Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: EGONINU
Platform: Egon Blockchain
Language: Solidity
Date: September 5th, 2023

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 EGON Tax team to perform the Security audit of the ENU 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 September 5th, 2023.

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

- EGON tax token contract is utilized for swapping and liquifying, adding new liquidity, and updating fees.
- Smart contracts offer various functions, such as adding liquidity, swapping and liquefying, and setting fees.

Audit scope

Name	**Code Review and Security Analysis Report for EGON Tax (ENU) Token Smart Contract**
Platform	**Egon Blockchain / Solidity**
File	EgonInu.sol
Online code link	0x4BD0BaA28652F233817f3eC24Ff476fb36879E08
Audit Date	September 5th, 2023
Revised Audit Date	September 30th, 2023

Email: audit@EtherAuthority.io

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
ENU.sol • Name: Egon Inu • Symbol: ENU • Decimals: 9	**YES, This is valid.**
Total Fees: 10% • EGON Rewards Fee: 5% • Liquidity Fee: 1% • Marketing Fee: 3% • Burn Fee: 1% • Extra Fee On Sell: 0%	**YES, This is valid.** **The smart contract owner controls these functions, so the owner must handle the private key of the owner's wallet very securely.** **Because if the private key is compromised, then it will create problems.**
The Owner has control over the following functions: • Set EGON reward fee, liquidity fee, marketing fee and burn fees. • Set a swap and liquify enabled status. • Set the marketing address. • Set the maximum transaction limit. • Exclude from reward address. .	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **“Secured”**. Also, these contracts contain owner control, which does not make them 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 2 low and 1 very low level issues.

We confirm that all these issues are fixed / acknowledged in the revised smart contract code.

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	Passed
	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	Passed
	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	Passed
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	Passed
	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. Smart contract contains Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in ENU 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 ENU Token.

The ENU 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 in the smart contracts. Ethereum's NatSpec commenting style is recommended.

Documentation

We were given an ENU Token smart contract code in the form of a file. The hash of that code is mentioned above in the table.

As mentioned above, code parts are not **well** commented on. but the logic is straightforward. So it is 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.

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 not 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	setFee	write	access only Owner	No Issue
4	setExtraFeeOnSell	write	Removed	-
5	setMaxtx	write	Removed	-
6	receive	external	Passed	No Issue
7	updateRouterAddress	write	Removed	-
8	excludeFromFees	write	Removed	-
9	setExcludeFromMaxTx	write	Removed	-
10	setExcludeFromAll	write	Removed	-
11	excludeMultipleAccountsFromFees	write	Removed	-
12	setAutomatedMarketMakerPair	write	Removed	-
13	_setAutomatedMarketMakerPair	write	Removed	-
14	updateGasForProcessing	write	Removed	-
15	updateClaimWait	external	Removed	-
16	getClaimWait	external	Removed	-
17	getTotalDividendsDistributed	external	Removed	-
18	isExcludedFromFees	read	Removed	-
19	isExcludedFromMaxTx	read	Removed	-
20	withdrawableDividendOf	read	Removed	-
21	dividendTokenBalanceOf	read	Removed	-
22	getAccountDividendsInfo	external	Removed	-
23	getAccountDividendsInfoAtIndex	external	Removed	-
24	processDividendTracker	external	Removed	-
25	claim	external	Removed	-
26	getLastProcessedIndex	external	Removed	-
27	getNumberOfDividendTokenHolders	external	Removed	-
28	excludeFromDividends	external	Removed	-
29	setSwapAndLiquifyEnabled	write	access only Owner	No Issue
30	_transfer	internal	Transferred 0 amount	Refer Audit Findings
31	swapAndLiquify	write	lockTheSwap	No Issue
32	swapTokensForEth	write	Passed	No Issue
33	swapAndSendEGONToMarketing	write	Removed	-
34	addLiquidity	write	Passed	No Issue
35	name	read	Passed	No Issue
36	symbol	read	Passed	No Issue
37	decimals	read	Passed	No Issue
38	totalSupply	read	Passed	No Issue
39	balanceOf	read	Passed	No Issue

