MaidaInu.Finance

Smart Contract Audit Report

SYSFIXED

07 Feb 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
MaidaInu.Finance	MAIDA	Binance Smart Chain

Addresses

Contract address	0x754eA224B4e85c1b3AF7A5c1C08b28B3a296b776
Contract deployer address	0xCe34E942A441Fa7Afa313EB482C9375608a8E2eC

Project Website

https://maidainu.finance/

Codebase

https://bscscan.com/address/0x754eA224B4e85c1b3AF7A5c1C08b28B3a296b776#code

SUMMARY

We are very happy to announce that we are going to enter the market very soon and hope to make history, step by step we will complete everything please stay with us and know everything see everything then invest

Contract Summary

Documentation Quality

MaidaInu.Finance provides a very good documentation with standard of solidity base code.

- The technical description is provided clearly and structured and also dont have any high risk issue.

Code Quality

The Overall quality of the basecode is standard.

- Standard solidity basecode and rules are already followed by MaidaInu.Finance with the discovery of several low issues.

Test Coverage

Test coverage of the project is 100% (Through Codebase)

Audit Findings Summary

- SWC-100 SWC-108 | Explicitly define visibility for all state variables on lines 174, 175, 176, 177, 183, 187, 188, 190, 191, 192, 195, 196, 197, 198, 199 and 212.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 14, 24, 33, 34, 44, 183, 183, 184, 184, 185, 185, 211, 211, 280, 286, 317 and 403.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 10.
- SWC-110 SWC-123 | It is recommended to use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 344, 345, 383 and 384.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 317 and 399.

CONCLUSION

We have audited the MaidaInu.Finance project released on February 2023 to discover issues and identify potential security vulnerabilities in MaidaInu.Finance Project. This process is used to find technical issues and security loopholes which might be found in the smart contract.

The security audit report provides a satisfactory result with some low-risk issues.

The issues found in the MaidaInu.Finance smart contract code do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are some arithmetic operation issues, a floating pragma is set, a state variable visibility is not set, weak sources of randomness and out of bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	ISSUE FOUND
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
Unprotected Ether Withdrawal	SWC-105	Due to missing or insufficient access controls, malicious parties can withdraw from the contract.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Uninitialized Storage Pointer	SWC-109	Uninitialized local storage variables can point to unexpected storage locations in the contract.	PASS
Assert Violation	SWC-110 SWC-123	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS

DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	PASS
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique ID	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	ISSUE FOUND
Write to Arbitrary Storage Location	SWC-124	The contract is responsible for ensuring that only authorized user or contract accounts may write to sensitive storage locations.	PASS
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS
Insufficient Gas Griefing	SWC-126	Insufficient gas griefing attacks can be performed on contracts which accept data and use it in a sub-call on another contract.	PASS
Arbitrary Jump Function	SWC-127	As Solidity doesnt support pointer arithmetics, it is impossible to change such variable to an arbitrary value.	PASS

Typographical Error	SWC-129	A typographical error can occur for example when the intent of a defined operation is to sum a number to a variable.	PASS
Override control character	SWC-130	Malicious actors can use the Right-To-Left-Override unicode character to force RTL text rendering and confuse users as to the real intent of a contract.	PASS
Unused variables	SWC-131 SWC-135	Unused variables are allowed in Solidity and they do not pose a direct security issue.	PASS
Unexpected Ether balance	SWC-132	Contracts can behave erroneously when they strictly assume a specific Ether balance.	PASS
Hash Collisions Variable	SWC-133	Using abi.encodePacked() with multiple variable length arguments can, in certain situations, lead to a hash collision.	PASS
Hardcoded gas amount	SWC-134	The transfer() and send() functions forward a fixed amount of 2300 gas.	PASS
Unencrypted Private Data	SWC-136	It is a common misconception that private type variables cannot be read.	PASS

SMART CONTRACT ANALYSIS

Started	Monday Feb 06 2023 17:56:40 GMT+0000 (Coordinated Universal Time)
Finished	Tuesday Feb 07 2023 20:33:02 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	MAIDAINU.sol

Detected Issues

ID	Title	Severity	Status
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged

SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 14

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- MAIDAINU.sol

Locations

```
13   function add(uint256 a, uint256 b) internal pure returns (uint256) {
14   uint256 c = a + b;
15   require(c >= a, "SafeMath: addition overflow");
16
17   return c;
18
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 24

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- MAIDAINU.sol

Locations

