Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: Mighty Zilla Token
Platform: Binance Smart Chain
Website: MightyZilla.com
Language: Solidity
Date: November 17th, 2021

Table of contents

Email: audit@EtherAuthority.io

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by the Mighty Zilla team to perform the Security audit of the Mighty Zilla Token smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on November 17th, 2021.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

Mighty Zilla is a BEP20 standard token smart contract with other customization like: Swapping, reflation, etc. This audit only considers Mighty Zilla token smart contracts, and does not cover any other smart contracts in the platform.

Audit scope

Name	**Code Review and Security Analysis Report for Mighty Zilla Token Smart Contract**
Platform	**BSC / Solidity**
File	MIGHTYZILLA.sol
File MD5 Hash	35A7317435153C71F0BD7C037F62ECE4
Online code	0xb4bfcd3401ff351603b9b2957298914919abc26e
Audit Date	November 17th, 2021

Email: audit@EtherAuthority.io

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: • Name: Mighty Zilla • Symbol: MIGHTYZILLA • Decimals: 9	**YES, This is valid.**
• Buy Tax Fee: 5% • Buy Stake Tax Fee: 3% • Buy Marketing Pool Fee: 2% • Sell Tax Fee: 5% • Sell Stake Tax Fee: 3% • Sell Marketing Pool Fee: 2% • Number of tokens to exchange for marketing pool: 100 Million • Total Supply: 100 Trillion • Wallet Limit: 1 Trillion • Maximum Tax Amount: 500 Billion	**YES, This is valid.** **Owner authorized wallet can set some percentage value and we suggest handling the private key of that wallet securely.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **“Secured”**. This token contract does contain owner control, which does not make it fully decentralized.

We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 4 low and some very low level issues. These issues are not critical ones.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Moderated
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Moderated
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Moderated
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 1 smart contract file. Smart contract contains Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in Mighty Zilla Token are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the Mighty Zilla Token.

The Mighty Zilla Token team has **not** provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are **not** well commented on smart contracts.

Documentation

We were given a Mighty Zilla Token smart contracts code in the form of a BSCscan web link.The hash of that code is mentioned above in the table.

As mentioned above, code parts are **not well** commented. So it is not easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website http://MightyZilla.com which provided rich information about the project architecture and tokenomics.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	lockTheSwap	modifier	Passed	No Issue
3	initContract	external	access only Owner	No Issue
4	openTrading	external	access only Owner	No Issue
5	setWalletlimit	external	access only Owner	No Issue
6	addDestLimit	external	access only Owner	No Issue
7	removeDestLimit	external	access only Owner	No Issue
8	isBlackListed	read	Passed	No Issue
9	name	read	Passed	No Issue
10	symbol	read	Passed	No Issue
11	decimals	read	Passed	No Issue
12	totalSupply	read	Passed	No Issue
13	balanceOf	read	Passed	No Issue
14	transfer	write	Passed	No Issue
15	allowance	read	Passed	No Issue
16	approve	write	Passed	No Issue
17	transferFrom	write	Passed	No Issue
18	increaseAllowance	write	Passed	No Issue
19	decreaseAllowance	write	Passed	No Issue
20	isExcluded	read	Passed	No Issue
21	setExcludeFromFee	external	Missing Events	Refer Audit Findings
22	totalFees	read	Passed	No Issue
23	RemoveSniper	external	access only Owner	No Issue
24	amnestySniper	external	access only Owner	No Issue
25	deliver	write	Missing Events	Refer Audit Findings
26	reflectionFromToken	read	Passed	No Issue
27	tokenFromReflection	read	Passed	No Issue
28	excludeAccount	external	Missing Events	Refer Audit Findings
29	includeAccount	external	Infinite loop possibility	Refer Audit Findings
30	removeAllFee	write	Passed	No Issue
31	restoreAllFee	write	Passed	No Issue
32	isExcludedFromFee	read	Passed	No Issue

33	_approve	write	Passed	No Issue
34	_transfer	write	Passed	No Issue
35	swapTokensForEth	write	Passed	No Issue
36	sendETHToMarketingPool	write	Missing Events	Refer Audit Findings
37	manualSwap	external	Missing Events	Refer Audit Findings
38	manualSend	external	Missing Events	Refer Audit Findings
39	setSwapEnabled	external	access only Owner	No Issue
40	_tokenTransfer	write	Gas Efficiency	Refer Audit Findings
41	_transferStandard	write	Passed	No Issue
42	_transferToExcluded	write	Passed	No Issue
43	_transferFromExcluded	write	Passed	No Issue
44	_transferBothExcluded	write	Passed	No Issue
45	_takeMarketingPool	write	Passed	No Issue
46	_takeStakePool	write	Passed	No Issue
47	_reflectFee	write	Passed	No Issue
48	_getValues	read	Passed	No Issue
49	receive	external	Passed	No Issue
50	_getValues2	read	Passed	No Issue
51	_getTValues	write	Passed	No Issue
52	_addonvalues	write	Passed	No Issue
53	_getRValues	write	Passed	No Issue
54	_getRate	read	Passed	No Issue
55	_getCurrentSupply	read	Passed	No Issue
56	_getTaxFee	read	Passed	No Issue
57	_getMaxTxAmount	read	Passed	No Issue
58	_getETHBalance	read	Passed	No Issue
59	_setTaxFee	external	access only Owner	No Issue
60	_setbuyTaxFee	external	access only Owner	No Issue
61	_setsellTaxFee	external	access only Owner	No Issue
62	_setStakeFee	external	access only Owner	No Issue
63	_setbuystakeFee	external	access only Owner	No Issue
64	_setsellstakeFee	external	access only Owner	No Issue
65	_setMarketingPoolFee	external	access only Owner	No Issue
66	_setTokenExchange	external	access only Owner	No Issue

67	_setbuyMarketingPoolFee	external	access only Owner	No Issue
68	_setsellMarketingPoolFee	external	access only Owner	No Issue
69	_setMarketingPoolWallet	external	access only Owner	No Issue
70	_setStakePoolAddress	external	access only Owner	No Issue
71	_setMaxTxAmount	external	access only Owner	No Issue
72	owner	read	Passed	No Issue
73	onlyOwner	modifier	Passed	No Issue
74	renounceOwnership	write	access only Owner	No Issue
75	transferOwnership	write	access only Owner	No Issue
76	geUnlockTime	read	Passed	No Issue
77	lock	write	Possible to gain ownership	Refer Audit Findings
78	unlock	write	Possible to gain ownership	Refer Audit Findings

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

(1) Infinite loop possibility:

```
function includeAccount(address account) external onlyOwner() {
    require(_isExcluded[account], "Account is already excluded");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

If there are many excluded wallets, then this logic will fail. Because it might hit the block's gas limit. If there are very limited exceptions, then this will work, but will cost more gas.

Resolution: Just use a mapping that will map the wallet to bool and make excluded wallets to be true. Alternatively, please exclude limited wallets only.

(2) Possible to gain ownership after renouncing the contract ownership:

Owner can renounce ownership and make a contract without the owner but he can regain ownership by following the following steps:

1) Owner calls the lock function in the contract to set the current owner as _previousOwner.

2) Owner calls unlock to unlock contract and set _owner = _previousOwner.

3) Owner called renounceOwnership to leave the contract without the owner.

4) Owner calls unlock to regain ownership.

Resolution: We advise updating/removing lock and unlock functions in the contract or call renounceOwnership function first before calling lock/unlock functions.

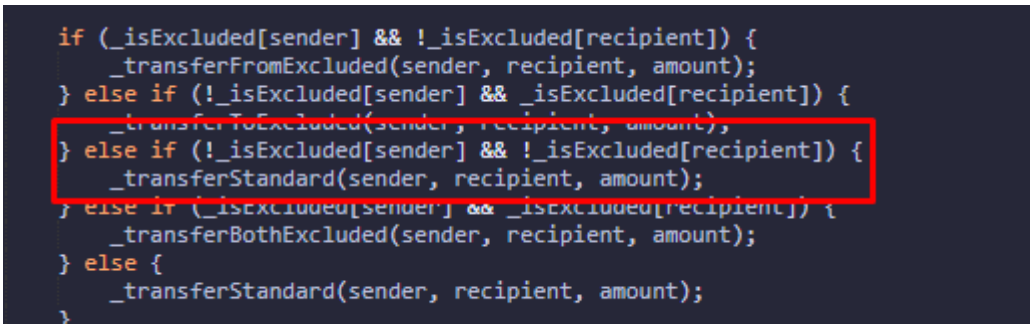
(3) Missing Events:

Following state changing functions should emit an event:

- setExcludeFromFee
- excludeAccount
- deliver
- sendETHToMarketingPool
- manualSwap
- manualSend

(4) Gas Efficiency:

When the contract enters the branch `else if (!_isExcluded[sender] && !_isExcluded[recipient])`, the contract will execute the same piece of code _transferStandard(sender, recipient, amount);



Resolution: We suggest removing this code to reduce some gas.

Very Low / Informational / Best practices:

(1) Declare variables constant:

```
string private _name = 'Mighty Zilla';
string private _symbol = 'MIGHTYZILLA';
uint8 private _decimals = 9;
```

```
uint256 private _tTotal = 100000000000000 * 10**9;
```

These variables' values will remain unchanged. so, we suggest making them constant. It is best practice and it also saves some gas. Just add a constant keyword.

(2) Use latest solidity version:

```
pragma solidity ^0.6.12;
```

Using the latest solidity will prevent any compiler level bugs.
Resolution: We suggest using 0.8.9 which is the latest version.

(3) Approve of ERC20 standard:
To prevent attack vectors regarding approve() like the one described here: https://docs.google.com/document/d/1YLPtQxZu1UAvO9cZ1O2RPXBbT0mooh4DYKjA_jp-RLM
clients SHOULD make sure to create user interfaces in such a way that they set the allowance first to 0 before setting it to another value for the same spender. THOUGH the contract itself shouldn't enforce it, to allow backwards compatibility with contracts deployed before.

(4) Visibility can be external over public:
Any functions which are not called internally, should be declared as external. This saves some gas and is considered a good practice.
https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- openTrading: Owner can open trading.
- setWalletlimit: Owner can set wallet limit.

- addDestLimit: Owner can set uniswap status true.
- removeDestLimit: Owner can set uniswap status false.
- setExcludeFromFee: Owner can set excluded from fee.
- RemoveSniper: Owner can check if the account is already blacklisted or not. then set a sniper account status true.
- amnestySniper: Owner can check if the account is already blacklisted or not. then set a sniper account status false.
- excludeAccount: Owner can check if the account is already excluded or not, set it true.
- includeAccount: Owner can check if the account is already excluded or not, set it false.
- manualSwap: Owner is able to manual swap and send in case the token is highly valued and 5M becomes too much.
- manualSend: Owner can send manual send in case the token is highly valued and 5M becomes too much.
- setSwapEnabled: Owner can set swap enabled status.
- _setTaxFee: Owner can check taxFee should be in 0 - 20 and set tax fee.
- _setbuyTaxFee: Owner can check taxFee should be in 0 - 20 and set buy tax fee.
- _setsellTaxFee: Owner can check taxFee should be in 0 - 20 and set sell tax fee.
- _setStakeFee: Owner can check stakeFee should be in 0 - 21 and set stake fee.
- _setbuystakeFee: Owner can check stakeFee should be in 0 - 21 and set buy stake fee.
- _setsellstakeFee: Owner can check stakeFee should be in 0 - 21 and set sell stake fee.
- _setMarketingPoolFee: Owner can check MarketingPoolFee should be in 0 - 21 and set marketing pool fee.
- _setTokenExchange: Owner can check set token exchange.
- _setbuyMarketingPoolFee: Owner can check MarketingPoolFee should be in 0 - 21 and set marketing pool fee.
- _setsellMarketingPoolFee: Owner can check MarketingPoolFee should be in 0 - 21 and set marketing pool fee.
- _setMarketingPoolWallet: Owner can check set marketing pool wallet.
- _setStakePoolAddress: Owner can check set stake pool address.
- _setMaxTxAmount: Owner can check set maximum tax amount.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts, but they are not critical ones. So, **it's good to go to production**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **"Secured"**.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

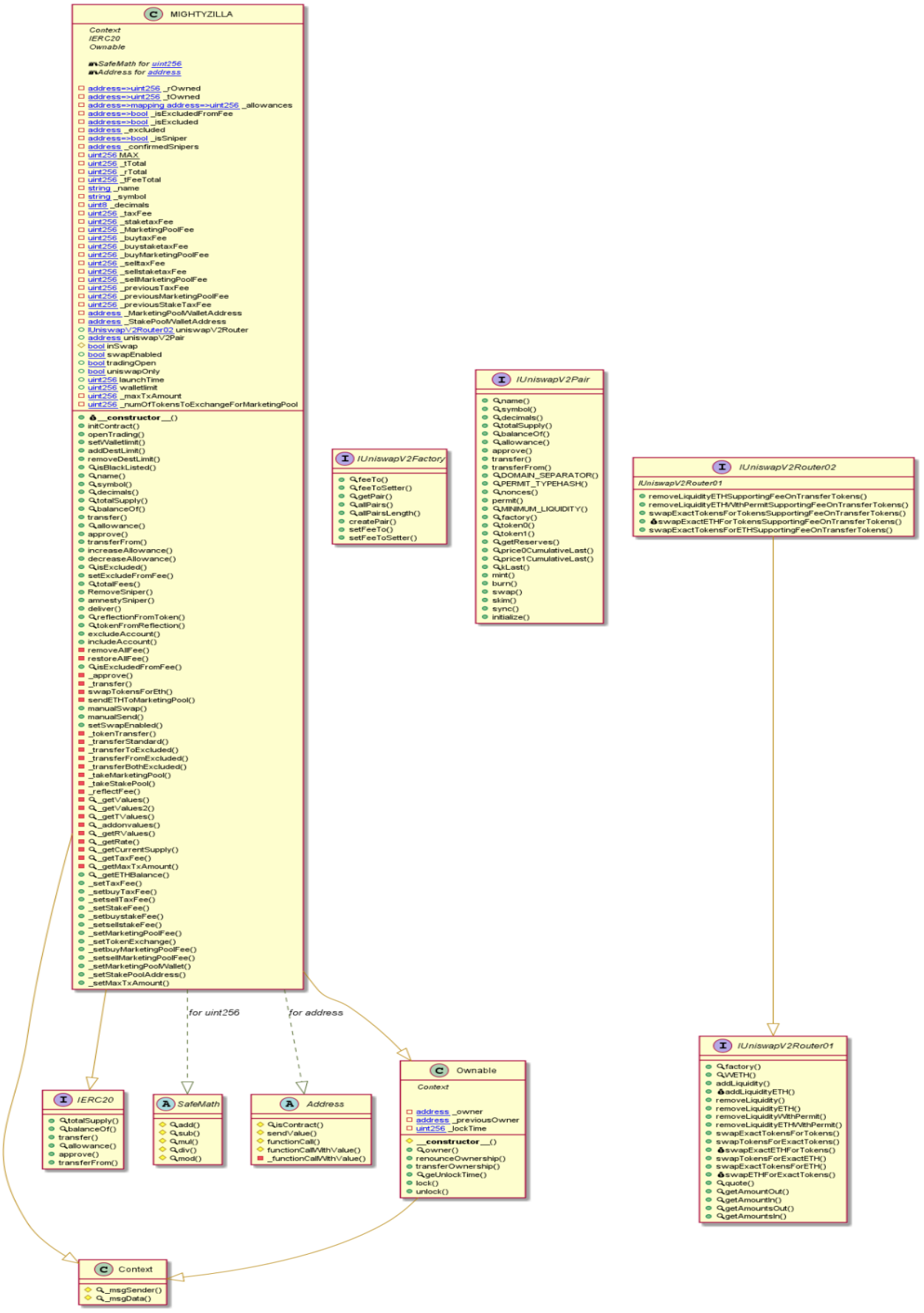
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

Appendix

Code Flow Diagram - Mighty Zilla Token



Slither Results Log

Slither log >> MIGHTYZILLA.sol

