EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	Lynx Finance
Website:	https://lynxfinance.net
Platform:	Avalanche
Language:	Solidity
Date:	December 17th, 2022

Table of contents

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.

Introduction

EtherAuthority was contracted by the Lynx Finance team to perform the Security audit of the Lynx Finance 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 December 17th, 2022.

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

- LYNX Finance offers the $LYNX token.
- $LYNX is a deflationary token on Avalanche (holders will receive rewards in USDC.e).
- Lynx Contracts have functions like swapping, launch, etc.

Audit scope

Name	**Code Review and Security Analysis Report for Lynx Token Smart Contract**
Platform	**Avalanche / Solidity**
File	Lynx.sol
File MD5 Hash	E9E74FA846B45592998B4A10C0588BCE
Updated File MD5 Hash	0B0E0DED38AC3A1D13F5570F4A10BA01
Online Code Link	https://github.com/Volfsorg/Lynx/blob/main/Lynx.sol
Audit Date	December 17th, 2022
Revised Audit Date	December 22nd, 2022

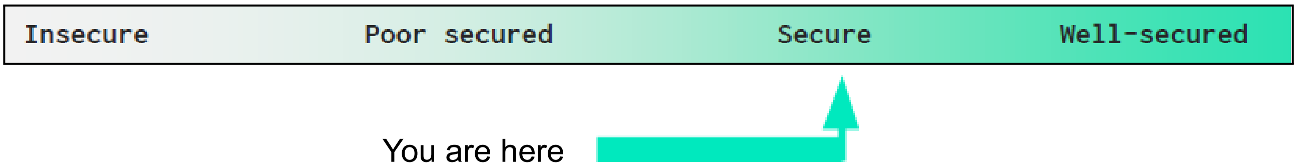
Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: ● Name: Lynx ● Symbol: LYNX ● Decimals: 18 ● Total Supply: 100 Million $LYNX	YES, This is valid.
Buy Fee: 10% ● Burn Fee: 5% ● Treasury Fee: 4% ● Dev Fee: 1% **Sell Fee: 15%** ● Reward Fee: 10% ● Burn Fee: 2% ● Treasury Fee: 2% ● Dev Fee: 1%	YES, This is valid.
Ownership Control: ● Owner can launch only once. ● Owner can set transfer enabled status.	YES, This is valid.
Authorized Control: ● Authorized can set swap settings. ● Authorized can set distributor gas settings. ● Authorized can set a new distributor. ● Authorized can set distribution addresses. ● Authorized can set distribution criteria. ● Authorized can set dex pair addresses. ● Authorized can set treasury fee receiver, developer fee receiver. ● Authorized can set transfer fees, sell fees,	YES, This is valid.

buy fees. - Authorized can set reflection token addresses.	
Other Specifications: - Fee Denominator: 1000 - Swap Maximum: 1% - Launch Price: $0.0015	**YES, This is valid.**

Audit Summary

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



We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.

All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 1 critical, 0 high, 0 medium and 3 low and some very low level issues.

All the issues have been resolved in the revised 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 the Lynx Finance 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 Lynx Token.

The Lynx Finance 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 used, which is a good thing.

Documentation

We were given a Lynx Token smart contract code in the form of a Github weblink. 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.

Another source of information was its website: https://lynxfinance.net/ which provided rich information about the project architecture.

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	receive	external	Passed	No Issue
3	checkTxLimit	internal	Passed	No Issue
4	launched	internal	Passed	No Issue
5	_basicTransfer	internal	Passed	No Issue
6	_transferFrom	internal	Passed	No Issue
7	shouldTakeFee	internal	Passed	No Issue
8	takeFee	internal	Passed	No Issue
9	shouldSwapBack	internal	Passed	No Issue
10	swapBack	internal	Passed	No Issue
11	buyTokens	internal	Removed	No Issue
12	allowance	external	Passed	No Issue
13	approve	write	Passed	No Issue
14	approveMax	external	Passed	No Issue
15	balanceOf	read	Passed	No Issue
16	decimals	external	Passed	No Issue
17	name	external	Passed	No Issue
18	symbol	external	Passed	No Issue
19	totalSupply	external	Passed	No Issue
20	transfer	external	Passed	No Issue
21	transferFrom	external	Passed	No Issue
22	getCirculatingSupply	read	Passed	No Issue
23	getDexPair	external	Passed	No Issue
24	getDexPair2	external	Passed	No Issue
25	getDexPair3	external	Passed	No Issue
26	getIsFree	read	access only Owner	No Issue
27	getMinDistribution	external	Passed	No Issue
28	getMinPeriod	external	Passed	No Issue
29	getOwner	external	Passed	No Issue
30	getReflectionToken	external	Passed	No Issue
31	getSwapAmount	read	Passed	No Issue
32	getTotalBuyFee	read	Passed	No Issue
33	getTotalSellFee	read	Passed	No Issue
34	getTotalTransferFee	read	Passed	No Issue
35	launch	write	access only Owner	No Issue
36	swapBackManual	external	access only authorized	No Issue
37	sweep	external	Removed	No Issue
38	setReflectionToken	external	access only authorized	No Issue
39	setTransferEnabled	write	access only Owner	No Issue
40	setMaxWallet	external	access only authorized	No Issue
41	setTxLimit	external	access only authorized	No Issue
42	setBuyFees	external	Passed	No Issue