```
23   require(b <= a, errorMessage);
24   uint256 c = a - b;
25
26   return c;
27   }
28
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 33

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- MAIDAINU.sol

Locations

```
32
33    uint256 c = a * b;
34    require(c / a == b, "SafeMath: multiplication overflow");
35
36    return c;
37
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 34

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- MAIDAINU.sol

Locations

```
33    uint256 c = a * b;
34    require(c / a == b, "SafeMath: multiplication overflow");
35
36    return c;
37    }
38
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 44

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- MAIDAINU.sol

Locations

```
43   require(b > 0, errorMessage);
44   uint256 c = a / b;
45   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
46
47   return c;
48
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 183

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- MAIDAINU.sol

Locations

```
182
183   uint256 _totalSupply = 100000000 * (10 ** _decimals);
184   uint256 public _maxTxAmount = (_totalSupply * 2) / 1000;
185   uint256 public _maxWalletSize = (_totalSupply * 2) / 1000;
186
187
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 183

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- MAIDAINU.sol

Locations

```
182
183   uint256 _totalSupply = 100000000 * (10 ** _decimals);
184   uint256 public _maxTxAmount = (_totalSupply * 2) / 1000;
185   uint256 public _maxWalletSize = (_totalSupply * 2) / 1000;
186
187
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 184

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- MAIDAINU.sol

Locations

```
183   uint256 _totalSupply = 100000000 * (10 ** _decimals);
184   uint256 public _maxTxAmount = (_totalSupply * 2) / 1000;
185   uint256 public _maxWalletSize = (_totalSupply * 2) / 1000;
186
187   mapping (address => uint256) _balances;
188
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 184

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- MAIDAINU.sol

Locations

```
183   uint256 _totalSupply = 100000000 * (10 ** _decimals);
184   uint256 public _maxTxAmount = (_totalSupply * 2) / 1000;
185   uint256 public _maxWalletSize = (_totalSupply * 2) / 1000;
186
187   mapping (address => uint256) _balances;
188
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 185

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- MAIDAINU.sol

Locations

```
184   uint256 public _maxTxAmount = (_totalSupply * 2) / 1000;
185   uint256 public _maxWalletSize = (_totalSupply * 2) / 1000;
186
187   mapping (address => uint256) _balances;
188   mapping (address => mapping (address => uint256)) _allowances;
189
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 185

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- MAIDAINU.sol

Locations

```
184   uint256 public _maxTxAmount = (_totalSupply * 2) / 1000;
185   uint256 public _maxWalletSize = (_totalSupply * 2) / 1000;
186
187   mapping (address => uint256) _balances;
188   mapping (address => mapping (address => uint256)) _allowances;
189
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 211

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- MAIDAINU.sol

Locations

```
210   bool public swapEnabled = true;
211   uint256 public swapThreshold = _totalSupply / 10000 * 50; // 0.25%
212   bool inSwap;
213   modifier swapping() { inSwap = true; _; inSwap = false; }
214
215
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 211

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- MAIDAINU.sol

Locations

```
210   bool public swapEnabled = true;
211   uint256 public swapThreshold = _totalSupply / 10000 * 50; // 0.25%
212   bool inSwap;
213   modifier swapping() { inSwap = true; _; inSwap = false; }
214
215
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 280

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- MAIDAINU.sol

Locations

```
279   if (recipient != pair && recipient != DEAD) {
280   require(isTxLimitExempt[recipient] || _balances[recipient] + amount <=
_maxWalletSize, "Transfer amount exceeds the bag size.");
281   }
282   if (sender == pair &&
283   opCooldownEnabled &&
284
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 286

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- MAIDAINU.sol

Locations

```
285   require(cooldownTimer[recipient] < block.timestamp,"Please wait for 1min between two operations");
286   cooldownTimer[recipient] = block.timestamp + cooldownTimerInterval;
287   }
288   if(shouldSwapBack()){ swapBack(); }
289
290
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 317

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- MAIDAINU.sol

Locations

```
316   function getTotalFee(bool selling) public view returns (uint256) {
317   if(launchedAt + 5 >= block.number){ return feeDenominator.sub(1); }
318   if(selling) { return totalFee.mul(_sellMultiplier); }
319   return totalFee;
320   }
321
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 403

