auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

NODE

DATED : 08 Mar 23'

AUDIT SUMMARY

Project name – NODE

Date: 08 March, 2023

Audit Status: **Passed with High Risk**

The audit result indicates that the token's logic sets certain restrictions on investors' ability to sell or transfer their tokens during the initial days after the launch, which presents a high centralization risk. It is crucial for investors to have the flexibility to manage their assets as needed, and any restrictions on selling or transferring tokens may be perceived as overly controlling or disadvantageous to investors

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	2	0	0	1
Acknowledged	0	0	0	0	0
Resolved	0	0	0	0	0

USED TOOLS

Tools:

1- Manual Review:
a line by line code review has been performed by audit ace team.

2- BSC Test Network:
all tests were done on BSC Test network, each test has its transaction has attached to it.

3- Slither : Static Analysis

Testnet Link: all tests were done using this contract, tests are done on BSC Testnet

https://testnet.bscscan.com/token/0x4b1E4e74aD727C61d0525aDf81Ec93002f4ae8BC

Token Information

Token Name : NODE

Token Symbol: NODE

Decimals: 18

Token Supply: 200,000,000

Token Address: -

Checksum:
9d5807e6ab1e61ae9bc1e4c51deed27a933d2fe4

Owner:-

Note: buyers cannot sell their tokens during the first 3 days after launch if they are in profit, during the next day (day 3-4) if they are in more than 50% profit, and on day 4 if they are in more than 100% profit

TOKEN OVERVIEW

Fees:

Buy Fees: upto 25%

Sell Fees: upto 25%

Transfer Fees: upto 25%

Fees Privilige: Owner

Ownership : Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Priviliges: changing fees - enable/disable sellCheck

AUDIT METHODOLOGY

The auditing process will follow a routine as special considerations by Auditace:

- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.
- Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.
- Test coverage analysis determines whether the test cases are covering the code and how much code isexercised when we run the test cases.
- Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.
- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

VULNERABILITY CHECKLIST

- [x] Return values of low-level calls
- [x] Private modifier
- [x] Multiple Sends
- [x] Using Suicide
- [x] Gas Limitand Loops
- [x] Address hardcoded
- [x] Exception Disorder
- [x] Using inline assembly
- [x] Divide before multiply
- [x] Missing Zero Address Validation
- [x] Compiler version not fixed
- [x] **Gasless Send**
- [x] Using block.timestamp
- [x] Re-entrancy
- [x] Tautology or contradiction
- [x] Timestamp Dependence
- [x] Revert/require functions
- [x] Use of tx.origin
- [x] Integer overflow/underflow
- [x] Dangerous strict equalities
- [x] Using SHA3
- [x] Using throw

