Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: 6ense Token
Website: https://www.6ense.it
Platform: Ethereum / BSC
Language: Solidity
Date: September 15th, 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 6OS team to perform the Security audit of the 6ense 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 15th, 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

- The 6ense token is a token smart contract that has functions like delivery, updating fees, receiving, swapping, liquidity, and adding liquidity.

Audit scope

Name	Code Review and Security Analysis Report for 6ense Token Smart Contract
Platform	**Ethereum / BSC**
File	6ense.sol
Github commit hash	be874bcd26a997db836b8002573c4436758d3931
Ethereum Code	0x85018d46c4F21460490d841ef43b07bbcc99F6Dc
BSC Code	0x223be542ffb661714d209e5ed7e4af18e83bc80c
Audit Date	September 15th, 2023

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: • Name: 6ense • Symbol: 6OS • Decimals: 18	**YES, This is valid.**
• Reward Fees: 2% • Transaction tax: 4% • Liquidity tax: 5% • Burn tax: 1% • Maximum transaction amount: 5 Million • Number of Blocks For Blacklist: 5	**YES, This is valid.**
Ownership Control: • Current owner can transfer the ownership. • Owner can renounce ownership. • Enable trade. • Update an account address on the block list. • Update reward fees. • Update the transaction. • Updated liquidity fee percentage. • Set the maximum transaction amount. • Set the swap and liquify enabled status.	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity based 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 1 low and 3 very low level issues.
We confirmed that all the issues have been acknowledged.

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	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	Moderated
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 6ense 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 6ense Token.

The 6ense 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 a 6ense Token smart contract code in the form of a github web link. The hash of that code is mentioned above in the table.

As mentioned above, code parts are not well commented. 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.

Another source of information was its official project URL: https://www.6ense.it 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 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	totalSupply	read	Passed	No Issue
4	balanceOf	read	Passed	No Issue
5	transfer	read	Passed	No Issue
6	allowance	read	Passed	No Issue
7	approve	write	Passed	No Issue
8	transferFrom	write	Passed	No Issue
9	increaseAllowance	write	Passed	No Issue
10	decreaseAllowance	write	Passed	No Issue
11	isExcludedFromReward	read	Passed	No Issue
12	totalFees	read	Passed	No Issue
13	deliver	write	Passed	No Issue
14	reflectionFromToken	read	Passed	No Issue
15	tokenFromReflection	read	Passed	No Issue
16	enableTrade	write	access only Owner	No Issue
17	isTradeEnabled	external	Passed	No Issue
18	addToBlacklist	write	access only Owner	No Issue
19	removeFromBlacklist	write	access only Owner	No Issue
20	excludeFromReward	write	access only Owner	No Issue
21	includeInReward	external	access only Owner	No Issue
22	_transferBothExcluded	write	Passed	No Issue
23	excludeFromFee	write	access only Owner	No Issue
24	includeInFee	write	access only Owner	No Issue
25	setRewardFeePercent	external	access only Owner	No Issue
26	setTransactionFeePercent	external	access only Owner	No Issue
27	setLiquidityFeePercent	external	access only Owner	No Issue
28	setNumTokensSellToAddToLiquidity	external	access only Owner	No Issue
29	setMaxTxAmount	external	access only Owner	No Issue
30	setMaxTxPercent	external	access only Owner	No Issue
31	setSwapAndLiquifyEnabled	write	access only Owner	No Issue
32	receive	external	Passed	No Issue
33	_reflectFee	write	Passed	No Issue
34	_getValues	read	Passed	No Issue
35	_getTValues	read	Passed	No Issue
36	_getRValues	write	Passed	No Issue
37	_getRate	read	Passed	No Issue
38	_getCurrentSupply	read	Passed	No Issue
39	_takeLiquidity	write	Passed	No Issue