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- MAIDAINU.sol

Locations

```
402   function setMaxWallet(uint256 amount) external onlyOwner {
403   require(amount >= _totalSupply / 1000 );
404   _maxWalletSize = amount;
405   }
406
407
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 10

low SEVERITY

The current pragma Solidity directive is ""^0.8.11"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- MAIDAINU.sol

Locations

```
9   //SPDX-License-Identifier: MIT
10   pragma solidity ^0.8.11;
11
12   library SafeMath {
13   function add(uint256 a, uint256 b) internal pure returns (uint256) {
14
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 174

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "WBNB" is internal. Other possible visibility settings are public and private.

Source File

- MAIDAINU.sol

Locations

```
173
174   address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c;
175   address DEAD = 0x000000000000000000000000000000000000dEaD;
176   address ZERO = 0x0000000000000000000000000000000000000000;
177   address routerAddress = 0x10ED43C718714eb63d5aA57B78B54704E256024E; // MAINNET
178
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 175

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "DEAD" is internal. Other possible visibility settings are public and private.

Source File

- MAIDAINU.sol

Locations

```
174   address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c;
175   address DEAD = 0x000000000000000000000000000000000000dEaD;
176   address ZERO = 0x0000000000000000000000000000000000000000;
177   address routerAddress = 0x10ED43C718714eb63d5aA57B78B54704E256024E; // MAINNET
178
179
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 176

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "ZERO" is internal. Other possible visibility settings are public and private.

Source File

- MAIDAINU.sol

Locations

```
175   address DEAD = 0x000000000000000000000000000000000000dEaD;
176   address ZERO = 0x0000000000000000000000000000000000000000;
177   address routerAddress = 0x10ED43C718714eb63d5aA57B78B54704E256024E; // MAINNET
178
179   string constant _name = "MaidaInu.Finance ";
180
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 177

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "routerAddress" is internal. Other possible visibility settings are public and private.

Source File

- MAIDAINU.sol

Locations

```
176   address ZERO = 0x0000000000000000000000000000000000000000;
177   address routerAddress = 0x10ED43C718714eb63d5aA57B78B54704E256024E; // MAINNET
178
179   string constant _name = "MaidaInu.Finance ";
180   string constant _symbol = "MAIDA";
181
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 183

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "_totalSupply" is internal. Other possible visibility settings are public and private.

Source File

- MAIDAINU.sol

Locations

```
182
183   uint256 _totalSupply = 100000000 * (10 ** _decimals);
184   uint256 public _maxTxAmount = (_totalSupply * 2) / 1000;
185   uint256 public _maxWalletSize = (_totalSupply * 2) / 1000;
186
187
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 187

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "_balances" is internal. Other possible visibility settings are public and private.

Source File

- MAIDAINU.sol

Locations

```
186
187   mapping (address => uint256) _balances;
188   mapping (address => mapping (address => uint256)) _allowances;
189
190   mapping (address => bool) isFeeExempt;
191
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 188

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "_allowances" is internal. Other possible visibility settings are public and private.

Source File

- MAIDAINU.sol

Locations

```
187   mapping (address => uint256) _balances;
188   mapping (address => mapping (address => uint256)) _allowances;
189
190   mapping (address => bool) isFeeExempt;
191   mapping (address => bool) isTxLimitExempt;
192
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 190

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "isFeeExempt" is internal. Other possible visibility settings are public and private.

Source File

- MAIDAINU.sol

Locations

```
189
190   mapping (address => bool) isFeeExempt;
191   mapping (address => bool) isTxLimitExempt;
192   mapping (address => bool) isTimelockExempt;
193   mapping (address => bool) public isBot;
194
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 191

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "isTxLimitExempt" is internal. Other possible visibility settings are public and private.

Source File

- MAIDAINU.sol

Locations

```
190   mapping (address => bool) isFeeExempt;
191   mapping (address => bool) isTxLimitExempt;
192   mapping (address => bool) isTimelockExempt;
193   mapping (address => bool) public isBot;
194
195
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 192

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "isTimelockExempt" is internal. Other possible visibility settings are public and private.

Source File

- MAIDAINU.sol

Locations

```
191   mapping (address => bool) isTxLimitExempt;
192   mapping (address => bool) isTimelockExempt;
193   mapping (address => bool) public isBot;
194
195   uint256 liquidityFee = 0;
196
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 195

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "liquidityFee" is internal. Other possible visibility settings are public and private.

