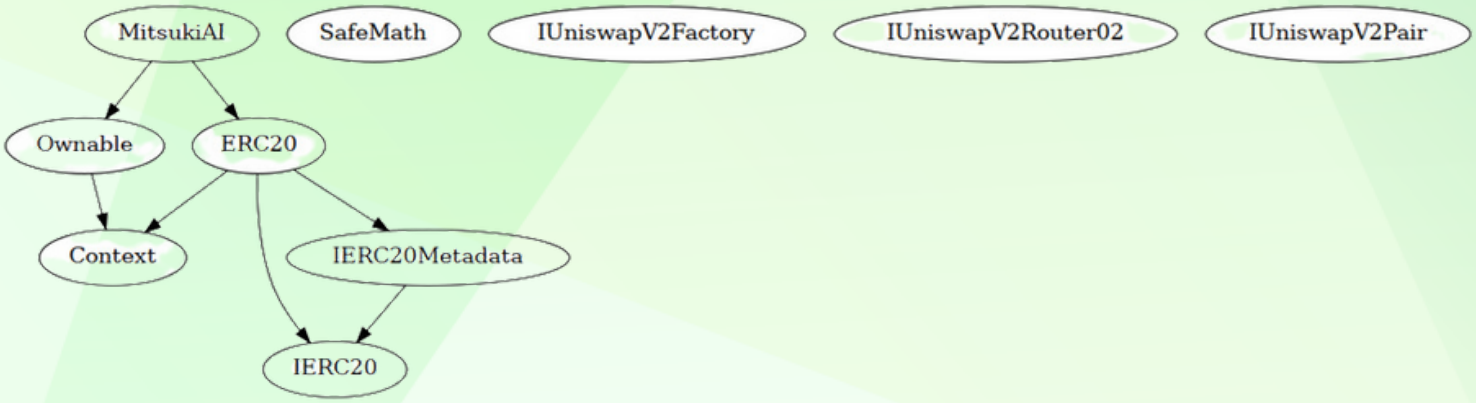
CLASSIFICATION OF RISK

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 able to set buy/sell fees up to 30%**
- Owner is not able to blacklist an arbitrary wallet
- Owner is not able to mint new tokens
- Owner is not able to disable trades
- Owner is not able to set max tx and maximum wallet to zero

STATIC ANALYSIS

```
INFO:Detectors:
Pragma version^0.8.17 (contracts/Token.sol#3) 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:
Function IUniswapV2Router02.WETH() (contracts/Token.sol#209) is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (contracts/Token.sol#253) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (contracts/Token.sol#255) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (contracts/Token.sol#286) is not in mixedCase
Parameter Ordinal20.changeTaxForMarketing(uint256,uint256)._taxBuy (contracts/Token.sol#839) is not in mixedCase
Parameter Ordinal20.changeTaxForMarketing(uint256,uint256)._taxSell (contracts/Token.sol#840) is not in mixedCase
Parameter Ordinal20.changeLimit(uint256,uint256)._maxTxAmount (contracts/Token.sol#852) is not in mixedCase
Parameter Ordinal20.changeLimit(uint256,uint256)._maxWalletAmount (contracts/Token.sol#853) is not in mixedCase
Variable Ordinal20._taxBuyForMarketing (contracts/Token.sol#660) is not in mixedCase
Variable Ordinal20._taxSellForMarketing (contracts/Token.sol#661) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Reentrancy in Ordinal20._transfer(address,address,uint256) (contracts/Token.sol#735-813):
        External calls:
        - sent = address(marketingWallet).send(address(this).balance) (contracts/Token.sol#755-757)
        State variables written after the call(s):
        - super._transfer(uniswapV2Pair,address(0xdead),amountToBurn) (contracts/Token.sol#770-774)
                - _balances[from] = fromBalance - amount (contracts/Token.sol#644)
                - _balances[to] += amount (contracts/Token.sol#647)
        - super._transfer(from,address(this),marketingShare) (contracts/Token.sol#807)
                - _balances[from] = fromBalance - amount (contracts/Token.sol#644)
                - _balances[to] += amount (contracts/Token.sol#647)
        - super._transfer(from,to,transferAmount) (contracts/Token.sol#809)
                - _balances[from] = fromBalance - amount (contracts/Token.sol#644)
                - _balances[to] += amount (contracts/Token.sol#647)
        - _marketingReserves += marketingShare (contracts/Token.sol#805)
        Event emitted after the call(s):
        - Transfer(from,to,amount) (contracts/Token.sol#650)
                - super._transfer(from,to,transferAmount) (contracts/Token.sol#809)
        - Transfer(from,to,amount) (contracts/Token.sol#650)
                - super._transfer(from,address(this),marketingShare) (contracts/Token.sol#807)
        - Transfer(from,to,amount) (contracts/Token.sol#650)
                - super._transfer(uniswapV2Pair,address(0xdead),amountToBurn) (contracts/Token.sol#770-774)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
Ordinal20._decimals (contracts/Token.sol#658) should be constant
Ordinal20._name (contracts/Token.sol#656) should be constant
Ordinal20._supply (contracts/Token.sol#659) should be constant
Ordinal20._symbol (contracts/Token.sol#657) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