40	_sendTransactionTax	write	Passed	No Issue
41	calculateFee	write	Passed	No Issue
42	removeAllFee	write	Passed	No Issue
43	restoreAllFee	write	Passed	No Issue
44	isExcludedFromFee	read	Passed	No Issue
45	_approve	internal	Passed	No Issue
46	_transfer	internal	Passed	No Issue
47	_burnTokens	write	Critical operation lacks event log, Tokens not burned while transferring from reward excluded address	Refer Audit Findings
48	swapAndLiquify	write	lockTheSwap	No Issue
49	swapTokensForEth	write	Passed	No Issue
50	addLiquidity	write	Passed	No Issue
51	_tokenTransfer	write	Unnecessary condition used	Refer Audit Findings
52	_transferStandard	write	Passed	No Issue
53	_transferToExcluded	write	Passed	No Issue
54	_transferFromExcluded	write	Passed	No Issue
55	name	read	Passed	No Issue
56	symbol	read	Passed	No Issue
57	decimals	read	Passed	No Issue
58	totalSupply	read	Passed	No Issue
59	balanceOf	read	Passed	No Issue
60	transfer	write	Passed	No Issue
61	allowance	read	Passed	No Issue
62	approve	write	Passed	No Issue
63	transferFrom	write	Passed	No Issue
64	increaseAllowance	write	Passed	No Issue
65	decreaseAllowance	write	Passed	No Issue
66	_transfer	internal	Passed	No Issue
67	_mint	internal	Passed	No Issue
68	_burn	internal	Passed	No Issue
69	_approve	internal	Passed	No Issue
70	_spendAllowance	internal	Passed	No Issue
71	_beforeTokenTransfer	internal	Passed	No Issue
72	_afterTokenTransfer	internal	Passed	No Issue
73	onlyOwner	modifier	Passed	No Issue
74	owner	read	Passed	No Issue
75	_checkOwner	internal	Passed	No Issue
76	renounceOwnership	write	access only Owner	No Issue
77	transferOwnership	write	access only Owner	No Issue
78	_transferOwnership	internal	Passed	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) Tokens not burned while transferring from reward excluded address:

```
function _burnTokens(
    address from,
    address to,
    uint256 tAmount,
    uint256 rate
) private {
    if (_isExcluded[from]) {
        tOwned[from] -= tAmount;
        _tOwned[to] += tAmount * rate;
    } else {
        _rOwned[from] -= tAmount;
        _rOwned[to] += tAmount * rate;
    }
}
```

1% of tokens do not get burned while transferring tokens from reward excluded addresses.

Resolution: We suggest correcting the logic by adding the proper amount for the burn address.

Status: Acknowledged by team as this does not have a major security impact.

Very Low / Informational / Best practices:

(1) Critical operation lacks event log:

Missing event log for:

- _burnTokens()

Resolution: Please write an event log for the listed events.

Status: Acknowledged by team as this does not have a major security impact.

(2) Unused event:

```
event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
```

The MinTokensBeforeSwapUpdated event is not used in the contract.

Resolution: We suggest removing unused events.

Status: Acknowledged by team as this does not have a major security impact.

(3) Unnecessary condition used:

```
//this method is responsible for taking all fee, if takeFee is true
function _tokenTransfer(address sender, address recipient, uint256 amount,bool takeFee) private {    infinite gas
    if(!takeFee)
        removeAllFee();

      if (_isExcluded[sender] && !_isExcluded[recipient]) {
        _transferFromExcluded(sender, recipient, amount);
    } else if (!_isExcluded[sender] && _isExcluded[recipient]) {
        _transferToExcluded(sender, recipient, amount);
    } else if (!_isExcluded[sender] && !_isExcluded[recipient]) {
        _transferStandard(sender, recipient, amount);
    } else if (_isExcluded[sender] && _isExcluded[recipient]) {
        _transferBothExcluded(sender, recipient, amount);
    } else {
        _transferStandard(sender, recipient, amount);
    }

    if(!takeFee)
        restoreAllFee();
}
```

Unnecessary "**else if"** condition used in _tokenTransfer function.

Resolution: We suggest removing unnecessary condition in the smart contract.

Status: Acknowledged by team as this does not have a major security impact.

