



Smart Contract Audit

FOR

MARSU

DATED : 29 MAY 23'



AUDIT SUMMARY

Project name – MARSU

Date: 29 May, 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	1

USED TOOLS

Tools:

1.Manual Review: The code has undergone a line-by-line review by the **Ace** team.

2.ETH Test Network: All tests were conducted on the ETH 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/0x98ED8A59CA121DA7AB01633B963c3b8C22Ba1308>



Token Information

Name : Marsupilami Inu

Symbol : MARSU

Decimals: 9

Network: Binance smart chain

Token Type: BEP20

Token Address:

0x8FEA3bf0bcdD542EC884A72319cE420D04eE2040

Owner:

0x7320764F01B443Ed7Db15161E02650115AF5bb42
(at time of writing the audit)

Deployer: 0x7320764F01B443Ed7Db15161E02650115
AF5bb42



Token Information

Fees:

Buy Fees: 8%

Sell Fees: 8%

Transfer Fees: 8%

Fees Privilege: static fees

Ownership :

0x7320764F01B443Ed7Db15161E02650115AF5bb42

Minting: None

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: - Initial distribution of the tokens

- excluding from fees

- including in fees

- enabling trades

- changing internal swap settings



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

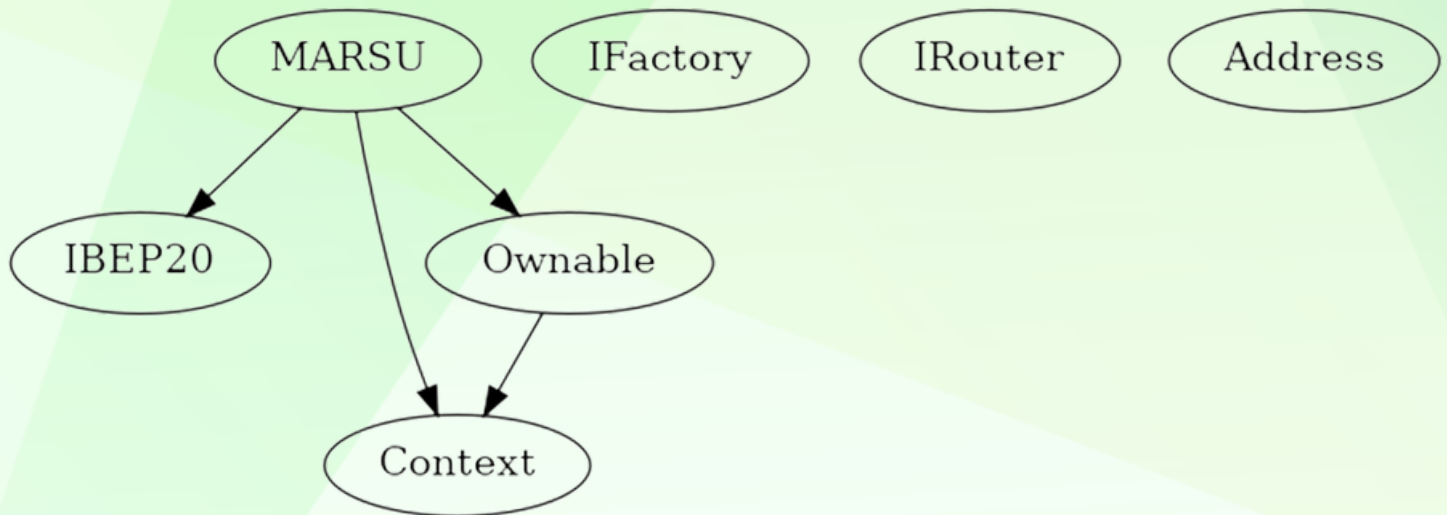
Findings

Severity

Found

◆ Critical	0
◆ High-Risk	0
◆ Medium-Risk	0
◆ Low-Risk	0
◆ Gas Optimization / Suggestions	1

INHERITANCE TREE





POINTS TO NOTE

- Fees are 0 (static)
 - Owner is not able to blacklist an arbitrary address.
 - Owner is not able to disable trades
 - Owner is not able to limit buy/sell/transfer/wallet amounts
 - Owner is not able to mint new tokens
-