Ordinal20._numTokensSellToAddToETH (contracts/Token.sol#673) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:./contracts/Token.sol analyzed (10 contracts with 88 detectors), 40 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

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
IERC20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
IERC20Metadata	Interface	IERC20		
└	name	External ❗		NO❗
└	decimals	External ❗		NO❗
└	symbol	External ❗		NO❗
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner

CONTRACT ASSESMENT

IUniswapV2Factory	Interface			
└	createPair	External ❗	🛑	NO❗
IUniswapV2Router02	Interface			
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	factory	External ❗		NO❗
└	WETH	External ❗		NO❗
└	addLiquidityETH	External ❗	💵	NO❗
IUniswapV2Pair	Interface			
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transfer	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
└	DOMAIN_SEPARATOR	External ❗		NO❗
└	PERMIT_TYPEHASH	External ❗		NO❗
└	nonces	External ❗		NO❗
└	permit	External ❗	🛑	NO❗
└	MINIMUM_LIQUIDITY	External ❗		NO❗
└	factory	External ❗		NO❗
└	token0	External ❗		NO❗
└	token1	External ❗		NO❗
└	getReserves	External ❗		NO❗
└	price0CumulativeLast	External ❗		NO❗
└	price1CumulativeLast	External ❗		NO❗
└	kLast	External ❗		NO❗
└	mint	External ❗	🛑	NO❗
└	burn	External ❗	🛑	NO❗
└	swap	External ❗	🛑	NO❗
└	skim	External ❗	🛑	NO❗
└	sync	External ❗	🛑	NO❗
└	initialize	External ❗	🛑	NO❗

CONTRACT ASSESMENT

ERC20	Implementation	Context, IERC20, IERC20Metadata		
└	<Constructor>	Public ❗	🛑	NO❗
└	symbol	External ❗		NO❗
└	name	External ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	decimals	Public ❗		NO❗
└	totalSupply	External ❗		NO❗
└	allowance	Public ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
└	decreaseAllowance	External ❗	🛑	NO❗
└	increaseAllowance	External ❗	🛑	NO❗
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_spendAllowance	Internal 🔒	🛑	
└	_transfer	Internal 🔒	🛑	
MitsukiAI	Implementation	ERC20, Ownable		
└	<Constructor>	Public ❗	🛑	ERC20
└	_transfer	Internal 🔒	🛑	
└	_swapTokensForEth	Private 🔐	🛑	lockTheSwap
└	changeMarketingWallet	Public ❗	🛑	onlyOwner
└	setIsExcludeFee	Public ❗	🛑	onlyOwner
└	changeThreshold	Public ❗	🛑	onlyOwner
└	changeTaxForMarketing	Public ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵	NO❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

FUNCTIONAL TESTING

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0x953289fb99640a6a3117e74506d476effcb2bdcaf34b646340aac219ba338f7d

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x81c352c956c7e3b5dbb458e310c26d257d29de8df1a4c938763195bd5aeb58e1

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x9a95a24423a9d93275094d59df371c168085e11ec9db90e3b820f07ba6fe4ebf

4- Transferring when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0x7be03d8d5b97f4213aade5a30fec4ca481efb0aab72e34e2957377def163f892

5- Buying when not excluded from fees (tax 0-30%) (passed):
https://testnet.bscscan.com/tx/0x7be03d8d5b97f4213aade5a30fec4ca481efb0aab72e34e2957377def163f892

6- Selling when not excluded from fees (tax 0-30%) (passed):
https://testnet.bscscan.com/tx/0xd283b0e8cb7133a590420f38d18713e65f024e0b289275cffaeb28e13ec19467

7- Transferring when not excluded from fees (0 % tax) (passed):
https://testnet.bscscan.com/tx/0xd54b12e178007ac5801dc1d8117734489a797692a1f600eaa1016270fe7c906f

7- Internal swap (Marketing BNB) (passed):
https://testnet.bscscan.com/tx/0xd283b0e8cb7133a590420f38d18713e65f024e0b289275cffaeb28e13ec19467

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