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:

6ense.sol

- enableTrade: Trade status can be set by the owner.
- addToBlacklist: An account address can be added to the blacklist by the owner.
- removeFromBlacklist: The account address can be removed from the blacklist by the owner.
- excludeFromReward: The account address can be excluded from rewards by the owner.
- includeInReward: The account address can be included in the reward by the owner.
- excludeFromFee: Account status can be set to be true and be excluded from the fee by the owner.
- includeInFee: Account status can be set to false to includeInFee by the owner.
- setRewardFeePercent: Reward fee percentage values can be set by the owner.
- setTransactionFeePercent: Transaction fee percentage values can be set by the owner.
- setLiquidityFeePercent: Liquidity fee percentage values can be set by the owner.
- setNumTokensSellToAddToLiquidity: The liquidity token amount can be set by the owner.
- setMaxTxAmount: The maximum transaction amount can be set by the owner.
- setMaxTxPercent: The maximum transaction percentage can be set by the owner.
- setSwapAndLiquifyEnabled: Swap and Liquify enabled status 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 github web link. And we have used all possible tests based on given objects as files. We had observed 1 low and 3 informational issues in the smart contracts. We confirmed that all the issues have been acknowledged. 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".**

Email: audit@EtherAuthority.io

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).

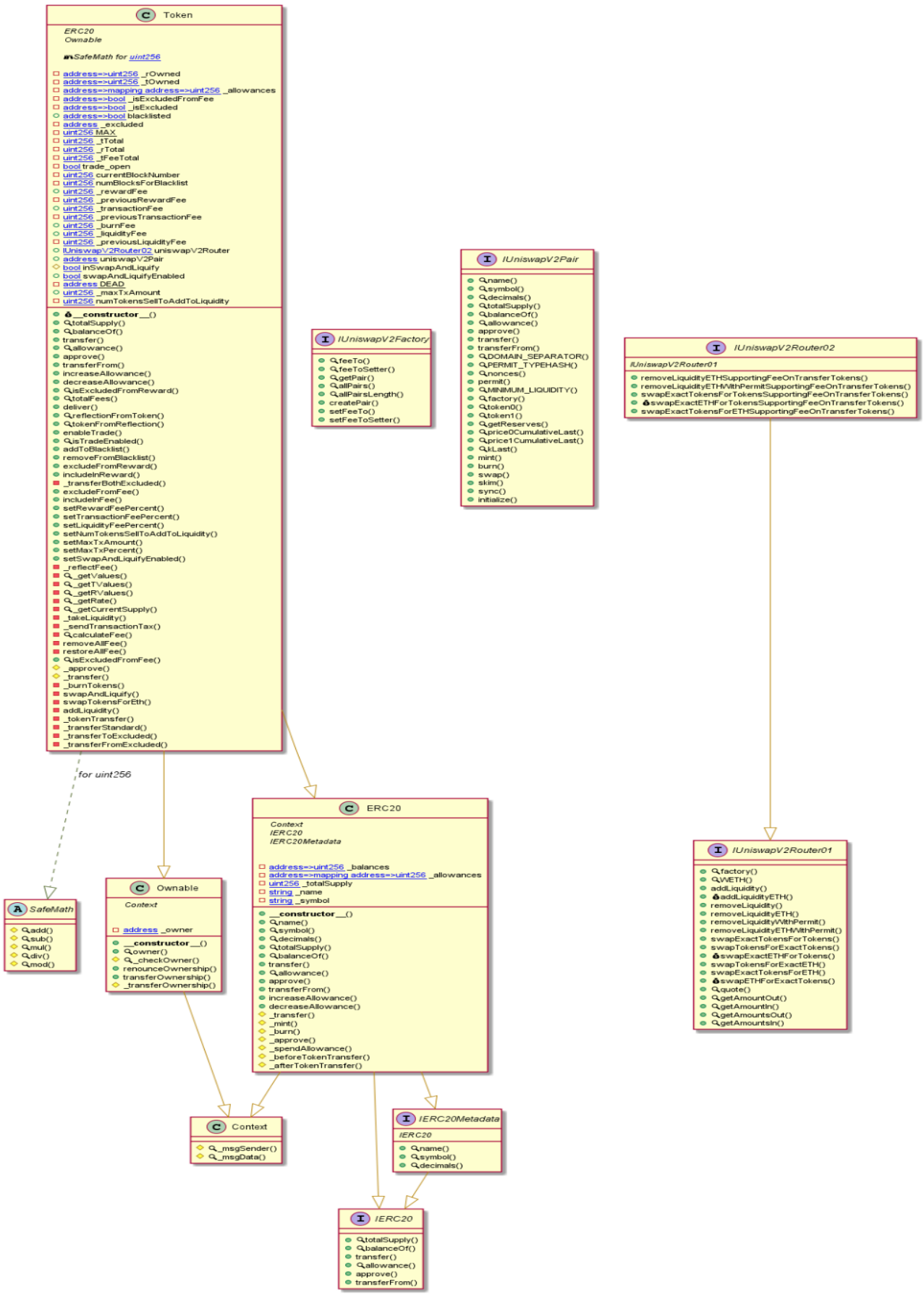
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we 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 - 6ense 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 >> 6ense.sol

```
Token.allowance(address,address).owner (6ense.sol#1009) shadows:
        - Ownable.owner() (6ense.sol#254-256) (function)
Token._approve(address,address,uint256).owner (6ense.sol#1260) shadows:
        - Ownable.owner() (6ense.sol#254-256) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Token.setRewardFeePercent(uint256) (6ense.sol#1132-1135) should emit an event for:
        - _rewardFee = rewardFee (6ense.sol#1134)
Token.setTransactionFeePercent(uint256) (6ense.sol#1137-1140) should emit an event for:
        - _transactionFee = transactionFee (6ense.sol#1139)
Token.setLiquidityFeePercent(uint256) (6ense.sol#1142-1145) should emit an event for:
        - _liquidityFee = liquidityFee (6ense.sol#1144)
Token.setNumTokensSellToAddToLiquidity(uint256) (6ense.sol#1147-1150) should emit an event for:
        - numTokensSellToAddToLiquidity = amount (6ense.sol#1149)
Token.setMaxTxAmount(uint256) (6ense.sol#1152-1155) should emit an event for:
        - _maxTxAmount = amount (6ense.sol#1154)
Token.setMaxTxPercent(uint256) (6ense.sol#1157-1161) should emit an event for:
        - _maxTxAmount = _tTotal.mul(maxTxPercent).div(10 ** 2) (6ense.sol#1158-1160)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Reentrancy in Token._transfer(address,address,uint256) (6ense.sol#1268-1333):
        External calls:
        - swapAndLiquify(contractTokenBalance) (6ense.sol#1314)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (6e
nse.sol#1396-1403)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.ti
mestamp) (6ense.sol#1382-1388)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (6ense.sol#1314)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (6e
nse.sol#1396-1403)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount,takeFee) (6ense.sol#1332)
                - _liquidityFee = _previousLiquidityFee (6ense.sol#1252)
                - _liquidityFee = 0 (6ense.sol#1246)
        - _tokenTransfer(from,to,amount,takeFee) (6ense.sol#1332)
                - _previousLiquidityFee = _liquidityFee (6ense.sol#1242)
        - _tokenTransfer(from,to,amount,takeFee) (6ense.sol#1332)
                - _previousRewardFee = _rewardFee (6ense.sol#1241)