```
INFO:Detectors:
MIGHTYZILLA.sendETHToMarketingPool(uint256) (MIGHTYZILLA.sol#1178-1181) sends eth to arbitrary user
        Dangerous calls:
        - _MarketingPoolWalletAddress.transfer(amount) (MIGHTYZILLA.sol#1179)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in MIGHTYZILLA._transfer(address,address,uint256) (MIGHTYZILLA.sol#1046-1158):
        External calls:
        - swapTokensForEth(contractTokenBalance) (MIGHTYZILLA.sol#1119)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (M
IGHTYZILLA.sol#1169-1175)
        External calls sending eth:
        - sendETHToMarketingPool(address(this).balance) (MIGHTYZILLA.sol#1123)
                - _MarketingPoolWalletAddress.transfer(amount) (MIGHTYZILLA.sol#1179)
        State variables written after the call(s):
        - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
                - _rOwned[address(_StakePoolWalletAddress)] = _rOwned[address(_StakePoolWalletAddress)].add(rStakePool) (MIGHTYZILLA.sol#
1275)
                - _rOwned[address(this)] = _rOwned[address(this)].add(rMarketingPool) (MIGHTYZILLA.sol#1267)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (MIGHTYZILLA.sol#1221)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (MIGHTYZILLA.sol#1231)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (MIGHTYZILLA.sol#1244)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (MIGHTYZILLA.sol#1255)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (MIGHTYZILLA.sol#1222)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (MIGHTYZILLA.sol#1233)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (MIGHTYZILLA.sol#1245)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (MIGHTYZILLA.sol#1257)
        - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
                - _rTotal = _rTotal.sub(rFee) (MIGHTYZILLA.sol#1281)
        - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
                - _tOwned[address(_StakePoolWalletAddress)] = _tOwned[address(_StakePoolWalletAddress)].add(tStakePool) (MIGHTYZILLA.sol#
1277)
                - _tOwned[address(this)] = _tOwned[address(this)].add(tMarketingPool) (MIGHTYZILLA.sol#1269)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (MIGHTYZILLA.sol#1243)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (MIGHTYZILLA.sol#1254)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (MIGHTYZILLA.sol#1232)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (MIGHTYZILLA.sol#1256)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
MIGHTYZILLA._setTaxFee(uint256) (MIGHTYZILLA.sol#1361-1365) contains a tautology or contradiction:
        - require(bool,string)(taxFee >= 0 && taxFee <= 20,taxFee should be in 0 - 20) (MIGHTYZILLA.sol#1362)
MIGHTYZILLA._setbuyTaxFee(uint256) (MIGHTYZILLA.sol#1367-1370) contains a tautology or contradiction:
        - require(bool,string)(buytaxFee >= 0 && buytaxFee <= 20,taxFee should be in 0 - 20) (MIGHTYZILLA.sol#1368)
MIGHTYZILLA._setsellTaxFee(uint256) (MIGHTYZILLA.sol#1372-1375) contains a tautology or contradiction:
        - require(bool,string)(selltaxFee >= 0 && selltaxFee <= 20,taxFee should be in 0 - 20) (MIGHTYZILLA.sol#1373)
MIGHTYZILLA._setStakeFee(uint256) (MIGHTYZILLA.sol#1377-1381) contains a tautology or contradiction:
        - require(bool,string)(stakeFee >= 0 && stakeFee <= 21,stakeFee should be in 0 - 21) (MIGHTYZILLA.sol#1378)
MIGHTYZILLA._setbuystakeFee(uint256) (MIGHTYZILLA.sol#1382-1385) contains a tautology or contradiction:
        - require(bool,string)(buystakeFee >= 0 && buystakeFee <= 21,stakeFee should be in 0 - 21) (MIGHTYZILLA.sol#1383)
MIGHTYZILLA._setsellstakeFee(uint256) (MIGHTYZILLA.sol#1386-1389) contains a tautology or contradiction:
        - require(bool,string)(sellstakeFee >= 0 && sellstakeFee <= 21,stakeFee should be in 0 - 21) (MIGHTYZILLA.sol#1387)
MIGHTYZILLA._setMarketingPoolFee(uint256) (MIGHTYZILLA.sol#1391-1395) contains a tautology or contradiction:
        - require(bool,string)(MarketingPoolFee >= 0 && MarketingPoolFee <= 21,MarketingPoolFee should be in 0 - 21) (MIGHTYZILLA.sol#139
2)
MIGHTYZILLA._setbuyMarketingPoolFee(uint256) (MIGHTYZILLA.sol#1401-1404) contains a tautology or contradiction:
        - require(bool,string)(buyMarketingPoolFee >= 0 && buyMarketingPoolFee <= 21,MarketingPoolFee should be in 0 - 21) (MIGHTYZILLA.s
ol#1402)
MIGHTYZILLA._setsellMarketingPoolFee(uint256) (MIGHTYZILLA.sol#1406-1409) contains a tautology or contradiction:
        - require(bool,string)(sellMarketingPoolFee >= 0 && sellMarketingPoolFee <= 21,MarketingPoolFee should be in 0 - 21) (MIGHTYZILLA
.sol#1407)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-contradiction
INFO:Detectors:
MIGHTYZILLA.allowance(address,address).owner (MIGHTYZILLA.sol#911) shadows:
        - Ownable.owner() (MIGHTYZILLA.sol#376-378) (function)
MIGHTYZILLA._approve(address,address,uint256).owner (MIGHTYZILLA.sol#1038) shadows:
        - Ownable.owner() (MIGHTYZILLA.sol#376-378) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
MIGHTYZILLA.constructor(address,address).MarketingPoolWalletAddress (MIGHTYZILLA.sol#699) lacks a zero-check on :
                - _MarketingPoolWalletAddress = MarketingPoolWalletAddress (MIGHTYZILLA.sol#700)
MIGHTYZILLA.constructor(address,address).StakePoolWalletAddress (MIGHTYZILLA.sol#699) lacks a zero-check on :
                - _StakePoolWalletAddress = StakePoolWalletAddress (MIGHTYZILLA.sol#701)
MIGHTYZILLA._setMarketingPoolWallet(address).MarketingPoolWalletAddress (MIGHTYZILLA.sol#1411) lacks a zero-check on :
                - _MarketingPoolWalletAddress = MarketingPoolWalletAddress (MIGHTYZILLA.sol#1412)
MIGHTYZILLA._setStakePoolAddress(address).StakePoolAddress (MIGHTYZILLA.sol#1415) lacks a zero-check on :
```

```
                - _StakePoolWalletAddress = StakePoolAddress (MIGHTYZILLA.sol#1416)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in MIGHTYZILLA._transfer(address,address,uint256) (MIGHTYZILLA.sol#1046-1158):
        External calls:
        - swapTokensForEth(contractTokenBalance) (MIGHTYZILLA.sol#1119)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (M
IGHTYZILLA.sol#1169-1175)
        External calls sending eth:
        - sendETHToMarketingPool(address(this).balance) (MIGHTYZILLA.sol#1123)
                - _MarketingPoolWalletAddress.transfer(amount) (MIGHTYZILLA.sol#1179)
        State variables written after the call(s):
        - removeAllFee() (MIGHTYZILLA.sol#1137)
                - _MarketingPoolFee = 0 (MIGHTYZILLA.sol#1024)
        - _MarketingPoolFee = _buyMarketingPoolFee (MIGHTYZILLA.sol#1140)
        - removeAllFee() (MIGHTYZILLA.sol#1145)
                - _MarketingPoolFee = 0 (MIGHTYZILLA.sol#1024)
        - _MarketingPoolFee = _sellMarketingPoolFee (MIGHTYZILLA.sol#1148)
        - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
                - _MarketingPoolFee = _previousMarketingPoolFee (MIGHTYZILLA.sol#1030)
                - _MarketingPoolFee = 0 (MIGHTYZILLA.sol#1024)
        - removeAllFee() (MIGHTYZILLA.sol#1137)
                - _previousMarketingPoolFee = _MarketingPoolFee (MIGHTYZILLA.sol#1020)
        - removeAllFee() (MIGHTYZILLA.sol#1145)
                - _previousMarketingPoolFee = _MarketingPoolFee (MIGHTYZILLA.sol#1020)
        - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
                - _previousMarketingPoolFee = _MarketingPoolFee (MIGHTYZILLA.sol#1020)
        - removeAllFee() (MIGHTYZILLA.sol#1137)
                - _previousStakeTaxFee = _staketaxFee (MIGHTYZILLA.sol#1021)
```

```
        - removeAllFee() (MIGHTYZILLA.sol#1145)
                - _previousStakeTaxFee = _staketaxFee (MIGHTYZILLA.sol#1021)
        - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
                - _previousStakeTaxFee = _staketaxFee (MIGHTYZILLA.sol#1021)
        - removeAllFee() (MIGHTYZILLA.sol#1137)
                - _previousTaxFee = _taxFee (MIGHTYZILLA.sol#1019)
        - removeAllFee() (MIGHTYZILLA.sol#1145)
                - _previousTaxFee = _taxFee (MIGHTYZILLA.sol#1019)
        - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
                - _previousTaxFee = _taxFee (MIGHTYZILLA.sol#1019)
        - removeAllFee() (MIGHTYZILLA.sol#1137)
                - _staketaxFee = 0 (MIGHTYZILLA.sol#1025)
        - _staketaxFee = _buystaketaxFee (MIGHTYZILLA.sol#1139)
        - removeAllFee() (MIGHTYZILLA.sol#1145)
                - _staketaxFee = 0 (MIGHTYZILLA.sol#1025)
        - _staketaxFee = _sellstaketaxFee (MIGHTYZILLA.sol#1147)
        - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
                - _staketaxFee = _previousStakeTaxFee (MIGHTYZILLA.sol#1031)
                - _staketaxFee = 0 (MIGHTYZILLA.sol#1025)
        - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
                - _tFeeTotal = _tFeeTotal.add(tFee) (MIGHTYZILLA.sol#1282)
        - removeAllFee() (MIGHTYZILLA.sol#1137)
                - _taxFee = 0 (MIGHTYZILLA.sol#1023)
        - _taxFee = _buytaxFee (MIGHTYZILLA.sol#1138)
        - removeAllFee() (MIGHTYZILLA.sol#1145)
                - _taxFee = 0 (MIGHTYZILLA.sol#1023)
        - _taxFee = _selltaxFee (MIGHTYZILLA.sol#1146)
        - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
                - _taxFee = _previousTaxFee (MIGHTYZILLA.sol#1029)
                - _taxFee = 0 (MIGHTYZILLA.sol#1023)
Reentrancy in MIGHTYZILLA.constructor(address,address) (MIGHTYZILLA.sol#699-712):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (MIGHTYZILLA.so
l#705-706)
        State variables written after the call(s):
        - _isExcludedFromFee[owner()] = true (MIGHTYZILLA.sol#709)
        - _isExcludedFromFee[address(this)] = true (MIGHTYZILLA.sol#710)
        - uniswapV2Router = _uniswapV2Router (MIGHTYZILLA.sol#708)
```