40	transfer	write	Passed	No Issue
41	allowance	read	Passed	No Issue
42	approve	write	Passed	No Issue
43	transferFrom	write	Passed	No Issue
44	increaseAllowance	write	Passed	No Issue
45	decreaseAllowance	write	Passed	No Issue
46	_transfer	internal	Removed	-
47	_mint	internal	Passed	No Issue
48	_burn	internal	Removed	-
49	_approve	internal	Passed	No Issue
50	_beforeTokenTransfer	internal	Removed	-
51	_afterTokenTransfer	internal	Removed	-
52	owner	read	Passed	No Issue
53	onlyOwner	modifier	Passed	No Issue
54	renounceOwnership	write	access only Owner	No Issue
55	transferOwnership	write	access only Owner	No Issue
56	_setOwner	write	Removed	-
57	setSafeManager	write	Removed	-
58	withdraw	external	Removed	-
59	withdrawEGON	external	Removed	-
60	openTrade	external	Removed	-
61	open	modifier	Removed	-
62	includeToWhiteList	external	Removed	-
63	_transfer	internal	Removed	-
64	withdrawDividend	write	Removed	-
65	excludeFromDividends	external	Removed	-
66	updateClaimWait	external	Removed	-
67	getLastProcessedIndex	external	Removed	-
68	getNumberOfTokenHolders	external	Removed	-
69	getAccount	write	Removed	-
70	getAccountAtIndex	write	Removed	-
71	canAutoClaim	read	Removed	-
72	setBalance	external	Removed	-
73	process	write	Removed	-
74	processAccount	write	Removed	-
75	receive	external	Removed	-
76	distributeDividends	write	Removed	-
77	withdrawDividend	write	Removed	-
78	_withdrawDividendOfUser	internal	Removed	-
79	dividendOf	read	Removed	-
80	withdrawableDividendOf	read	Removed	-
81	withdrawnDividendOf	read	Removed	-
82	accumulativeDividendOf	read	Removed	-
83	_transfer	internal	Passed	No Issue
84	_mint	internal	Removed	-
85	_burn	internal	Removed	-

86	_setBalance	internal	Removed	-
87	isExcludedFromFee	write	Passed	No Issue
88	excludeFromFee	write	Passed	No Issue
89	includeInFee	write	Passed	No Issue
90	_transferTokens	internal	Passed	No Issue
91	_setupDecimals	internal	Passed	No Issue
92	setMinimumTokensBeforeSwap	external	access only Owner	No Issue
93	setMarketingAddress	external	access only Owner	No Issue
94	getTreasuryAddress	read	Passed	No Issue
95	excludeFromReward	external	access only Owner	No Issue
96	setMaxTxnLimit	external	access only Owner	No Issue

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) Transfers 0 Amount:

```
function _transfer(address from, address to, uint256 amount) open(from, to) internal override
{
    require(from != address(0), "ERC20: transfer from the zero address");
    require(to != address(0), "ERC20: transfer to the zero address");

    if(amount == 0) {
        super._transfer(from, to, 0);
        return;
    }
```

Transfers 0 amounts.

Resolution: We suggest avoiding the 0 amount transfer in the _transfer() internal function.

Status: **Fixed,** Updated code in revised smart contract code.

(2) Owners can drain tokens and coins:

```
    function withdraw(address _token, uint256 _amount) external {
        require(msg.sender == safeManager);
        IERC20(_token).transfer(safeManager, _amount);
    }

    function withdrawBNB(uint256 _amount) external {
        require(msg.sender == safeManager);
        safeManager.transfer(_amount);
    }
```

Using the withdraw function, the owner can drain any tokens from the contract.
Using the withdrawEGON function, the owner can drain coins from the contract.

Resolution: We suggest confirming before contract deployment.
Status: **Fixed,** removed these functions in revised smart contract code.

Very Low / Informational / Best practices:

(1) Hardcoded address:

```
marketingWallet = payable(0xe743684437245F4bB5bc8311cF53Af387d2C4Cc6);
```

The marketingWallet is set by a hard coded address.