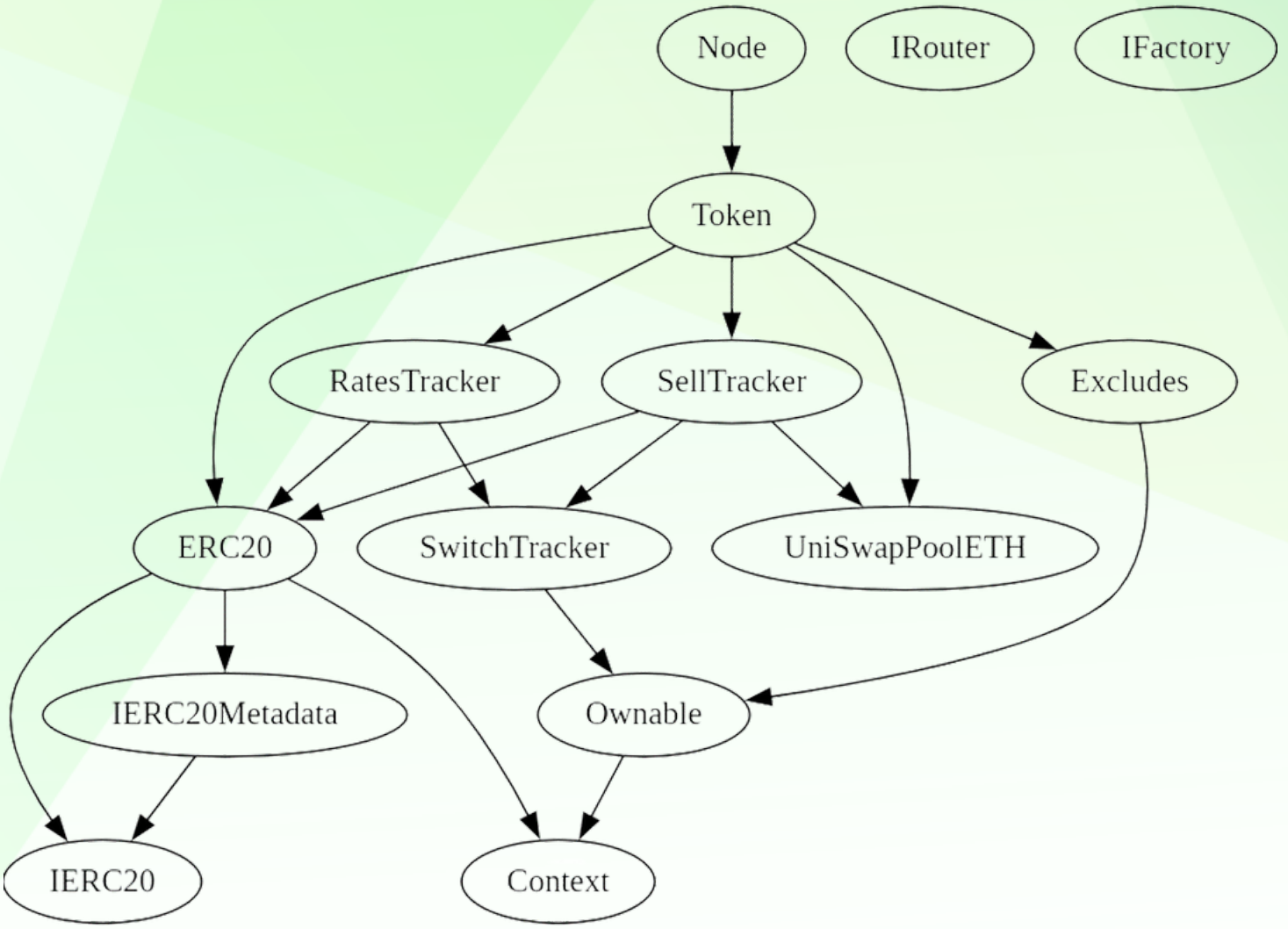
CLASSIFICATION OF RISK

Severity	Description
Critical	These vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
High-Risk	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
Medium-Risk	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
Low-Risk	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
Gas Optimization /Suggestion	A vulnerability that has an informational character but is not affecting any of the code.

Findings

Severity	Found
Critical	0
High-Risk	2
Medium-Risk	0
Low-Risk	0
Gas Optimization / Suggestions	1

INHERITANCE TREE



POINTS TO NOTE

- Owner must enable trading in order for investors to be able to trade (openTrade function)
- Investors must be aware of sellCheck function in this contract, how it works and its risks, so please read the audit report carefully
- Owner is able to set buy-sell-transfer fees up to 25%
- Owner is not able to set max buy/sell/transfer/hold amount
- Owner is not able to blacklist an arbitrary wallet
- Owner is not able to disable trades
- Owner is not able to mint new tokens

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
IERC20Metadata	Interface	IERC20		
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
ERC20	Implementation	Context, IERC20, IERC20Metadata		
└	<Constructor>	Public ❗	🛑	NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	decimals	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	transfer	Public ❗	🛑	NO❗
└	allowance	Public ❗		NO❗
└	approve	Public ❗	🛑	NO❗
└	transferFrom	Public ❗	🛑	NO❗
└	increaseAllowance	Public ❗	🛑	NO❗
└	decreaseAllowance	Public ❗	🛑	NO❗
└	_transfer	Internal 🔒	🛑	
└	_takeTransfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_spendAllowance	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
└	_afterTokenTransfer	Internal 🔒	🛑	