```
Reentrancy in MIGHTYZILLA.transferFrom(address,address,uint256) (MIGHTYZILLA.sol#920-924):
        External calls:
        - _transfer(sender,recipient,amount) (MIGHTYZILLA.sol#921)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (M
IGHTYZILLA.sol#1169-1175)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (MIGHTYZILLA.sol#921)
                - _MarketingPoolWalletAddress.transfer(amount) (MIGHTYZILLA.sol#1179)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (MIGHTYZIL
LA.sol#922)
                - _allowances[owner][spender] = amount (MIGHTYZILLA.sol#1042)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in MIGHTYZILLA._transfer(address,address,uint256) (MIGHTYZILLA.sol#1046-1158):
        External calls:
        - swapTokensForEth(contractTokenBalance) (MIGHTYZILLA.sol#1119)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (M
IGHTYZILLA.sol#1169-1175)
        External calls sending eth:
        - sendETHToMarketingPool(address(this).balance) (MIGHTYZILLA.sol#1123)
                - _MarketingPoolWalletAddress.transfer(amount) (MIGHTYZILLA.sol#1179)
        Event emitted after the call(s):
        - Transfer(sender,recipient,tTransferAmount) (MIGHTYZILLA.sol#1226)
                - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
        - Transfer(sender,recipient,tTransferAmount) (MIGHTYZILLA.sol#1237)
                - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
        - Transfer(sender,recipient,tTransferAmount) (MIGHTYZILLA.sol#1249)
                - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
        - Transfer(sender,recipient,tTransferAmount) (MIGHTYZILLA.sol#1261)
                - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
Reentrancy in MIGHTYZILLA.constructor(address,address) (MIGHTYZILLA.sol#699-712):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (MIGHTYZILLA.so
l#705-706)
        Event emitted after the call(s):
        - Transfer(address(0),_msgSender(),_tTotal) (MIGHTYZILLA.sol#711)
Reentrancy in MIGHTYZILLA.transferFrom(address,address,uint256) (MIGHTYZILLA.sol#920-924):
```

```
        External calls:
        - _transfer(sender,recipient,amount) (MIGHTYZILLA.sol#921)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (M
IGHTYZILLA.sol#1169-1175)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (MIGHTYZILLA.sol#921)
                - _MarketingPoolWalletAddress.transfer(amount) (MIGHTYZILLA.sol#1179)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (MIGHTYZILLA.sol#1043)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (M
IGHTYZILLA.sol#922)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Ownable.unlock() (MIGHTYZILLA.sol#423-428) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now > _lockTime,Contract is locked until 7 days) (MIGHTYZILLA.sol#425)
MIGHTYZILLA._transfer(address,address,uint256) (MIGHTYZILLA.sol#1046-1158) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp < launchTime + 5 (MIGHTYZILLA.sol#1091)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (MIGHTYZILLA.sol#240-249) uses assembly
        - INLINE ASM (MIGHTYZILLA.sol#247)
Address._functionCallWithValue(address,bytes,uint256,string) (MIGHTYZILLA.sol#333-354) uses assembly
        - INLINE ASM (MIGHTYZILLA.sol#346-349)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (MIGHTYZILLA.sol#333-354) is never used and should be removed
Address.functionCall(address,bytes) (MIGHTYZILLA.sol#293-295) is never used and should be removed
Address.functionCall(address,bytes,string) (MIGHTYZILLA.sol#303-305) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (MIGHTYZILLA.sol#318-320) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (MIGHTYZILLA.sol#328-331) is never used and should be removed
Address.isContract(address) (MIGHTYZILLA.sol#240-249) is never used and should be removed
Address.sendValue(address,uint256) (MIGHTYZILLA.sol#267-273) is never used and should be removed
Context._msgData() (MIGHTYZILLA.sol#9-12) is never used and should be removed
```

```
MIGHTYZILLA._getMaxTxAmount() (MIGHTYZILLA.sol#1353-1355) is never used and should be removed
MIGHTYZILLA._getTaxFee() (MIGHTYZILLA.sol#1349-1351) is never used and should be removed
SafeMath.mod(uint256,uint256) (MIGHTYZILLA.sol#200-202) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (MIGHTYZILLA.sol#216-219) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
MIGHTYZILLA._rTotal (MIGHTYZILLA.sol#651) is set pre-construction with a non-constant function or state variable:
        - (MAX - (MAX % _tTotal))
MIGHTYZILLA._previousTaxFee (MIGHTYZILLA.sol#670) is set pre-construction with a non-constant function or state variable:
        - _taxFee
MIGHTYZILLA._previousMarketingPoolFee (MIGHTYZILLA.sol#671) is set pre-construction with a non-constant function or state variable:
        - _MarketingPoolFee
MIGHTYZILLA._previousStakeTaxFee (MIGHTYZILLA.sol#672) is set pre-construction with a non-constant function or state variable:
        - _staketaxFee
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state-variables
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (MIGHTYZILLA.sol#267-273):
        - (success) = recipient.call{value: amount}() (MIGHTYZILLA.sol#271)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (MIGHTYZILLA.sol#333-354):
        - (success,returndata) = target.call{value: weiValue}(data) (MIGHTYZILLA.sol#337)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (MIGHTYZILLA.sol#462) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (MIGHTYZILLA.sol#463) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (MIGHTYZILLA.sol#480) is not in mixedCase
Function IUniswapV2Router01.WETH() (MIGHTYZILLA.sol#500) is not in mixedCase
Parameter MIGHTYZILLA.setWalletlimit(uint256)._walletlimit (MIGHTYZILLA.sol#868) is not in mixedCase
Function MIGHTYZILLA.RemoveSniper(address) (MIGHTYZILLA.sol#948-953) is not in mixedCase
Function MIGHTYZILLA._getETHBalance() (MIGHTYZILLA.sol#1357-1359) is not in mixedCase
Function MIGHTYZILLA._setTaxFee(uint256) (MIGHTYZILLA.sol#1361-1365) is not in mixedCase
Function MIGHTYZILLA._setbuyTaxFee(uint256) (MIGHTYZILLA.sol#1367-1370) is not in mixedCase
Function MIGHTYZILLA._setsellTaxFee(uint256) (MIGHTYZILLA.sol#1372-1375) is not in mixedCase
Function MIGHTYZILLA._setStakeFee(uint256) (MIGHTYZILLA.sol#1377-1381) is not in mixedCase
Function MIGHTYZILLA._setbuystakeFee(uint256) (MIGHTYZILLA.sol#1382-1385) is not in mixedCase
Function MIGHTYZILLA._setsellstakeFee(uint256) (MIGHTYZILLA.sol#1386-1389) is not in mixedCase
Function MIGHTYZILLA._setMarketingPoolFee(uint256) (MIGHTYZILLA.sol#1391-1395) is not in mixedCase
Parameter MIGHTYZILLA._setMarketingPoolFee(uint256).MarketingPoolFee (MIGHTYZILLA.sol#1391) is not in mixedCase
Function MIGHTYZILLA._setTokenExchange(uint256) (MIGHTYZILLA.sol#1397-1399) is not in mixedCase
```