Resolution: We suggest confirming before contract deployment.
Status: **Acknowledged**

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:

EgonInu.sol

- excludeFromFee: Exclude fee account address set status true by the owner.
- includeInFee: Exclude fee account address set status false by the owner.
- setFee: The owner can set a EGON reward fee, liquidity fee, marketing fee, and burn fee.
- setMinimumTokensBeforeSwap: Minimum Tokens Before Swap value can be set by the owner.
- setSwapAndLiquifyEnabled: Swap and liquify enabled status can be set by the owner.
- setMarketingAddress: Marketing address can be set by the owner.
- excludeFromReward: Exclude from reward address can be set by the owner.
- setMaxTxnLimit: Maximum transaction limit can be set by the owner.

Ownable.sol

- renounce Ownership: Deleting ownership will leave the contract without an owner, removing any owner-only functionality.
- transfer ownership: The current owner can transfer ownership of the contract to a new account.

To make the smart contract 100% decentralized, we suggest renouncing ownership of the smart contract once its function is completed.

Conclusion

We were given a contract code in the form of a file. And we have used all possible tests based on given objects as files. We had observed 2 low issues and 1 informational issue in the smart contracts. We confirm that all these issues have been fixed / acknowledged in the revised smart contract code. So, **it's good to go for the 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 smart 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).

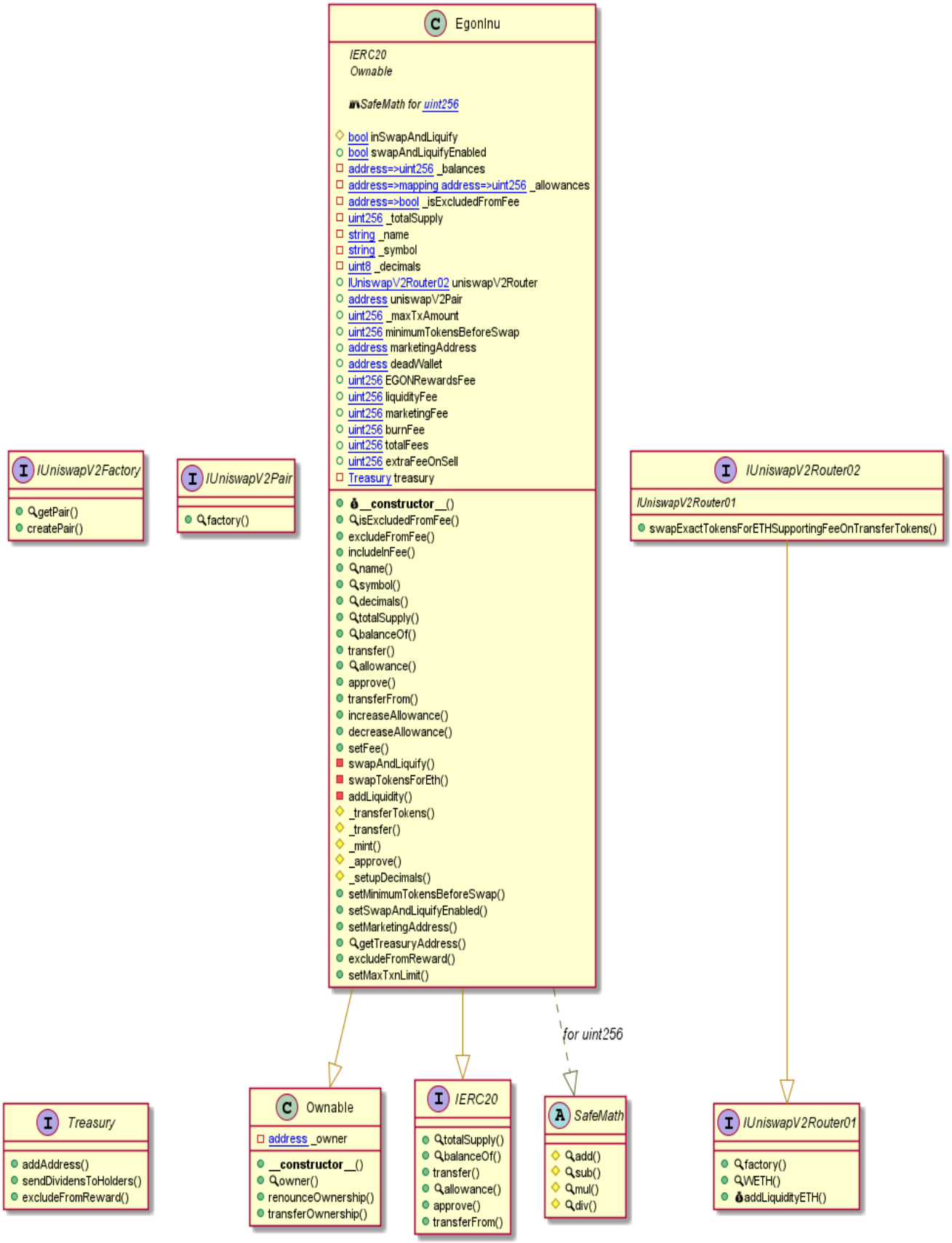
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we 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 - EGON Tax (ENU) Token