CONTRACT ASSESMENT

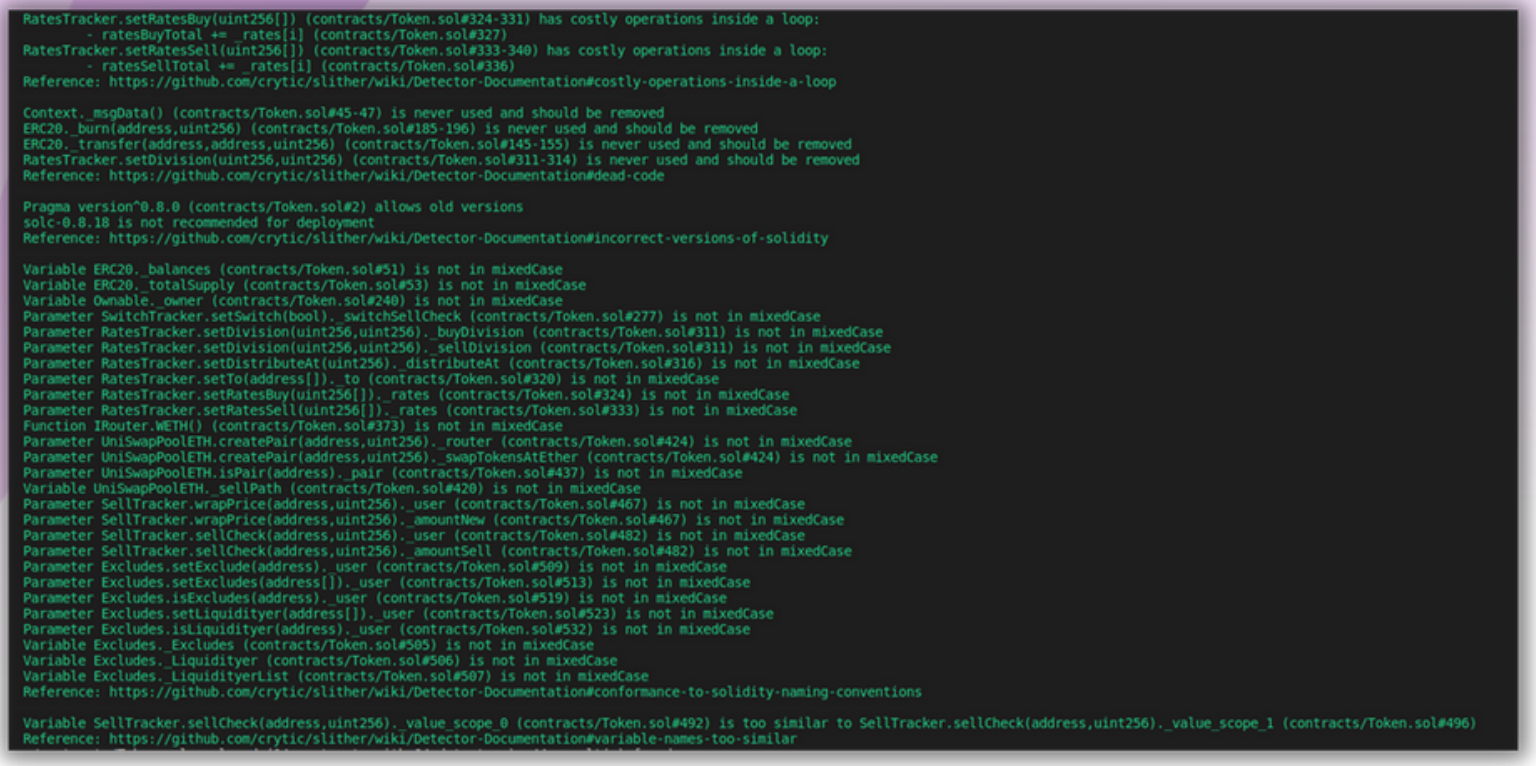
Ownable	Implementation	Context	
└	<Constructor>	Public ❗	🛑 |NO❗ |
└	owner	Public ❗	| |NO❗ |
└	_checkOwner	Internal 🔒	| |
└	renounceOwnership	Public ❗	🛑 | onlyOwner |
└	_transferOwnership	Internal 🔒	🛑 | |
SwitchTracker	Implementation	Ownable	
└	setSwitch	Public ❗	🛑 | onlyOwner |
RatesTracker	Implementation	ERC20, SwitchTracker	
└	<Constructor>	Public ❗	🛑 |NO❗ |
└	setDivision	Internal 🔒	🛑 | |
└	setDistributeAt	Public ❗	🛑 | onlyOwner |
└	setTo	Public ❗	🛑 | onlyOwner |
└	setRatesBuy	Public ❗	🛑 | onlyOwner |
└	setRatesSell	Public ❗	🛑 | onlyOwner |
└	processBuyFees	Internal 🔒	🛑 | |
└	processSellFees	Internal 🔒	🛑 | |
└	distribute	Internal 🔒	🛑 | |
IRouter	Interface	| |
└	factory	External ❗	| |NO❗ |
└	WETH	External ❗	| |NO❗ |
└	addLiquidityETH	External ❗	💵 |NO❗ |
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑 |NO❗ |
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑 |NO❗ |
└	getAmountsOut	External ❗	| |NO❗ |
IFactory	Interface	| |
└	createPair	External ❗	🛑 |NO❗ |
UniSwapPoolETH	Implementation	| |
└	<Receive Ether>	External ❗	💵 |NO❗ |
└	createPair	Internal 🔒	🛑 | |
└	isPair	Internal 🔒	| |
SellTracker	Implementation	ERC20, UniSwapPoolETH, SwitchTracker	
└	recordStartAt	Internal 🔒	🛑 | |
└	getPrice	Internal 🔒	| |
└	wrapPrice	Internal 🔒	🛑 | |