```
Parameter MIGHTYZILLA._setTokenExchange(uint256).TokenExchange (MIGHTYZILLA.sol#1397) is not in mixedCase
Function MIGHTYZILLA._setbuyMarketingPoolFee(uint256) (MIGHTYZILLA.sol#1401-1404) is not in mixedCase
Function MIGHTYZILLA._setsellMarketingPoolFee(uint256) (MIGHTYZILLA.sol#1406-1409) is not in mixedCase
Function MIGHTYZILLA._setMarketingPoolWallet(address) (MIGHTYZILLA.sol#1411-1413) is not in mixedCase
Parameter MIGHTYZILLA._setMarketingPoolWallet(address).MarketingPoolWalletAddress (MIGHTYZILLA.sol#1411) is not in mixedCase
Function MIGHTYZILLA._setStakePoolAddress(address) (MIGHTYZILLA.sol#1415-1417) is not in mixedCase
Parameter MIGHTYZILLA._setStakePoolAddress(address).StakePoolAddress (MIGHTYZILLA.sol#1415) is not in mixedCase
Function MIGHTYZILLA._setMaxTxAmount(uint256) (MIGHTYZILLA.sol#1419-1421) is not in mixedCase
Variable MIGHTYZILLA._MarketingPoolFee (MIGHTYZILLA.sol#662) is not in mixedCase
Variable MIGHTYZILLA._MarketingPoolWalletAddress (MIGHTYZILLA.sol#674) is not in mixedCase
Variable MIGHTYZILLA._StakePoolWalletAddress (MIGHTYZILLA.sol#675) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (MIGHTYZILLA.sol#10)" inContext (MIGHTYZILLA.sol#4-13)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Reentrancy in MIGHTYZILLA._transfer(address,address,uint256) (MIGHTYZILLA.sol#1046-1158):
        External calls:
        - sendETHToMarketingPool(address(this).balance) (MIGHTYZILLA.sol#1123)
                - _MarketingPoolWalletAddress.transfer(amount) (MIGHTYZILLA.sol#1179)
        State variables written after the call(s):
        - removeAllFee() (MIGHTYZILLA.sol#1137)
                - _MarketingPoolFee = 0 (MIGHTYZILLA.sol#1024)
        - _MarketingPoolFee = _buyMarketingPoolFee (MIGHTYZILLA.sol#1140)
        - removeAllFee() (MIGHTYZILLA.sol#1145)
                - _MarketingPoolFee = 0 (MIGHTYZILLA.sol#1024)
        - _MarketingPoolFee = _sellMarketingPoolFee (MIGHTYZILLA.sol#1148)
        - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
                - _MarketingPoolFee = _previousMarketingPoolFee (MIGHTYZILLA.sol#1030)
                - _MarketingPoolFee = 0 (MIGHTYZILLA.sol#1024)
        - removeAllFee() (MIGHTYZILLA.sol#1137)
                - _previousMarketingPoolFee = _MarketingPoolFee (MIGHTYZILLA.sol#1020)
        - removeAllFee() (MIGHTYZILLA.sol#1145)
                - _previousMarketingPoolFee = _MarketingPoolFee (MIGHTYZILLA.sol#1020)
        - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
                - _previousMarketingPoolFee = _MarketingPoolFee (MIGHTYZILLA.sol#1020)
        - removeAllFee() (MIGHTYZILLA.sol#1137)
                - _previousStakeTaxFee = _staketaxFee (MIGHTYZILLA.sol#1021)
```

```
                - _previousStakeTaxFee = _staketaxFee (MIGHTYZILLA.sol#1021)
        - removeAllFee() (MIGHTYZILLA.sol#1145)
                - _previousStakeTaxFee = _staketaxFee (MIGHTYZILLA.sol#1021)
        - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
                - _previousStakeTaxFee = _staketaxFee (MIGHTYZILLA.sol#1021)
        - removeAllFee() (MIGHTYZILLA.sol#1137)
                - _previousTaxFee = _taxFee (MIGHTYZILLA.sol#1019)
        - removeAllFee() (MIGHTYZILLA.sol#1145)
                - _previousTaxFee = _taxFee (MIGHTYZILLA.sol#1019)
        - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
                - _previousTaxFee = _taxFee (MIGHTYZILLA.sol#1019)
        - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
                - _rOwned[address(_StakePoolWalletAddress)] = _rOwned[address(_StakePoolWalletAddress)].add(rStakePool) (MIGHTYZILLA.sol#
1275)
                - _rOwned[address(this)] = _rOwned[address(this)].add(rMarketingPool) (MIGHTYZILLA.sol#1267)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (MIGHTYZILLA.sol#1221)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (MIGHTYZILLA.sol#1231)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (MIGHTYZILLA.sol#1244)
                - _rOwned[sender] = _rOwned[sender].sub(rAmount) (MIGHTYZILLA.sol#1255)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (MIGHTYZILLA.sol#1222)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (MIGHTYZILLA.sol#1233)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (MIGHTYZILLA.sol#1245)
                - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (MIGHTYZILLA.sol#1257)
        - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
                - _rTotal = _rTotal.sub(rFee) (MIGHTYZILLA.sol#1281)
        - removeAllFee() (MIGHTYZILLA.sol#1137)
                - _staketaxFee = 0 (MIGHTYZILLA.sol#1025)
        - _staketaxFee = _buystaketaxFee (MIGHTYZILLA.sol#1139)
        - removeAllFee() (MIGHTYZILLA.sol#1145)
                - _staketaxFee = 0 (MIGHTYZILLA.sol#1025)
        - _staketaxFee = _sellstaketaxFee (MIGHTYZILLA.sol#1147)
        - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
                - _staketaxFee = _previousStakeTaxFee (MIGHTYZILLA.sol#1031)
```

```
                - _staketaxFee = 0 (MIGHTYZILLA.sol#1025)
        - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
                - _tFeeTotal = _tFeeTotal.add(tFee) (MIGHTYZILLA.sol#1282)
        - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
                - _tOwned[address(_StakePoolWalletAddress)] = _tOwned[address(_StakePoolWalletAddress)].add(tStakePool) (MIGHTYZILLA.sol#
1277)
                - _tOwned[address(this)] = _tOwned[address(this)].add(tMarketingPool) (MIGHTYZILLA.sol#1269)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (MIGHTYZILLA.sol#1243)
                - _tOwned[sender] = _tOwned[sender].sub(tAmount) (MIGHTYZILLA.sol#1254)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (MIGHTYZILLA.sol#1232)
                - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (MIGHTYZILLA.sol#1256)
        - removeAllFee() (MIGHTYZILLA.sol#1137)
                - _taxFee = 0 (MIGHTYZILLA.sol#1023)
        - _taxFee = _buytaxFee (MIGHTYZILLA.sol#1138)
        - removeAllFee() (MIGHTYZILLA.sol#1145)
                - _taxFee = 0 (MIGHTYZILLA.sol#1023)
        - _taxFee = _selltaxFee (MIGHTYZILLA.sol#1146)
        - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
                - _taxFee = _previousTaxFee (MIGHTYZILLA.sol#1029)
                - _taxFee = 0 (MIGHTYZILLA.sol#1023)
        Event emitted after the call(s):
        - Transfer(sender,recipient,tTransferAmount) (MIGHTYZILLA.sol#1226)
                - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
        - Transfer(sender,recipient,tTransferAmount) (MIGHTYZILLA.sol#1237)
                - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
        - Transfer(sender,recipient,tTransferAmount) (MIGHTYZILLA.sol#1249)
                - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
        - Transfer(sender,recipient,tTransferAmount) (MIGHTYZILLA.sol#1261)
                - _tokenTransfer(sender,recipient,amount,takeFee) (MIGHTYZILLA.sol#1155)
Reentrancy in MIGHTYZILLA.transferFrom(address,address,uint256) (MIGHTYZILLA.sol#920-924):
        External calls:
        - _transfer(sender,recipient,amount) (MIGHTYZILLA.sol#921)
                - _MarketingPoolWalletAddress.transfer(amount) (MIGHTYZILLA.sol#1179)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (MIGHTYZIL
LA.sol#922)
                - _allowances[owner][spender] = amount (MIGHTYZILLA.sol#1042)
        Event emitted after the call(s):
```