Slither Results Log

Slither is a Solidity static analysis framework that uses vulnerability detectors, displays contract details, and provides an API for writing custom analyses. It helps developers identify vulnerabilities, improve code comprehension, and prototype custom analyses quickly. The analysis includes a report with warnings and errors, allowing developers to quickly prototype and fix issues.

We did the analysis of the project altogether. Below are the results.

Slither log >> EgonInu.sol

```
EgonInu.constructor(address) (EgonInu.sol#163-191) performs a multiplication on the result of a division:
        - _maxTxAmount = totalSupply().div(100).mul(1) (EgonInu.sol#184)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

EgonInu.addLiquidity(uint256,uint256) (EgonInu.sol#312-322) ignores return value by uniswapV2Router.addLiquidityETH{value: eth
Amount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (EgonInu.sol#314-321)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

EgonInu.allowance(address,address).owner (EgonInu.sol#238) shadows:
        - Ownable.owner() (EgonInu.sol#15-17) (function)
EgonInu._approve(address,address,uint256).owner (EgonInu.sol#385) shadows:
        - Ownable.owner() (EgonInu.sol#15-17) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

EgonInu.setFee(uint256,uint256,uint256,uint256) (EgonInu.sol#268-276) should emit an event for:
        - liquidityFee = _liquidityFee (EgonInu.sol#271)
        - marketingFee = _marketingFee (EgonInu.sol#272)
        - burnFee = _burnFee (EgonInu.sol#273)
        - totalFees = EGONRewardsFee.add(liquidityFee).add(marketingFee).add(burnFee) (EgonInu.sol#274)
EgonInu.setMinimumTokensBeforeSwap(uint256) (EgonInu.sol#398-401) should emit an event for:
        - minimumTokensBeforeSwap = _minimumTokensBeforeSwap (EgonInu.sol#400)
EgonInu.setMaxTxnLimit(uint256) (EgonInu.sol#427-430) should emit an event for:
        - _maxTxAmount = maxTxAmount (EgonInu.sol#429)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Ownable.constructor().msgSender (EgonInu.sol#10) lacks a zero-check on :
                - _owner = msgSender (EgonInu.sol#11)
EgonInu.setMarketingAddress(address)._marketingAddress (EgonInu.sol#409) lacks a zero-check on :
                - marketingAddress = address(_marketingAddress) (EgonInu.sol#411)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Reentrancy in EgonInu._transferTokens(address,address,uint256) (EgonInu.sol#325-364):
        External calls:
        - treasury.addAddress(to) (EgonInu.sol#333)
        - swapAndLiquify(contractTokenBalance) (EgonInu.sol#343)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Eg
onInu.sol#314-321)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.ti
mestamp) (EgonInu.sol#299-305)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (EgonInu.sol#343)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Eg
onInu.sol#314-321)
                - marketingAddress.transfer(marketingShare) (EgonInu.sol#287)
                - address(address(treasury)).transfer(newBalance.sub(marketingShare).sub(liquidityShare)) (EgonInu.sol#290)
        State variables written after the call(s):
        - swapAndLiquify(contractTokenBalance) (EgonInu.sol#343)
                - _allowances[owner][spender] = amount (EgonInu.sol#389)
Reentrancy in EgonInu.swapAndLiquify(uint256) (EgonInu.sol#279-291):
        External calls:
        - swapTokensForEth(swapableTokens) (EgonInu.sol#284)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.ti
mestamp) (EgonInu.sol#299-305)
        - addLiquidity(halfLiquidityTokens,liquidityShare) (EgonInu.sol#289)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Eg
onInu.sol#314-321)
        External calls sending eth:
        - marketingAddress.transfer(marketingShare) (EgonInu.sol#287)
        - addLiquidity(halfLiquidityTokens,liquidityShare) (EgonInu.sol#289)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Eg
onInu.sol#314-321)
```

```
        State variables written after the call(s):
        - addLiquidity(halfLiquidityTokens,liquidityShare) (EgonInu.sol#289)
                - _allowances[owner][spender] = amount (EgonInu.sol#389)
Reentrancy in EgonInu.transferFrom(address,address,uint256) (EgonInu.sol#249-253):
        External calls:
        - _transferTokens(sender,recipient,amount) (EgonInu.sol#250)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Eg
onInu.sol#314-321)
                - treasury.addAddress(to) (EgonInu.sol#333)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.ti
mestamp) (EgonInu.sol#299-305)
```

```
                - marketingAddress.transfer(marketingShare) (EgonInu.sol#287)
                - address(address(treasury)).transfer(newBalance.sub(marketingShare).sub(liquidityShare)) (EgonInu.sol#290)
        State variables written after the call(s):
        - _approve(sender,msg.sender,_allowances[sender][msg.sender].sub(amount,ERC20: transfer amount exceeds allowance)) (EgonInu.sol#251)