CONTRACT ASSESMENT

Contract	Type	Bases			
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**	
IBEP20 Interface					
└	totalSupply	External	!		NO !
└	balanceOf	External	!		NO !
└	transfer	External	!		NO !
└	allowance	External	!		NO !
└	approve	External	!		NO !
└	transferFrom	External	!		NO !
Context Implementation					
└	_msgSender	Internal	🔒		
└	_msgData	Internal	🔒		
Ownable Implementation Context					
└	<Constructor>	Public	!		NO !
└	owner	Public	!		NO !
└	renounceOwnership	Public	!		onlyOwner
└	transferOwnership	Public	!		onlyOwner
└	_setOwner	Private	🔒		
IFactory Interface					
└	createPair	External	!		NO !
IRouter Interface					
└	factory	External	!		NO !
└	WETH	External	!		NO !
└	addLiquidityETH	External	!		NO !
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External	!		NO !
Address Library					
└	sendValue	Internal	🔒		
MARSU Implementation Context, IBEP20, Ownable					
└	<Constructor>	Public	!		NO !
└	name	Public	!		NO !
└	symbol	Public	!		NO !
└	decimals	Public	!		NO !
└	totalSupply	Public	!		NO !
└	balanceOf	Public	!		NO !
└	allowance	Public	!		NO !
└	approve	Public	!		NO !

CONTRACT ASSESMENT

		transferFrom		Public	!		●		NO	!	
		increaseAllowance		Public	!		●		NO	!	
		decreaseAllowance		Public	!		●		NO	!	
		transfer		Public	!		●		NO	!	
		isExcludedFromReward		Public	!				NO	!	
		reflectionFromToken		Public	!				NO	!	
		EnableTrading		External	!		●		onlyOwner		
		updatedecline		External	!		●		onlyOwner		
		tokenFromReflection		Public	!				NO	!	
		excludeFromReward		Public	!		●		onlyOwner		
		includeInReward		External	!		●		onlyOwner		
		excludeFromFee		Public	!		●		onlyOwner		
		includeInFee		Public	!		●		onlyOwner		
		isExcludedFromFee		Public	!				NO	!	
		_reflectRfi		Private	🔒		●				
		_takeLiquidity		Private	🔒		●				
		_takeMarketing		Private	🔒		●				
		_takeOps		Private	🔒		●				
		_takeDev		Private	🔒		●				
		_getValues		Private	🔒						
		_getTValues		Private	🔒						
		_getRValues1		Private	🔒						
		_getRValues2		Private	🔒						
		_getRate		Private	🔒						
		_getCurrentSupply		Private	🔒						
		_approve		Private	🔒		●				
		_transfer		Private	🔒		●				
		_tokenTransfer		Private	🔒		●				
		swapAndLiquify		Private	🔒		●		lockTheSwap		
		addLiquidity		Private	🔒		●				
		swapTokensForBNB		Private	🔒		●				
		bulkExcludeFromFee		External	!		●		onlyOwner		
		bulkIncludeInFee		External	!		●		onlyOwner		
		updateMarketingWallet		External	!		●		onlyOwner		
		updateDevWallet		External	!		●		onlyOwner		
		updateOpsWallet		External	!		●		onlyOwner		
		updateSwapTokensAtAmount		External	!		●		onlyOwner		
		updateSwapEnabled		External	!		●		onlyOwner		
		rescueBNB		External	!		●		onlyOwner		
		rescueAnyBEP20Tokens		Public	!		●		onlyOwner		
		<Receive Ether>		External	!		🟢		NO	!	