```
        - Approval(owner,spender,amount) (MIGHTYZILLA.sol#1043)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (M
IGHTYZILLA.sol#922)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (MIGHTYZILLA.sol
#504) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (
MIGHTYZILLA.sol#505)
Variable MIGHTYZILLA._getRValues(uint256,uint256,uint256,uint256,uint256).rMarketingPool (MIGHTYZILLA.sol#1325) is too similar to MIGHTYZ
ILLA._transferFromExcluded(address,address,uint256).tMarketingPool (MIGHTYZILLA.sol#1241)
Variable MIGHTYZILLA._getRValues(uint256,uint256,uint256,uint256,uint256).rMarketingPool (MIGHTYZILLA.sol#1325) is too similar to MIGHTYZ
ILLA._transferToExcluded(address,address,uint256).tMarketingPool (MIGHTYZILLA.sol#1230)
Variable MIGHTYZILLA._getRValues(uint256,uint256,uint256,uint256,uint256).rMarketingPool (MIGHTYZILLA.sol#1325) is too similar to MIGHTYZ
ILLA._getRValues(uint256,uint256,uint256,uint256,uint256).tMarketingPool (MIGHTYZILLA.sol#1322)
Variable MIGHTYZILLA._getRValues(uint256,uint256,uint256,uint256,uint256).rMarketingPool (MIGHTYZILLA.sol#1325) is too similar to MIGHTYZ
ILLA._transferBothExcluded(address,address,uint256).tMarketingPool (MIGHTYZILLA.sol#1253)
Variable MIGHTYZILLA._getRValues(uint256,uint256,uint256,uint256,uint256).rMarketingPool (MIGHTYZILLA.sol#1325) is too similar to MIGHTYZ
ILLA._getTValues(uint256,uint256,uint256,uint256).tMarketingPool (MIGHTYZILLA.sol#1309)
Variable MIGHTYZILLA._getRValues(uint256,uint256,uint256,uint256,uint256).rMarketingPool (MIGHTYZILLA.sol#1325) is too similar to MIGHTYZ
ILLA._takeMarketingPool(uint256).tMarketingPool (MIGHTYZILLA.sol#1264)
Variable MIGHTYZILLA._getRValues(uint256,uint256,uint256,uint256,uint256).rMarketingPool (MIGHTYZILLA.sol#1325) is too similar to MIGHTYZ
ILLA._transferStandard(address,address,uint256).tMarketingPool (MIGHTYZILLA.sol#1220)
Variable MIGHTYZILLA._getRValues(uint256,uint256,uint256,uint256,uint256).rMarketingPool (MIGHTYZILLA.sol#1325) is too similar to MIGHTYZ
ILLA._getValues(uint256).tMarketingPool (MIGHTYZILLA.sol#1289)
Variable MIGHTYZILLA.reflectionFromToken(uint256,bool).rTransferAmount (MIGHTYZILLA.sol#982) is too similar to MIGHTYZILLA._transferFromE
xcluded(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1241)
Variable MIGHTYZILLA._transferBothExcluded(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1253) is too similar to MIGHTYZILLA.
_transferToExcluded(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1230)
Variable MIGHTYZILLA.reflectionFromToken(uint256,bool).rTransferAmount (MIGHTYZILLA.sol#982) is too similar to MIGHTYZILLA._transferStand
ard(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1220)
Variable MIGHTYZILLA.reflectionFromToken(uint256,bool).rTransferAmount (MIGHTYZILLA.sol#982) is too similar to MIGHTYZILLA._transferToExc
luded(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1230)
Variable MIGHTYZILLA._transferBothExcluded(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1253) is too similar to MIGHTYZILLA.
_transferFromExcluded(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1241)
Variable MIGHTYZILLA._transferBothExcluded(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1253) is too similar to MIGHTYZILLA.
_transferStandard(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1220)
Variable MIGHTYZILLA.reflectionFromToken(uint256,bool).rTransferAmount (MIGHTYZILLA.sol#982) is too similar to MIGHTYZILLA._getValues(uin
t256).tTransferAmount (MIGHTYZILLA.sol#1289)
```

```
Variable MIGHTYZILLA._transferBothExcluded(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1253) is too similar to MIGHTYZILLA.
_transferBothExcluded(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1253)
Variable MIGHTYZILLA._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (MIGHTYZILLA.sol#1327) is too similar to MIGHTY
ZILLA._transferToExcluded(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1230)
Variable MIGHTYZILLA._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (MIGHTYZILLA.sol#1327) is too similar to MIGHTY
ZILLA._transferFromExcluded(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1241)
Variable MIGHTYZILLA._transferToExcluded(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1230) is too similar to MIGHTYZILLA._t
ransferStandard(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1220)
Variable MIGHTYZILLA._transferToExcluded(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1230) is too similar to MIGHTYZILLA._t
ransferToExcluded(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1230)
Variable MIGHTYZILLA._transferToExcluded(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1230) is too similar to MIGHTYZILLA._g
etValues(uint256).tTransferAmount (MIGHTYZILLA.sol#1289)
Variable MIGHTYZILLA._transferBothExcluded(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1253) is too similar to MIGHTYZILLA.
_getValues(uint256).tTransferAmount (MIGHTYZILLA.sol#1289)
Variable MIGHTYZILLA._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (MIGHTYZILLA.sol#1327) is too similar to MIGHTY
ZILLA._transferStandard(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1220)
Variable MIGHTYZILLA._transferStandard(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1220) is too similar to MIGHTYZILLA._tra
nsferStandard(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1220)
Variable MIGHTYZILLA._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (MIGHTYZILLA.sol#1327) is too similar to MIGHTY
ZILLA._getValues(uint256).tTransferAmount (MIGHTYZILLA.sol#1289)
Variable MIGHTYZILLA.reflectionFromToken(uint256,bool).rTransferAmount (MIGHTYZILLA.sol#982) is too similar to MIGHTYZILLA._transferBothE
xcluded(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1253)
Variable MIGHTYZILLA._takeMarketingPool(uint256).rMarketingPool (MIGHTYZILLA.sol#1266) is too similar to MIGHTYZILLA._transferStandard(ad
dress,address,uint256).tMarketingPool (MIGHTYZILLA.sol#1220)
Variable MIGHTYZILLA._takeMarketingPool(uint256).rMarketingPool (MIGHTYZILLA.sol#1266) is too similar to MIGHTYZILLA._transferToExcluded(
address,address,uint256).tMarketingPool (MIGHTYZILLA.sol#1230)
Variable MIGHTYZILLA._takeMarketingPool(uint256).rMarketingPool (MIGHTYZILLA.sol#1266) is too similar to MIGHTYZILLA._getTValues(uint256,
uint256,uint256,uint256).tMarketingPool (MIGHTYZILLA.sol#1309)
Variable MIGHTYZILLA._getRValues(uint256,uint256,uint256,uint256,uint256).rMarketingPool (MIGHTYZILLA.sol#1325) is too similar to MIGHTYZ
ILLA._getValues2(uint256,uint256,uint256,uint256).tMarketingPool (MIGHTYZILLA.sol#1296)
Variable MIGHTYZILLA._takeMarketingPool(uint256).rMarketingPool (MIGHTYZILLA.sol#1266) is too similar to MIGHTYZILLA._getValues2(uint256,
uint256,uint256,uint256).tMarketingPool (MIGHTYZILLA.sol#1296)
Variable MIGHTYZILLA._takeMarketingPool(uint256).rMarketingPool (MIGHTYZILLA.sol#1266) is too similar to MIGHTYZILLA._getRValues(uint256,
uint256,uint256,uint256,uint256).tMarketingPool (MIGHTYZILLA.sol#1322)
```

```
Variable MIGHTYZILLA._takeMarketingPool(uint256).rMarketingPool (MIGHTYZILLA.sol#1266) is too similar to MIGHTYZILLA._transferBothExclude
d(address,address,uint256).tMarketingPool (MIGHTYZILLA.sol#1253)
Variable MIGHTYZILLA._takeMarketingPool(uint256).rMarketingPool (MIGHTYZILLA.sol#1266) is too similar to MIGHTYZILLA._takeMarketingPool(u
int256).tMarketingPool (MIGHTYZILLA.sol#1264)
Variable MIGHTYZILLA._takeMarketingPool(uint256).rMarketingPool (MIGHTYZILLA.sol#1266) is too similar to MIGHTYZILLA._transferFromExclude
d(address,address,uint256).tMarketingPool (MIGHTYZILLA.sol#1241)
Variable MIGHTYZILLA._takeMarketingPool(uint256).rMarketingPool (MIGHTYZILLA.sol#1266) is too similar to MIGHTYZILLA._getValues(uint256).
tMarketingPool (MIGHTYZILLA.sol#1289)
Variable MIGHTYZILLA._transferToExcluded(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1230) is too similar to MIGHTYZILLA._t
ransferBothExcluded(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1253)
Variable MIGHTYZILLA.reflectionFromToken(uint256,bool).rTransferAmount (MIGHTYZILLA.sol#982) is too similar to MIGHTYZILLA._addonvalues(u
int256,uint256,uint256,uint256).tTransferAmount (MIGHTYZILLA.sol#1317)
Variable MIGHTYZILLA._getValues(uint256).rTransferAmount (MIGHTYZILLA.sol#1291) is too similar to MIGHTYZILLA._transferBothExcluded(addre
ss,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1253)
Variable MIGHTYZILLA._transferFromExcluded(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1241) is too similar to MIGHTYZILLA.
_addonvalues(uint256,uint256,uint256,uint256).tTransferAmount (MIGHTYZILLA.sol#1317)
Variable MIGHTYZILLA._getValues2(uint256,uint256,uint256,uint256).rTransferAmount (MIGHTYZILLA.sol#1300) is too similar to MIGHTYZILLA._a
ddonvalues(uint256,uint256,uint256,uint256).tTransferAmount (MIGHTYZILLA.sol#1317)
Variable MIGHTYZILLA._getValues(uint256).rTransferAmount (MIGHTYZILLA.sol#1291) is too similar to MIGHTYZILLA._transferFromExcluded(addre
ss,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1241)
Variable MIGHTYZILLA._getValues2(uint256,uint256,uint256,uint256).rTransferAmount (MIGHTYZILLA.sol#1300) is too similar to MIGHTYZILLA._t
ransferToExcluded(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1230)
Variable MIGHTYZILLA._transferStandard(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1220) is too similar to MIGHTYZILLA._add
onvalues(uint256,uint256,uint256,uint256).tTransferAmount (MIGHTYZILLA.sol#1317)
Variable MIGHTYZILLA._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (MIGHTYZILLA.sol#1327) is too similar to MIGHTY
ZILLA._addonvalues(uint256,uint256,uint256,uint256).tTransferAmount (MIGHTYZILLA.sol#1317)
Variable MIGHTYZILLA._transferToExcluded(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1230) is too similar to MIGHTYZILLA._t
ransferFromExcluded(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1241)
Variable MIGHTYZILLA._getValues(uint256).rTransferAmount (MIGHTYZILLA.sol#1291) is too similar to MIGHTYZILLA._getValues(uint256).tTransf
erAmount (MIGHTYZILLA.sol#1289)
Variable MIGHTYZILLA._getValues2(uint256,uint256,uint256,uint256).rTransferAmount (MIGHTYZILLA.sol#1300) is too similar to MIGHTYZILLA._t
ransferStandard(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1220)
Variable MIGHTYZILLA._transferToExcluded(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1230) is too similar to MIGHTYZILLA._a
ddonvalues(uint256,uint256,uint256,uint256).tTransferAmount (MIGHTYZILLA.sol#1317)
Variable MIGHTYZILLA._transferStandard(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1220) is too similar to MIGHTYZILLA._tra
nsferFromExcluded(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1241)
Variable MIGHTYZILLA._transferFromExcluded(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1241) is too similar to MIGHTYZILLA.
_transferToExcluded(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1230)
```