        - _tokenTransfer(from,to,amount,takeFee) (6ense.sol#1332)
                - _previousTransactionFee = _transactionFee (6ense.sol#1243)
        - _tokenTransfer(from,to,amount,takeFee) (6ense.sol#1332)
                - _rewardFee = _previousRewardFee (6ense.sol#1251)
                - _rewardFee = 0 (6ense.sol#1245)
        - _tokenTransfer(from,to,amount,takeFee) (6ense.sol#1332)
                - _tFeeTotal = _tFeeTotal.add(tFee) (6ense.sol#1174)
        - _tokenTransfer(from,to,amount,takeFee) (6ense.sol#1332)
                - _transactionFee = _previousTransactionFee (6ense.sol#1253)
                - _transactionFee = 0 (6ense.sol#1247)
Reentrancy in Token.swapAndLiquify(uint256) (6ense.sol#1350-1371):
        External calls:
        - swapTokensForEth(half) (6ense.sol#1362)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.ti
mestamp) (6ense.sol#1382-1388)
        - addLiquidity(otherHalf,newBalance) (6ense.sol#1368)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (6e
nse.sol#1396-1403)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (6ense.sol#1368)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (6e
nse.sol#1396-1403)
        State variables written after the call(s):
        - addLiquidity(otherHalf,newBalance) (6ense.sol#1368)
                - _allowances[owner][spender] = amount (6ense.sol#1264)
```

```
Reentrancy in Token.transferFrom(address,address,uint256) (6ense.sol#1018-1022):
        External calls:
        - _transfer(sender,recipient,amount) (6ense.sol#1019)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (6e
nse.sol#1396-1403)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.ti
mestamp) (6ense.sol#1382-1388)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (6ense.sol#1019)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (6e
nse.sol#1396-1403)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance))
 (6ense.sol#1020)
                - _allowances[owner][spender] = amount (6ense.sol#1264)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Token._transfer(address,address,uint256) (6ense.sol#1268-1333):
        External calls:
        - swapAndLiquify(contractTokenBalance) (6ense.sol#1314)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (6e
nse.sol#1396-1403)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.ti
mestamp) (6ense.sol#1382-1388)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (6ense.sol#1314)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (6e
nse.sol#1396-1403)
        Event emitted after the call(s):
        - Transfer(sender,recipient,tTransferAmount) (6ense.sol#1434)
                - _tokenTransfer(from,to,amount,takeFee) (6ense.sol#1332)
        - Transfer(sender,recipient,tTransferAmount) (6ense.sol#1456)
                - _tokenTransfer(from,to,amount,takeFee) (6ense.sol#1332)
        - Transfer(sender,recipient,tTransferAmount) (6ense.sol#1445)
                - _tokenTransfer(from,to,amount,takeFee) (6ense.sol#1332)
        - Transfer(sender,recipient,tTransferAmount) (6ense.sol#1120)
```

```
                - _tokenTransfer(from,to,amount,takeFee) (6ense.sol#1332)
Reentrancy in Token.swapAndLiquify(uint256) (6ense.sol#1350-1371):
        External calls:
        - swapTokensForEth(half) (6ense.sol#1362)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.ti
mestamp) (6ense.sol#1382-1388)
        - addLiquidity(otherHalf,newBalance) (6ense.sol#1368)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (6e
nse.sol#1396-1403)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (6ense.sol#1368)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (6e
nse.sol#1396-1403)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (6ense.sol#1265)
                - addLiquidity(otherHalf,newBalance) (6ense.sol#1368)
        - SwapAndLiquify(half,newBalance,otherHalf) (6ense.sol#1370)
Reentrancy in Token.transferFrom(address,address,uint256) (6ense.sol#1018-1022):
        External calls:
        - _transfer(sender,recipient,amount) (6ense.sol#1019)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (6e
nse.sol#1396-1403)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.ti
mestamp) (6ense.sol#1382-1388)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (6ense.sol#1019)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (6e
nse.sol#1396-1403)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (6ense.sol#1265)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds all
owance)) (6ense.sol#1020)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
Token.includeInReward(address) (6ense.sol#1098-1109) has costly operations inside a loop:
        - _excluded.pop() (6ense.sol#1105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

Context._msgData() (6ense.sol#222-224) is never used and should be removed
ERC20._afterTokenTransfer(address,address,uint256) (6ense.sol#694-698) is never used and should be removed
ERC20._approve(address,address,uint256) (6ense.sol#623-633) is never used and should be removed
ERC20._beforeTokenTransfer(address,address,uint256) (6ense.sol#674-678) is never used and should be removed
ERC20._burn(address,uint256) (6ense.sol#592-608) is never used and should be removed
ERC20._mint(address,uint256) (6ense.sol#566-579) is never used and should be removed
ERC20._spendAllowance(address,address,uint256) (6ense.sol#643-658) is never used and should be removed
ERC20._transfer(address,address,uint256) (6ense.sol#530-555) is never used and should be removed
SafeMath.mod(uint256,uint256) (6ense.sol#195-197) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (6ense.sol#211-214) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Token._rTotal (6ense.sol#933) is set pre-construction with a non-constant function or state variable:
        - (MAX - (MAX % _tTotal))
Token._previousRewardFee (6ense.sol#942) is set pre-construction with a non-constant function or state variable:
        - _rewardFee
Token._previousTransactionFee (6ense.sol#945) is set pre-construction with a non-constant function or state variable:
        - _transactionFee
Token._previousLiquidityFee (6ense.sol#950) is set pre-construction with a non-constant function or state variable:
        - _liquidityFee
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state

Pragma version0.8.19 (6ense.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.1
6
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (6ense.sol#738) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (6ense.sol#739) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (6ense.sol#756) is not in mixedCase
Function IUniswapV2Router01.WETH() (6ense.sol#778) is not in mixedCase
Parameter Token.enableTrade(bool)._enable (6ense.sol#1068) is not in mixedCase
Parameter Token.setSwapAndLiquifyEnabled(bool)._enabled (6ense.sol#1163) is not in mixedCase
Parameter Token.calculateFee(uint256,uint256)._amount (6ense.sol#1234) is not in mixedCase
Variable Token.trade_open (6ense.sol#936) is not in mixedCase
Variable Token._rewardFee (6ense.sol#941) is not in mixedCase
Variable Token._transactionFee (6ense.sol#944) is not in mixedCase
Variable Token._burnFee (6ense.sol#947) is not in mixedCase
Variable Token._liquidityFee (6ense.sol#949) is not in mixedCase
Variable Token._maxTxAmount (6ense.sol#960) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (6ens
e.sol#783) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).
amountBDesired (6ense.sol#784)
Variable Token._getRValues(uint256,uint256,uint256,uint256,uint256).rTransaction (6ense.sol#1195) is too similar to Token._tra
nsferToExcluded(address,address,uint256).tTransaction (6ense.sol#1438)
Variable Token._sendTransactionTax(uint256).rTransaction (6ense.sol#1227) is too similar to Token._getRValues(uint256,uint256,
uint256,uint256,uint256).tTransaction (6ense.sol#1191)
Variable Token._sendTransactionTax(uint256).rTransaction (6ense.sol#1227) is too similar to Token._getValues(uint256).tTransac
tion (6ense.sol#1178)