CONTRACT ASSESMENT

└	sellCheck	Internal 🔒	🛑	
Excludes	Implementation	Ownable		
└	setExclude	Internal 🔒	🛑	
└	setExcludes	Public ❗	🛑	onlyOwner
└	isExcludes	Internal 🔒		
└	setLiquidityer	Public ❗	🛑	onlyOwner
└	isLiquidityer	Internal 🔒		
└	removeLiquidityer	Internal 🔒	🛑	
Token	Implementation	ERC20, UniSwapPoolETH, RatesTracker, SellTracker, Excludes		
└	<Constructor>	Public ❗	🛑	ERC20
└	openTrading	Public ❗	🛑	onlyOwner
└	_transfer	Internal 🔒	🛑	
└	handSwap	Internal 🔒	🛑	
Node	Implementation	Token		
└	<Constructor>	Public ❗	🛑	Token

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS



```
RatesTracker.setRatesBuy(uint256[]) (contracts/Token.sol#324-331) has costly operations inside a loop:
        - ratesBuyTotal += _rates[i] (contracts/Token.sol#327)
RatesTracker.setRatesSell(uint256[]) (contracts/Token.sol#333-340) has costly operations inside a loop:
        - ratesSellTotal += _rates[i] (contracts/Token.sol#336)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

Context._msgData() (contracts/Token.sol#45-47) is never used and should be removed
ERC20._burn(address,uint256) (contracts/Token.sol#185-196) is never used and should be removed
ERC20._transfer(address,address,uint256) (contracts/Token.sol#145-155) is never used and should be removed
RatesTracker.setDivision(uint256,uint256) (contracts/Token.sol#311-314) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (contracts/Token.sol#2) allows old versions
solc-0.8.18 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Variable ERC20._balances (contracts/Token.sol#51) is not in mixedCase
Variable ERC20._totalSupply (contracts/Token.sol#53) is not in mixedCase
Variable Ownable._owner (contracts/Token.sol#240) is not in mixedCase
Parameter SwitchTracker.setSwitch(bool)._switchSellCheck (contracts/Token.sol#277) is not in mixedCase
Parameter RatesTracker.setDivision(uint256,uint256)._buyDivision (contracts/Token.sol#311) is not in mixedCase
Parameter RatesTracker.setDivision(uint256,uint256)._sellDivision (contracts/Token.sol#311) is not in mixedCase
Parameter RatesTracker.setDistributeAt(uint256)._distributeAt (contracts/Token.sol#316) is not in mixedCase
Parameter RatesTracker.setTo(address[])._to (contracts/Token.sol#320) is not in mixedCase
Parameter RatesTracker.setRatesBuy(uint256[])._rates (contracts/Token.sol#324) is not in mixedCase
Parameter RatesTracker.setRatesSell(uint256[])._rates (contracts/Token.sol#333) is not in mixedCase
Function IRouter.WETH() (contracts/Token.sol#373) is not in mixedCase
Parameter UniSwapPoolETH.createPair(address,uint256)._router (contracts/Token.sol#424) is not in mixedCase
Parameter UniSwapPoolETH.createPair(address,uint256)._swapTokensAtEther (contracts/Token.sol#424) is not in mixedCase
Parameter UniSwapPoolETH.isPair(address)._pair (contracts/Token.sol#437) is not in mixedCase
Variable UniSwapPoolETH._sellPath (contracts/Token.sol#420) is not in mixedCase
Parameter SellTracker.wrapPrice(address,uint256)._user (contracts/Token.sol#467) is not in mixedCase
Parameter SellTracker.wrapPrice(address,uint256)._amountNew (contracts/Token.sol#467) is not in mixedCase
Parameter SellTracker.sellCheck(address,uint256)._user (contracts/Token.sol#482) is not in mixedCase
Parameter SellTracker.sellCheck(address,uint256)._amountSell (contracts/Token.sol#482) is not in mixedCase
Parameter Excludes.setExclude(address)._user (contracts/Token.sol#509) is not in mixedCase
Parameter Excludes.setExcludes(address[])._user (contracts/Token.sol#513) is not in mixedCase
Parameter Excludes.isExcludes(address)._user (contracts/Token.sol#519) is not in mixedCase
Parameter Excludes.setLiquidityer(address[])._user (contracts/Token.sol#523) is not in mixedCase
Parameter Excludes.isLiquidityer(address)._user (contracts/Token.sol#532) is not in mixedCase
Variable Excludes._Excludes (contracts/Token.sol#505) is not in mixedCase
Variable Excludes._Liquidityer (contracts/Token.sol#506) is not in mixedCase
Variable Excludes._LiquidityerList (contracts/Token.sol#507) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable SellTracker.sellCheck(address,uint256)._value_scope_0 (contracts/Token.sol#492) is too similar to SellTracker.sellCheck(address,uint256)._value_scope_1 (contracts/Token.sol#496)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
```