```
Variable MIGHTYZILLA._transferBothExcluded(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1253) is too similar to MIGHTYZILLA.
_addonvalues(uint256,uint256,uint256,uint256).tTransferAmount (MIGHTYZILLA.sol#1317)
Variable MIGHTYZILLA._getValues(uint256).rTransferAmount (MIGHTYZILLA.sol#1291) is too similar to MIGHTYZILLA._addonvalues(uint256,uint25
6,uint256,uint256).tTransferAmount (MIGHTYZILLA.sol#1317)
Variable MIGHTYZILLA._getValues2(uint256,uint256,uint256,uint256).rTransferAmount (MIGHTYZILLA.sol#1300) is too similar to MIGHTYZILLA._t
ransferFromExcluded(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1241)
Variable MIGHTYZILLA._getValues(uint256).rTransferAmount (MIGHTYZILLA.sol#1291) is too similar to MIGHTYZILLA._transferToExcluded(address
,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1230)
Variable MIGHTYZILLA._transferStandard(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1220) is too similar to MIGHTYZILLA._get
Values(uint256).tTransferAmount (MIGHTYZILLA.sol#1289)
Variable MIGHTYZILLA._transferFromExcluded(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1241) is too similar to MIGHTYZILLA.
_transferStandard(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1220)
Variable MIGHTYZILLA._getValues2(uint256,uint256,uint256,uint256).rTransferAmount (MIGHTYZILLA.sol#1300) is too similar to MIGHTYZILLA._g
etValues(uint256).tTransferAmount (MIGHTYZILLA.sol#1289)
Variable MIGHTYZILLA._getValues(uint256).rTransferAmount (MIGHTYZILLA.sol#1291) is too similar to MIGHTYZILLA._transferStandard(address,a
ddress,uint256).tTransferAmount (MIGHTYZILLA.sol#1220)
Variable MIGHTYZILLA._transferFromExcluded(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1241) is too similar to MIGHTYZILLA.
_transferFromExcluded(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1241)
Variable MIGHTYZILLA._transferStandard(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1220) is too similar to MIGHTYZILLA._tra
nsferToExcluded(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1230)
Variable MIGHTYZILLA._transferFromExcluded(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1241) is too similar to MIGHTYZILLA.
_transferBothExcluded(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1253)
Variable MIGHTYZILLA._transferFromExcluded(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1241) is too similar to MIGHTYZILLA.
_getValues(uint256).tTransferAmount (MIGHTYZILLA.sol#1289)
Variable MIGHTYZILLA._getValues2(uint256,uint256,uint256,uint256).rTransferAmount (MIGHTYZILLA.sol#1300) is too similar to MIGHTYZILLA._t
ransferBothExcluded(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1253)
Variable MIGHTYZILLA._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (MIGHTYZILLA.sol#1327) is too similar to MIGHTY
ZILLA._transferBothExcluded(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1253)
Variable MIGHTYZILLA._transferStandard(address,address,uint256).rTransferAmount (MIGHTYZILLA.sol#1220) is too similar to MIGHTYZILLA._tra
nsferBothExcluded(address,address,uint256).tTransferAmount (MIGHTYZILLA.sol#1253)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
MIGHTYZILLA.initContract() (MIGHTYZILLA.sol#714-858) uses literals with too many digits:
        - _isSniper[address(0x000000000000084e91743124a982076C59f10084)] = true (MIGHTYZILLA.sol#759)
MIGHTYZILLA.initContract() (MIGHTYZILLA.sol#714-858) uses literals with too many digits:
        - _confirmedSnipers.push(address(0x000000000000084e91743124a982076C59f10084)) (MIGHTYZILLA.sol#760)
MIGHTYZILLA.initContract() (MIGHTYZILLA.sol#714-858) uses literals with too many digits:
        - _isSniper[address(0x00000005804B22091aa9830E50459A15E7C9241)] = true (MIGHTYZILLA.sol#768)
```

```
MIGHTYZILLA.initContract() (MIGHTYZILLA.sol#714-858) uses literals with too many digits:
        - _confirmedSnipers.push(address(0x00000005804B22091aa9830E50459A15E7C9241)) (MIGHTYZILLA.sol#769)
MIGHTYZILLA.initContract() (MIGHTYZILLA.sol#714-858) uses literals with too many digits:
        - _isSniper[address(0x000000000007673393729D5618DC555FD13f9aA)] = true (MIGHTYZILLA.sol#777)
MIGHTYZILLA.initContract() (MIGHTYZILLA.sol#714-858) uses literals with too many digits:
        - _confirmedSnipers.push(address(0x000000000007673393729D5618DC555FD13f9aA)) (MIGHTYZILLA.sol#778)
MIGHTYZILLA.initContract() (MIGHTYZILLA.sol#714-858) uses literals with too many digits:
        - _isSniper[address(0x00000000000003441d59DdE9A90BFfb1CD3fABf1)] = true (MIGHTYZILLA.sol#780)
MIGHTYZILLA.initContract() (MIGHTYZILLA.sol#714-858) uses literals with too many digits:
        - _confirmedSnipers.push(address(0x00000000000003441d59DdE9A90BFfb1CD3fABf1)) (MIGHTYZILLA.sol#781)
MIGHTYZILLA.initContract() (MIGHTYZILLA.sol#714-858) uses literals with too many digits:
        - _isSniper[address(0x000000917de6037d52b1F0a306eeCD208405f7cd)] = true (MIGHTYZILLA.sol#786)
MIGHTYZILLA.initContract() (MIGHTYZILLA.sol#714-858) uses literals with too many digits:
        - _confirmedSnipers.push(address(0x000000917de6037d52b1F0a306eeCD208405f7cd)) (MIGHTYZILLA.sol#787)
MIGHTYZILLA.initContract() (MIGHTYZILLA.sol#714-858) uses literals with too many digits:
        - _isSniper[address(0x00000000003b3cc22aF3aE1EAc0440BcEe416B40)] = true (MIGHTYZILLA.sol#834)
MIGHTYZILLA.initContract() (MIGHTYZILLA.sol#714-858) uses literals with too many digits:
        - _confirmedSnipers.push(address(0x00000000003b3cc22aF3aE1EAc0440BcEe416B40)) (MIGHTYZILLA.sol#835)
MIGHTYZILLA.slitherConstructorVariables() (MIGHTYZILLA.sol#633-1423) uses literals with too many digits:
        - _tTotal = 100000000000000 * 10 ** 9 (MIGHTYZILLA.sol#650)
MIGHTYZILLA.slitherConstructorVariables() (MIGHTYZILLA.sol#633-1423) uses literals with too many digits:
        - walletlimit = 1000000000000 * 10 ** 9 (MIGHTYZILLA.sol#686)
MIGHTYZILLA.slitherConstructorVariables() (MIGHTYZILLA.sol#633-1423) uses literals with too many digits:
        - _maxTxAmount = 500000000000 * 10 ** 9 (MIGHTYZILLA.sol#687)
MIGHTYZILLA.slitherConstructorVariables() (MIGHTYZILLA.sol#633-1423) uses literals with too many digits:
        - _numOfTokensToExchangeForMarketingPool = 100000000 * 10 ** 9 (MIGHTYZILLA.sol#689)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
MIGHTYZILLA._decimals (MIGHTYZILLA.sol#656) should be constant
MIGHTYZILLA._name (MIGHTYZILLA.sol#654) should be constant
MIGHTYZILLA._symbol (MIGHTYZILLA.sol#655) should be constant
MIGHTYZILLA._tTotal (MIGHTYZILLA.sol#650) should be constant
MIGHTYZILLA.inSwap (MIGHTYZILLA.sol#680) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
```