Variable Token._sendTransactionTax(uint256).rTransaction (6ense.sol#1227) is too similar to Token._transferStandard(address,ad
dress,uint256).tTransaction (6ense.sol#1428)
Variable Token._sendTransactionTax(uint256).rTransaction (6ense.sol#1227) is too similar to Token._transferToExcluded(address,
address,uint256).tTransaction (6ense.sol#1438)
Variable Token.reflectionFromToken(uint256,bool).rTransferAmount (6ense.sol#1057) is too similar to Token._transferStandard(ad
dress,address,uint256).tTransferAmount (6ense.sol#1428)
Variable Token.reflectionFromToken(uint256,bool).rTransferAmount (6ense.sol#1057) is too similar to Token._transferToExcluded(
address,address,uint256).tTransferAmount (6ense.sol#1438)
Variable Token._transferToExcluded(address,address,uint256).rTransferAmount (6ense.sol#1438) is too similar to Token._transfer
ToExcluded(address,address,uint256).tTransferAmount (6ense.sol#1438)
Variable Token._getValues(uint256).rTransferAmount (6ense.sol#1179) is too similar to Token._transferToExcluded(address,addres
s,uint256).tTransferAmount (6ense.sol#1438)
```

```
Variable Token.reflectionFromToken(uint256,bool).rTransferAmount (6ense.sol#1057) is too similar to Token._transferBothExclude
d(address,address,uint256).tTransferAmount (6ense.sol#1112)
Variable Token._transferToExcluded(address,address,uint256).rTransferAmount (6ense.sol#1438) is too similar to Token._transfer
Standard(address,address,uint256).tTransferAmount (6ense.sol#1428)
Variable Token._transferFromExcluded(address,address,uint256).rTransferAmount (6ense.sol#1449) is too similar to Token._transf
erToExcluded(address,address,uint256).tTransferAmount (6ense.sol#1438)
Variable Token._transferFromExcluded(address,address,uint256).rTransferAmount (6ense.sol#1449) is too similar to Token._transf
erFromExcluded(address,address,uint256).tTransferAmount (6ense.sol#1449)
Variable Token._transferBothExcluded(address,address,uint256).rTransferAmount (6ense.sol#1112) is too similar to Token._transf
erFromExcluded(address,address,uint256).tTransferAmount (6ense.sol#1449)
Variable Token._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (6ense.sol#1196) is too similar to Token._
transferFromExcluded(address,address,uint256).tTransferAmount (6ense.sol#1449)
Variable Token._getRValues(uint256,uint256,uint256,uint256,uint256).rTransferAmount (6ense.sol#1196) is too similar to Token._
transferToExcluded(address,address,uint256).tTransferAmount (6ense.sol#1438)
Variable Token._transferBothExcluded(address,address,uint256).rTransferAmount (6ense.sol#1112) is too similar to Token._transf
erToExcluded(address,address,uint256).tTransferAmount (6ense.sol#1438)
Variable Token._getValues(uint256).rTransferAmount (6ense.sol#1179) is too similar to Token._getTValues(uint256).tTransferAmou
nt (6ense.sol#1187)
Variable Token._getValues(uint256).rTransferAmount (6ense.sol#1179) is too similar to Token._getValues(uint256).tTransferAmoun
t (6ense.sol#1178)
Variable Token._getValues(uint256).rTransferAmount (6ense.sol#1179) is too similar to Token._transferBothExcluded(address,addr
ess,uint256).tTransferAmount (6ense.sol#1112)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

Token.slitherConstructorVariables() (6ense.sol#916-1458) uses literals with too many digits:
        - _tTotal = 9630000000 * 10 ** 18 (6ense.sol#932)
Token.slitherConstructorVariables() (6ense.sol#916-1458) uses literals with too many digits:
        - _maxTxAmount = 5000000 * 10 ** 18 (6ense.sol#960)
Token.slitherConstructorVariables() (6ense.sol#916-1458) uses literals with too many digits:
        - numTokensSellToAddToLiquidity = 500000 * 10 ** 18 (6ense.sol#961)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Token._burnFee (6ense.sol#947) should be constant
Token._tTotal (6ense.sol#932) should be constant
Token.numBlocksForBlacklist (6ense.sol#939) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
6ense.sol analyzed (11 contracts with 84 detectors), 112 result(s) found
```