                - _allowances[owner][spender] = amount (EgonInu.sol#389)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in EgonInu._transferTokens(address,address,uint256) (EgonInu.sol#325-364):
        External calls:
        - treasury.addAddress(to) (EgonInu.sol#333)
        - swapAndLiquify(contractTokenBalance) (EgonInu.sol#343)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (EgonInu.sol#314-321)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (EgonInu.sol#299-305)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (EgonInu.sol#343)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (EgonInu.sol#314-321)
                - marketingAddress.transfer(marketingShare) (EgonInu.sol#287)
                - address(address(treasury)).transfer(newBalance.sub(marketingShare).sub(liquidityShare)) (EgonInu.sol#290)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (EgonInu.sol#390)
                - swapAndLiquify(contractTokenBalance) (EgonInu.sol#343)
        - SwapTokensForETH(tokenAmount,path) (EgonInu.sol#307)
                - swapAndLiquify(contractTokenBalance) (EgonInu.sol#343)
Reentrancy in EgonInu._transferTokens(address,address,uint256) (EgonInu.sol#325-364):
        External calls:
        - treasury.addAddress(to) (EgonInu.sol#333)
        - swapAndLiquify(contractTokenBalance) (EgonInu.sol#343)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (EgonInu.sol#314-321)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (EgonInu.sol#299-305)
        - treasury.sendDividensToHolders() (EgonInu.sol#349)
        External calls sending eth:
```

```
        - swapAndLiquify(contractTokenBalance) (EgonInu.sol#343)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (EgonInu.sol#314-321)
                - marketingAddress.transfer(marketingShare) (EgonInu.sol#287)
                - address(address(treasury)).transfer(newBalance.sub(marketingShare).sub(liquidityShare)) (EgonInu.sol#290)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (EgonInu.sol#372)
                - _transfer(from,address(this),_feeTokens) (EgonInu.sol#355)
        - Transfer(sender,recipient,amount) (EgonInu.sol#372)
                - _transfer(from,deadWallet,_burnTokens) (EgonInu.sol#358)
        - Transfer(sender,recipient,amount) (EgonInu.sol#372)
                - _transfer(from,to,amount) (EgonInu.sol#363)
Reentrancy in EgonInu.swapAndLiquify(uint256) (EgonInu.sol#279-291):
        External calls:
        - swapTokensForEth(swapableTokens) (EgonInu.sol#284)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (EgonInu.sol#299-305)
        - addLiquidity(halfLiquidityTokens,liquidityShare) (EgonInu.sol#289)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (EgonInu.sol#314-321)
        External calls sending eth:
        - marketingAddress.transfer(marketingShare) (EgonInu.sol#287)
        - addLiquidity(halfLiquidityTokens,liquidityShare) (EgonInu.sol#289)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (EgonInu.sol#314-321)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (EgonInu.sol#390)
                - addLiquidity(halfLiquidityTokens,liquidityShare) (EgonInu.sol#289)
Reentrancy in EgonInu.swapTokensForEth(uint256) (EgonInu.sol#294-308):
        External calls:
        - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (EgonInu.sol#299-305)
        Event emitted after the call(s):
        - SwapTokensForETH(tokenAmount,path) (EgonInu.sol#307)
Reentrancy in EgonInu.transferFrom(address,address,uint256) (EgonInu.sol#249-253):
        External calls:
        - _transferTokens(sender,recipient,amount) (EgonInu.sol#250)
```

```
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (EgonInu.sol#314-321)
                - treasury.addAddress(to) (EgonInu.sol#333)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (EgonInu.sol#299-305)
                - treasury.sendDividensToHolders() (EgonInu.sol#349)
        External calls sending eth:
        - _transferTokens(sender,recipient,amount) (EgonInu.sol#250)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (EgonInu.sol#314-321)
                - marketingAddress.transfer(marketingShare) (EgonInu.sol#287)
                - address(address(treasury)).transfer(newBalance.sub(marketingShare).sub(liquidityShare)) (EgonInu.sol#290)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (EgonInu.sol#390)
                - _approve(sender,msg.sender,_allowances[sender][msg.sender].sub(amount,ERC20: transfer amount exceeds allowance)) (EgonInu.sol#251)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

EgonInu._setupDecimals(uint8) (EgonInu.sol#394-396) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.18 (EgonInu.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.18 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Function IUniswapV2Router01.WETH() (EgonInu.sol#100) is not in mixedCase
Parameter EgonInu.setFee(uint256,uint256,uint256,uint256)._egonRewardFee (EgonInu.sol#268) is not in mixedCase
Parameter EgonInu.setFee(uint256,uint256,uint256,uint256)._liquidityFee (EgonInu.sol#268) is not in mixedCase
Parameter EgonInu.setFee(uint256,uint256,uint256,uint256)._marketingFee (EgonInu.sol#268) is not in mixedCase