```
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (MIGHTYZILLA.sol#395-398)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (MIGHTYZILLA.sol#404-408)
geUnlockTime() should be declared external:
        - Ownable.geUnlockTime() (MIGHTYZILLA.sol#410-412)
lock(uint256) should be declared external:
        - Ownable.lock(uint256) (MIGHTYZILLA.sol#415-420)
unlock() should be declared external:
        - Ownable.unlock() (MIGHTYZILLA.sol#423-428)
isBlackListed(address) should be declared external:
        - MIGHTYZILLA.isBlackListed(address) (MIGHTYZILLA.sol#881-883)
name() should be declared external:
        - MIGHTYZILLA.name() (MIGHTYZILLA.sol#885-887)
symbol() should be declared external:
        - MIGHTYZILLA.symbol() (MIGHTYZILLA.sol#889-891)
decimals() should be declared external:
        - MIGHTYZILLA.decimals() (MIGHTYZILLA.sol#893-895)
totalSupply() should be declared external:
        - MIGHTYZILLA.totalSupply() (MIGHTYZILLA.sol#897-899)
transfer(address,uint256) should be declared external:
        - MIGHTYZILLA.transfer(address,uint256) (MIGHTYZILLA.sol#906-909)
allowance(address,address) should be declared external:
        - MIGHTYZILLA.allowance(address,address) (MIGHTYZILLA.sol#911-913)
approve(address,uint256) should be declared external:
        - MIGHTYZILLA.approve(address,uint256) (MIGHTYZILLA.sol#915-918)
transferFrom(address,address,uint256) should be declared external:
        - MIGHTYZILLA.transferFrom(address,address,uint256) (MIGHTYZILLA.sol#920-924)
increaseAllowance(address,uint256) should be declared external:
        - MIGHTYZILLA.increaseAllowance(address,uint256) (MIGHTYZILLA.sol#926-929)
decreaseAllowance(address,uint256) should be declared external:
        - MIGHTYZILLA.decreaseAllowance(address,uint256) (MIGHTYZILLA.sol#931-934)
isExcluded(address) should be declared external:
        - MIGHTYZILLA.isExcluded(address) (MIGHTYZILLA.sol#936-938)
totalFees() should be declared external:
        - MIGHTYZILLA.totalFees() (MIGHTYZILLA.sol#944-946)
deliver(uint256) should be declared external:
```

```
deliver(uint256) should be declared external:
        - MIGHTYZILLA.deliver(uint256) (MIGHTYZILLA.sol#967-974)
reflectionFromToken(uint256,bool) should be declared external:
        - MIGHTYZILLA.reflectionFromToken(uint256,bool) (MIGHTYZILLA.sol#976-985)
isExcludedFromFee(address) should be declared external:
        - MIGHTYZILLA.isExcludedFromFee(address) (MIGHTYZILLA.sol#1034-1036)
_getETHBalance() should be declared external:
        - MIGHTYZILLA._getETHBalance() (MIGHTYZILLA.sol#1357-1359)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MIGHTYZILLA.sol analyzed (10 contracts with 75 detectors), 185 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

MIGHTYZILLA.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address._functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 333:8:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in MIGHTYZILLA.(address payable,address payable): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 699:8:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in MIGHTYZILLA.swapTokensForEth(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1160:8:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 863:21:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 1091:16:

Low level calls:

Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.

more

Pos: 271:31:

Email: audit@EtherAuthority.io

Gas & Economy

Gas costs:

Gas requirement of function MIGHTYZILLA.transferOwnership is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 404:8:

Gas costs:

Gas requirement of function MIGHTYZILLA.uniswapV2Pair is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 678:8:

Gas costs:

Gas requirement of function MIGHTYZILLA.name is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 885:8:

Gas costs:

Gas requirement of function MIGHTYZILLA.symbol is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 889:8:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point.
Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 1340:12:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point.
Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 957:8:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type

more

Pos: 453:8:

Miscellaneous

Constant/View/Pure functions:

SafeMath.sub(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 107:8:

Constant/View/Pure functions:

SafeMath.div(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 164:8:

Constant/View/Pure functions:

MIGHTYZILLA.reflectionFromToken(uint256,bool) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 976:8:

Similar variable names:

MIGHTYZILLA.(address payable,address payable) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.

Pos: 702:13:

Similar variable names:

MIGHTYZILLA.(address payable,address payable) : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.

Pos: 702:37:

Similar variable names:

MIGHTYZILLA.(address payable,address payable) : Variables have very similar names "_tTotal" and "_rTotal". Note: Modifiers are currently not considered by this static analysis.

Pos: 711:52:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 949:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 950:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 956:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 969:12:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1407:12:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 147:20:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 182:24:

Solhint Linter

MIGHTYZILLA.sol

```
MIGHTYZILLA.sol:2:1: Error: Compiler version ^0.6.12 does not satisfy
the r semver requirement
MIGHTYZILLA.sol:418:25: Error: Avoid to make time-based decisions in
your business logic
MIGHTYZILLA.sol:425:21: Error: Avoid to make time-based decisions in
your business logic
MIGHTYZILLA.sol:462:9: Error: Function name must be in mixedCase
MIGHTYZILLA.sol:463:9: Error: Function name must be in mixedCase
MIGHTYZILLA.sol:480:9: Error: Function name must be in mixedCase
MIGHTYZILLA.sol:500:9: Error: Function name must be in mixedCase
MIGHTYZILLA.sol:633:5: Error: Contract has 36 states declarations but
allowed no more than 15
MIGHTYZILLA.sol:654:32: Error: Use double quotes for string literals
MIGHTYZILLA.sol:655:34: Error: Use double quotes for string literals
MIGHTYZILLA.sol:662:25: Error: Variable name must be in mixedCase
MIGHTYZILLA.sol:674:33: Error: Variable name must be in mixedCase
MIGHTYZILLA.sol:675:33: Error: Variable name must be in mixedCase
MIGHTYZILLA.sol:680:9: Error: Explicitly mark visibility of state
MIGHTYZILLA.sol:699:22: Error: Variable name must be in mixedCase
MIGHTYZILLA.sol:699:65: Error: Variable name must be in mixedCase
MIGHTYZILLA.sol:863:22: Error: Avoid to make time-based decisions in
your business logic
MIGHTYZILLA.sol:948:9: Error: Function name must be in mixedCase
MIGHTYZILLA.sol:949:72: Error: Use double quotes for string literals
MIGHTYZILLA.sol:994:76: Error: Use double quotes for string literals
MIGHTYZILLA.sol:1091:17: Error: Avoid to make time-based decisions in
your business logic
MIGHTYZILLA.sol:1174:17: Error: Avoid to make time-based decisions in
your business logic
MIGHTYZILLA.sol:1286:36: Error: Code contains empty blocks
MIGHTYZILLA.sol:1306:63: Error: Variable name must be in mixedCase
MIGHTYZILLA.sol:1306:89: Error: Variable name must be in mixedCase
MIGHTYZILLA.sol:1362:50: Error: Use double quotes for string literals
MIGHTYZILLA.sol:1368:56: Error: Use double quotes for string literals
MIGHTYZILLA.sol:1373:58: Error: Use double quotes for string literals
MIGHTYZILLA.sol:1378:54: Error: Use double quotes for string literals
MIGHTYZILLA.sol:1383:60: Error: Use double quotes for string literals
MIGHTYZILLA.sol:1387:62: Error: Use double quotes for string literals
MIGHTYZILLA.sol:1391:39: Error: Variable name must be in mixedCase
MIGHTYZILLA.sol:1392:70: Error: Use double quotes for string literals
MIGHTYZILLA.sol:1397:36: Error: Variable name must be in mixedCase
MIGHTYZILLA.sol:1402:76: Error: Use double quotes for string literals
MIGHTYZILLA.sol:1407:78: Error: Use double quotes for string literals
MIGHTYZILLA.sol:1411:42: Error: Variable name must be in mixedCase
MIGHTYZILLA.sol:1415:39: Error: Variable name must be in mixedCase
```

Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

Email: audit@EtherAuthority.io