Result => A static analysis of contract's source code has been performed using slither,

No major issues were found in the output

FUNCTIONAL TESTING

Router (PCS V2):
0xD99D1c33F9fC3444f8101754aBC46c52416550D1

All the functionalities have been tested, no issues were found

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0x27be83491705c583b01d37f537d01b4260dee93b8bdbec0074b39cd21676461c

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0xaa167a3a9291874722824484a65cc4272dbcaaba11d9e2d753b730dddeb4971d

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0xa45825e7a85efbe844bb3a70ed2d3f38c909e6ce84d0cf215720a90c679ee257

4- Transferring when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x1b8ed7fae64b3411843841375d0c4bc6888c59900c7dda46625f0ff602b51160

FUNCTIONAL TESTING

5- Buying when not excluded from fees (can have up to 25% tax) (passed):
https://testnet.bscscan.com/tx/0xeca2a3b671e9a94402505aae20b96f920b8365f58ac784358881ed5e3fb40440

6- Selling when not excluded from fees (can have up to 25% tax) (passed):
https://testnet.bscscan.com/tx/0x4713e7afa4951fa8f9fb450f72abac4548015eaec479e3922b758fdab894712c

7- Transferring when not excluded from fees (can have up to 25% tax) (passed):
https://testnet.bscscan.com/tx/0x6d9f43705ea2e6b409e5be2284f97b3d39878234d26e36394d54b38839c903fe

8-Collected fees going to destinations (passed):
as can be seen in below sell transaction, all the tax receivers, received a portion of collected fees:
https://testnet.bscscan.com/tx/0x4713e7afa4951fa8f9fb450f72abac4548015eaec479e3922b758fdab894712c