Parameter EgonInu.setFee(uint256,uint256,uint256,uint256)._burnFee (EgonInu.sol#268) is not in mixedCase
Parameter EgonInu.setMinimumTokensBeforeSwap(uint256)._minimumTokensBeforeSwap (EgonInu.sol#398) is not in mixedCase
Parameter EgonInu.setSwapAndLiquifyEnabled(bool)._enabled (EgonInu.sol#403) is not in mixedCase
Parameter EgonInu.setMarketingAddress(address)._marketingAddress (EgonInu.sol#409) is not in mixedCase
Parameter EgonInu.excludeFromReward(address,bool)._enable (EgonInu.sol#422) is not in mixedCase
Variable EgonInu._maxTxAmount (EgonInu.sol#151) is not in mixedCase
Variable EgonInu.EGONRewardsFee (EgonInu.sol#155) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Reentrancy in EgonInu._transferTokens(address,address,uint256) (EgonInu.sol#325-364):
        External calls:
        - swapAndLiquify(contractTokenBalance) (EgonInu.sol#343)
                - marketingAddress.transfer(marketingShare) (EgonInu.sol#287)
                - address(address(treasury)).transfer(newBalance.sub(marketingShare).sub(liquidityShare)) (EgonInu.sol#290)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (EgonInu.sol#343)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Eg
onInu.sol#314-321)
                - marketingAddress.transfer(marketingShare) (EgonInu.sol#287)
                - address(address(treasury)).transfer(newBalance.sub(marketingShare).sub(liquidityShare)) (EgonInu.sol#290)
        State variables written after the call(s):
        - _transfer(from,address(this),_feeTokens) (EgonInu.sol#355)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (EgonInu.sol#370)
                - _balances[recipient] = _balances[recipient].add(amount) (EgonInu.sol#371)
        - _transfer(from,deadWallet,_burnTokens) (EgonInu.sol#358)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (EgonInu.sol#370)
                - _balances[recipient] = _balances[recipient].add(amount) (EgonInu.sol#371)
        - _transfer(from,to,amount) (EgonInu.sol#363)
                - _balances[sender] = _balances[sender].sub(amount,ERC20: transfer amount exceeds balance) (EgonInu.sol#370)
                - _balances[recipient] = _balances[recipient].add(amount) (EgonInu.sol#371)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (EgonInu.sol#372)

Reentrancy in EgonInu.swapAndLiquify(uint256) (EgonInu.sol#279-291):
        External calls:
        - marketingAddress.transfer(marketingShare) (EgonInu.sol#287)
        External calls sending eth:
        - marketingAddress.transfer(marketingShare) (EgonInu.sol#287)
        - addLiquidity(halfLiquidityTokens,liquidityShare) (EgonInu.sol#289)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Eg
onInu.sol#314-321)
        State variables written after the call(s):
        - addLiquidity(halfLiquidityTokens,liquidityShare) (EgonInu.sol#289)
                - _allowances[owner][spender] = amount (EgonInu.sol#389)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (EgonInu.sol#390)
                - addLiquidity(halfLiquidityTokens,liquidityShare) (EgonInu.sol#289)

                - marketingAddress.transfer(marketingShare) (EgonInu.sol#287)
                - address(address(treasury)).transfer(newBalance.sub(marketingShare).sub(liquidityShare)) (EgonInu.sol#290)
        External calls sending eth:
        - _transferTokens(sender,recipient,amount) (EgonInu.sol#250)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Eg
onInu.sol#314-321)
                - marketingAddress.transfer(marketingShare) (EgonInu.sol#287)
                - address(address(treasury)).transfer(newBalance.sub(marketingShare).sub(liquidityShare)) (EgonInu.sol#290)
        State variables written after the call(s):
        - _approve(sender,msg.sender,_allowances[sender][msg.sender].sub(amount,ERC20: transfer amount exceeds allowance)) (Eg
onInu.sol#251)
                - _allowances[owner][spender] = amount (EgonInu.sol#389)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (EgonInu.sol#390)
                - _approve(sender,msg.sender,_allowances[sender][msg.sender].sub(amount,ERC20: transfer amount exceeds allowan
ce)) (EgonInu.sol#251)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Variable EgonInu.EGONRewardsFee (EgonInu.sol#155) is too similar to EgonInu.setFee(uint256,uint256,uint256,uint256)._egonRewar
dFee (EgonInu.sol#268)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

EgonInu.constructor(address) (EgonInu.sol#163-191) uses literals with too many digits:
        - _mint(owner(),542000000000 * (10 ** 9)) (EgonInu.sol#177)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

EgonInu.deadWallet (EgonInu.sol#154) should be constant
EgonInu.extraFeeOnSell (EgonInu.sol#160) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

EgonInu._name (EgonInu.sol#145) should be immutable
EgonInu._symbol (EgonInu.sol#146) should be immutable
EgonInu.treasury (EgonInu.sol#161) should be immutable
EgonInu.uniswapV2Pair (EgonInu.sol#150) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
EgonInu.sol analyzed (9 contracts with 84 detectors), 44 result(s) found
```