Source File

- MAIDAINU.sol

Locations

```
194
195   uint256 liquidityFee = 0;
196   uint256 devFee = 2;
197   uint256 marketingFee = 2;
198   uint256 totalFee = 4;
199
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 196

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "devFee" is internal. Other possible visibility settings are public and private.

Source File

- MAIDAINU.sol

Locations

```
195   uint256 liquidityFee = 0;
196   uint256 devFee = 2;
197   uint256 marketingFee = 2;
198   uint256 totalFee = 4;
199   uint256 feeDenominator = 100;
200
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 197

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "marketingFee" is internal. Other possible visibility settings are public and private.

Source File

- MAIDAINU.sol

Locations

```
196   uint256 devFee = 2;
197   uint256 marketingFee = 2;
198   uint256 totalFee = 4;
199   uint256 feeDenominator = 100;
200   uint256 public _sellMultiplier = 1;
201
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 198

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "totalFee" is internal. Other possible visibility settings are public and private.

Source File

- MAIDAINU.sol

Locations

```
197   uint256 marketingFee = 2;
198   uint256 totalFee = 4;
199   uint256 feeDenominator = 100;
200   uint256 public _sellMultiplier = 1;
201
202
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 199

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "feeDenominator" is internal. Other possible visibility settings are public and private.

Source File

- MAIDAINU.sol

Locations

```
198   uint256 totalFee = 4;
199   uint256 feeDenominator = 100;
200   uint256 public _sellMultiplier = 1;
201
202   address public marketingFeeReceiver = 0x54A263e1f8c842dEF0b81Ef02fdC894c270a3A75;
203
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 212

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "inSwap" is internal. Other possible visibility settings are public and private.

Source File

- MAIDAINU.sol

Locations

```
211   uint256 public swapThreshold = _totalSupply / 10000 * 50; // 0.25%
212   bool inSwap;
213   modifier swapping() { inSwap = true; _; inSwap = false; }
214
215   // Cooldown & timer functionality
216
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 344

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- MAIDAINU.sol

Locations

```
343   address[] memory path = new address[](2);
344   path[0] = address(this);
345   path[1] = WBNB;
346
347   uint256 balanceBefore = address(this).balance;
348
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 345

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- MAIDAINU.sol

Locations

```
344   path[0] = address(this);
345   path[1] = WBNB;
346
347   uint256 balanceBefore = address(this).balance;
348
349
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 383

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- MAIDAINU.sol

Locations

```
382   address[] memory path = new address[](2);
383   path[0] = WBNB;
384   path[1] = address(this);
385
386   router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(
387
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 384

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- MAIDAINU.sol

Locations

```
383   path[0] = WBNB;
384   path[1] = address(this);
385
386   router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(
387   0,
388
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 317

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- MAIDAINU.sol

Locations

```
316   function getTotalFee(bool selling) public view returns (uint256) {
317   if(launchedAt + 5 >= block.number){ return feeDenominator.sub(1); }
318   if(selling) { return totalFee.mul(_sellMultiplier); }
319   return totalFee;
320   }
321
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 399

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- MAIDAINU.sol

Locations

```
398   function launch() internal {
399   launchedAt = block.number;
400   }
401
402   function setMaxWallet(uint256 amount) external onlyOwner {
403
```

DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to, or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without Sysfixed's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Sysfixed to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model, or legal compliance.

This is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

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 Sysfixed and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Sysfixed) owe no duty of care.

ABOUT US

Sysfixed is a blockchain security certification organization established in 2021 with the objective to provide smart contract security services and verify their correctness in blockchain-based protocols. Sysfixed automatically scans for security vulnerabilities in Ethereum and other EVM-based blockchain smart contracts. Sysfixed a comprehensive range of analysis techniques—including static analysis, dynamic analysis, and symbolic execution—can accurately detect security vulnerabilities to provide an in-depth analysis report. With a vibrant ecosystem of world-class integration partners that amplify developer productivity, Sysfixed can be utilized in all phases of your project's lifecycle. Our team of security experts is dedicated to the research and improvement of our tools and techniques used to fortify your code.