MANUAL TESTING

Issue: owner's ability to restart **sellcheck**

Type : centralization

Severity: **High**

Function: sellCheck – setSwitch

Overview: The owner has the ability to restart the sellCheck functionality even after the initial 4-day period following the launch. This represents a significant centralization risk as the owner could restrict selling even during low-volume periods. The reason is that block.timestamp > sellPresets[2] condition will always be true once it is met, which may result in arbitrary on-off of restrictions on selling.

```
uint256 costValue = buyPriceMap[_user] * _amountSell;
uint256 currentValue = getPrice(_amountSell) * _amountSell;
if (block.timestamp > sellPresets[0]) {
    uint256 _value = (costValue * sellAmountTimes[0]) / 1 ether;
    require(currentValue <= _value - sellValueMap[_user]);
    sellValueMap[_user] += currentValue;
} else if (block.timestamp > sellPresets[1]) {
    uint256 _value = (costValue * sellAmountTimes[1]) / 1 ether;
    require(currentValue <= _value - sellValueMap[_user]);
    sellValueMap[_user] += currentValue;
} else if (block.timestamp > sellPresets[2]) {
    uint256 _value = (costValue * sellAmountTimes[2]) / 1 ether;
    require(currentValue <= _value - sellValueMap[_user]);
    sellValueMap[_user] += currentValue;
```

Recommendations

- The **block.timestamp > sellPresets[2]** condition has a permanent nature and could be a significant centralization risk. To avoid indefinite restrictions on selling, it is recommended that the owner implement measures to disable this condition once the **sellCheck** function has been active for the predefined period. This could be achieved by changing the condition to **block.timestamp < sellPresets[0]**, which ensures that the function is disabled permanently after the four-day period has elapsed.
- Clear and transparent communication should be provided to potential buyers and investors regarding the functionality of the sellCheck function, to ensure that they fully understand the implications of this control over the selling volume.

MANUAL TESTING

IIssue: owner’s ability to control sells
Type : centralization
Severity: **High**
Function: **sellCheck – setSwitch**

Overview: The sellCheck function enables the owner to control the selling of tokens for a period of 4 days after the launch. During this period**, buyers are restricted from selling their tokens if they are making a profit more than a defined amount (to restrict huge sells**).
Specifically, buyers cannot sell their tokens during the first 3 days after launch if they are in profit, during the next day (day 3-4) if they are in more than 50% profit, and on day 4 if they are in more than 100% profit.

Recommendations

- While the **sellCheck** function does provide a level of control over the selling volume, it is important to note that it may impact buyers' ability to sell their tokens during the first 4 days after the launch. However, it should be noted that this feature is not necessarily an issue, but rather a design decision made by the owner.
- That being said, it is recommended that the owner provide clear and transparent communication about this feature to potential buyers and investors, to ensure they fully understand the implications of this control over the selling volume. Additionally, it may be beneficial to consider implementing a more flexible selling control mechanism that still allows buyers to sell their tokens without unnecessary restrictions.

MANUAL TESTING

Issue: owner's ability to change buy/sell fees

Type : centralization

Severity: **Informational**

Function: setRatesBuy – set RatesSell

Overview: The owner has the ability to set buy and sell fees up to 25% each, which can result in a combined fee of 50% of the buy-sell trade. This represents a significant risk to investors if they are set to their max limit.

Also this must be noted that transfer fee is calculated using sell fee (I.e they are always equal)

```
function setRatesBuy(uint256[] memory _rates) public onlyOwner {
    ratesBuyTotal = 0;
    for (uint i = 0; i < _rates.length; i++) {
        ratesBuyTotal += _rates[i];
    }
    require(ratesBuyTotal < 2500, "rates must less than 25%");
    buyRates = _rates;
}

ftrace | funcSig
function setRatesSell(uint256[] memory _rates) public onlyOwner {
    ratesSellTotal = 0;
    for (uint i = 0; i < _rates.length; i++) {
        ratesSellTotal += _rates[i];
    }
    require(ratesSellTotal < 2500, "rates must less than 25%");
    sellRates = _rates;
}
```

Social Media Overview

Here are the Social Media Accounts of

Node

https://t.me/nodebsc

https://twitter.com/nodebsc

https://node.nxdefi.net/

DISCLAIMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed. The Auditace team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Auditace receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token. The Auditace team disclaims any liability for the resulting losses.

ABOUT AUDITACE

We specializes in providing thorough and reliable audits for Web3 projects. With a team of experienced professionals, we use cutting-edge technology and rigorous methodologies to evaluate the security and integrity of blockchain systems. We are committed to helping our clients ensure the safety and transparency of their digital assets and transactions.

https://auditace.tech/

https://t.me/Audit_Ace

https://twitter.com/auditace_

https://github.com/Audit-Ace