Solidity Static Analysis

EgonInu.sol

Check-effects-interaction:

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

more

Pos: 164:4:

Check-effects-interaction:

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

more

Pos: 328:4:

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: 307:12:

Gas costs:

Gas requirement of function EgonInu.setFee 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: 269:4:

Email: audit@EtherAuthority.io

Gas costs:

Gas requirement of function EgonInu.setFee 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: 269:4:

Constant/View/Pure functions:

Treasury.addAddress(address) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 119:3:

Similar variable names:

EgonInu.swapAndLiquify(uint256) : Variables have very similar names "totalFee" and "totalFees". Note: Modifiers are currently not considered by this static analysis.
Pos: 283:81:

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: 390:8:

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: 67:16:

Solhint Linter

EgonInu.sol

```
Compiler version 0.8.18 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:8
Error message for require is too long
Pos: 9:29
Error message for require is too long
Pos: 9:66
Function name must be in mixedCase
Pos: 5:99
Contract has 21 states declarations but allowed no more than 15
Pos: 1:124
Explicitly mark visibility of state
Pos: 7:131
Variable name must be in mixedCase
Pos: 5:154
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:163
Avoid making time-based decisions in your business logic
Pos: 13:306
Avoid making time-based decisions in your business logic
Pos: 13:322
Error message for require is too long
Pos: 9:370
Error message for require is too long
Pos: 9:371
Error message for require is too long
Pos: 9:388
Error message for require is too long
Pos: 9:389
Code contains empty blocks
Pos: 32:416
```

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