Solidity Static Analysis

6ense.sol

Check-effects-interaction:

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

more

Pos: 977:4:

Check-effects-interaction:

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

more

Pos: 1373:4:

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: 1402:12:

Gas costs:

Gas requirement of function Token.includeInReward 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: 1098:4:

Email: audit@EtherAuthority.io

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: 1100:8:

ERC20:

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

more

Pos: 729:4:

Constant/View/Pure functions:

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

more

Pos: 1051:4:

Similar variable names:

Token._transferFromExcluded(address,address,uint256) : Variables have very similar names "rFee" and "tFee". Note: Modifiers are currently not considered by this static analysis.

Pos: 1455:26:

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: 1293:12:

Solhint Linter

6ense.sol

```
Compiler version 0.8.19 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
Error message for require is too long
Pos: 9:141
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:238
Error message for require is too long
Pos: 9:280
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:337
Error message for require is too long
Pos: 9:504
Error message for require is too long
Pos: 9:534
Error message for require is too long
Pos: 9:535
Error message for require is too long
Pos: 9:540
Error message for require is too long
Pos: 9:592
Error message for require is too long
Pos: 9:597
Error message for require is too long
Pos: 9:627
Error message for require is too long
Pos: 9:628
Code contains empty blocks
Pos: 24:677
Code contains empty blocks
Pos: 24:697
Function name must be in mixedCase
Pos: 5:737
Function name must be in mixedCase
Pos: 5:738
Function name must be in mixedCase
Pos: 5:755
Function name must be in mixedCase
Pos: 5:777
Contract has 24 states declarations but allowed no more than 15
Pos: 1:915
Variable name must be in mixedCase
Pos: 5:935
Explicitly mark visibility of state
Pos: 5:954
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
```

```
Pos: 5:976
Error message for require is too long
Pos: 9:1043
Error message for require is too long
Pos: 9:1062
Error message for require is too long
Pos: 9:1083
Error message for require is too long
Pos: 9:1132
Error message for require is too long
Pos: 9:1137
Error message for require is too long
Pos: 9:1142
Code contains empty blocks
Pos: 32:1168
Error message for require is too long
Pos: 9:1260
Error message for require is too long
Pos: 9:1261
Error message for require is too long
Pos: 9:1268
Error message for require is too long
Pos: 9:1269
Error message for require is too long
Pos: 9:1270
Error message for require is too long
Pos: 13:1292
Avoid making time-based decisions in your business logic
Pos: 13:1386
Avoid making time-based decisions in your business logic
Pos: 13:1401
```

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