CONTRACT ASSESMENT

Legend

Symbol	Meaning
:	
●	Function can modify state
💰	Function is payable



STATIC ANALYSIS

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
MARSU.includeInReward(address) (contracts/Token.sol#404-415) has costly operations inside a loop:
- _excluded.pop() (contracts/Token.sol#411)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
Context._msgData() (contracts/Token.sol#46-49) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
MARSU._rTotal (contracts/Token.sol#165) is set pre-construction with a non-constant function or state variable:
- (MAX - (MAX % _tTotal))
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state
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-0.8.20 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#126-137):
- (success) = recipient.call{value: amount}() (contracts/Token.sol#132)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
Function IRouter.WETH() (contracts/Token.sol#102) is not in mixedCase
Struct MARSU.valuesFromGetValues (contracts/Token.sol#202-216) is not in CapWords
Function MARSU.EnableTrading() (contracts/Token.sol#370-375) is not in mixedCase
Parameter MARSU.updatedDeadline(uint256)._deadline (contracts/Token.sol#377) is not in mixedCase
Parameter MARSU.updateSwapEnabled(bool)._enabled (contracts/Token.sol#796) is not in mixedCase
Parameter MARSU.rescueAnyBEP20Tokens(address,address,uint256)._tokenAddr (contracts/Token.sol#808) is not in mixedCase
Parameter MARSU.rescueAnyBEP20Tokens(address,address,uint256)._to (contracts/Token.sol#809) is not in mixedCase
Parameter MARSU.rescueAnyBEP20Tokens(address,address,uint256)._amount (contracts/Token.sol#810) is not in mixedCase
Constant MARSU._decimals (contracts/Token.sol#161) is not in UPPER_CASE_WITH_UNDERSCORES
Variable MARSU.genesis_block (contracts/Token.sol#169) is not in mixedCase
Constant MARSU._name (contracts/Token.sol#177) is not in UPPER_CASE_WITH_UNDERSCORES
Constant MARSU._symbol (contracts/Token.sol#178) 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#47)" inContext (contracts/Token.sol#41-50)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
MARSU.updateSwapTokensAtAmount(uint256) (contracts/Token.sol#788-794) uses literals with too many digits:
- require(bool,string)(amount <= 420690000000000, Cannot set swap threshold amount higher than 1% of tokens) (contracts/Token.sol#789-792)
MARSU.slitherConstructorVariables() (contracts/Token.sol#140-820) uses literals with too many digits:
- _tTotal = 4206900000000000 * 10 ** _decimals (contracts/Token.sol#164)
MARSU.slitherConstructorVariables() (contracts/Token.sol#140-820) uses literals with too many digits:
- swapTokensAtAmount = 420690000000000 * 10 ** 9 (contracts/Token.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
MARSU._lastSell (contracts/Token.sol#156) is never used in MARSU (contracts/Token.sol#140-820)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
MARSU._tTotal (contracts/Token.sol#164) should be constant
MARSU._deadWallet (contracts/Token.sol#172) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
MARSU._pair (contracts/Token.sol#159) should be immutable
MARSU._router (contracts/Token.sol#158) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

Static Analysis

an static analysis of the code were performed using
slither. No issues were found



FUNCTIONAL TESTING

Router (PCS V2):

0xD99D1c33F9fC3444f8101754aBC46c52416550D1

1- Adding liquidity (passed):

<https://testnet.bscscan.com/tx/0xdfd4f148cc87e8f446915375c11cf3ff71a071cbbfd4f48b61a6bab80d8133a5>

2- Buying (0% tax) (passed):

<https://testnet.bscscan.com/tx/0x16ccb38c0b6a2fd0e7a5f4a9e4c52159ccd96ea8b773d562d765525ff28520ac>

3- Selling (0% tax) (passed):

<https://testnet.bscscan.com/tx/0x09948eed5f1f8cced7e2e508a1d9c7e4c808e10b9eef5f24a4fa7e07dfefb29d5>

4- Transferring 0% tax) (passed):

<https://testnet.bscscan.com/tx/0x7f9b6cb28089e717eddf834c1515c000c410440054f2ef9ae94d04dbc62f9c3f>

5- Buying when not excluded from fees (0-10% tax) (passed):

<https://testnet.bscscan.com/tx/0xfc30a4a5318d2b8bc425d22a7c7b06d1b9340bef81dfbb80089cbaf98bd9d6ae>

6- Selling when not excluded from fees (0-10% tax) (passed):

<https://testnet.bscscan.com/tx/0xc92f551a22aa9387aabe5d65f1c04f29d76f3d13ddb661fa7d7c05bcc6957d44>

7- Transferring when not excluded from fees (0-10% tax) (passed):

<https://testnet.bscscan.com/tx/0x9b5f0f3778e9c4e4518ce099cdcbbbfa0da23915a20394128f037b316087cec1>



FUNCTIONAL TESTING

8- Internal swap & rewards distribution (passed):

- BNB sent to marketing wallet
 - A portion of tokens burnt
 - A portion of tokens were added to liquidity
 - A portion of tokens swapped to BUSD and sent to dividend tracker
 - Dividend tracker distributed reward tokens (BUSD)
-

FUNCTIONAL TESTING

Centralization – Trades must be enabled

Severity: **Informational**

function: EnableTrading

Status: **Resolved (owned by safu dev)**

Overview:

The smart contract owner must enable trades for holders. If trading remain disabled, no one would be able to buy/sell/transfer tokens.

```
function EnableTrading() external onlyOwner {  
    require(!tradingEnabled, "Cannot re-enable trading");  
    tradingEnabled = true;  
    swapEnabled = true;  
    genesis_block = block.number;  
}
```

Suggestion

To mitigate this centralization issue, we propose the following options:

1. Renounce Ownership: Consider relinquishing control of the smart contract by renouncing ownership. This would remove the ability for a single entity to manipulate the router, reducing centralization risks.
2. Multi-signature Wallet: Transfer ownership to a multi-signature wallet. This would require multiple approvals for any changes to the mainRouter, adding an additional layer of security and reducing the centralization risk.
3. Transfer ownership to a trusted and valid 3rd party in order to guarantee enabling of the trades



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 specialize 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>