43	setSellFees	external	Passed	No Issue
44	setTransferFees	external	Passed	No Issue
45	setFeeReceivers	external	access only authorized	No Issue
46	setFree	write	access only Owner	No Issue
47	unSetFree	write	access only Owner	No Issue
48	setIsDividendExempt	external	access only authorized	No Issue
49	setIsFeeExempt	external	access only authorized	No Issue
50	setIsTxLimitExempt	external	access only authorized	No Issue
51	setDexPair	external	access only authorized	No Issue
52	setDexPair2	external	access only authorized	No Issue
53	setDexPair3	external	access only authorized	No Issue
54	setDistributionCriteria	external	access only authorized	No Issue
55	setDistributorAddress	external	access only authorized	No Issue
56	setNewDistributor	external	access only authorized	No Issue
57	setDistributorSettings	external	access only authorized	No Issue
58	setSwapBackSettings	external	access only authorized	No Issue
59	swapping	modifier	Passed	No Issue
60	onlyOwner	modifier	Passed	No Issue
61	authorized	modifier	Passed	No Issue
62	authorize	write	Passed	No Issue
63	unauthorize	write	Passed	No Issue
64	isAuthorized	read	Passed	No Issue
65	isOwner	read	Passed	No Issue
66	renounceOwnership	write	access only Owner	No Issue
67	transferOwnership	write	access only Owner	No Issue
68	initialization	modifier	Removed	No Issue
69	onlyToken	modifier	Passed	No Issue
70	claimDividend	external	Passed	No Issue
71	deposit	external	Passed	No Issue
72	distributeDividend	internal	Passed	No Issue
73	process	external	Passed	No Issue
74	shouldDistribute	internal	Passed	No Issue
75	getCumulativeDividends	internal	Passed	No Issue
76	getMinDistribution	external	Passed	No Issue
77	getMinPeriod	external	Passed	No Issue
78	getUnpaidEarnings	read	Passed	No Issue
79	setDistributionCriteria	external	Passed	No Issue
80	setReflectionToken	external	Passed	No Issue
81	setShare	external	Passed	No Issue
82	addShareholder	internal	Passed	No Issue
83	removeShareholder	internal	Passed	No Issue

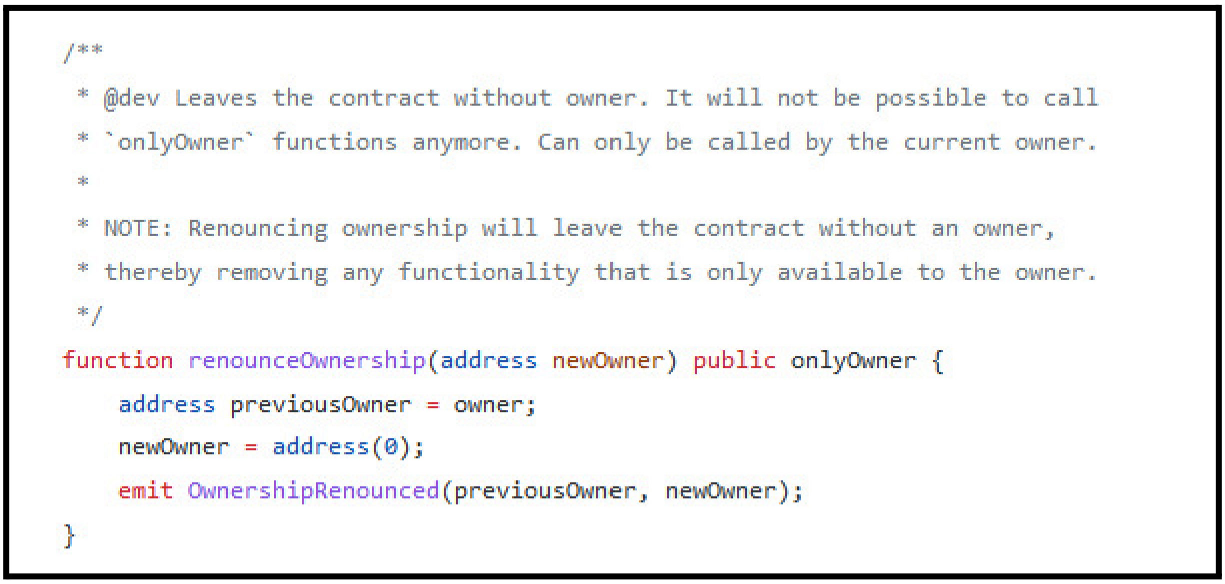
Severity Definitions

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.

Audit Findings

Critical Severity

(1) Not renouncing the contract ownership:



```
/**
 * @dev Leaves the contract without owner. It will not be possible to call
 * `onlyOwner` functions anymore. Can only be called by the current owner.
 *
 * NOTE: Renouncing ownership will leave the contract without an owner,
 * thereby removing any functionality that is only available to the owner.
 */
function renounceOwnership(address newOwner) public onlyOwner {
    address previousOwner = owner;
    newOwner = address(0);
    emit OwnershipRenounced(previousOwner, newOwner);
}
```

This function should be used to renounce ownership so that the contract will be without an owner. But here the owner does not get set by address(0). So this will do nothing.This contract has always an owner.

Resolution: We suggest setting the owner to address(0) to renounce the contract ownership.
Status: **This issue is fixed in the revised contract code.**

High Severity

No High severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

(1) The owner can drain contract funds:

By using a sweep function the owner can drain contract funds.

Resolution: We suggest confirming this functionality.

Status: This issue is fixed in the revised contract code.

(2) Function input parameters lack of check:

Some functions require validation before execution.

Functions are:

LynxAuthorization

- authorize() - onlyOwner
- unauthorize()

LynxDividendDistributor

- setDistributionCriteria() - onlyToken
- setReflectionToken() - onlyToken

Resolution: We suggest using validation like for numerical variables that should be greater than 0 and for address type check variables that are not address(0).

Status: This issue is fixed in the revised contract code.

(3) Critical operation lacks event log:

Missing event log for:

LynxDividendDistributor

- claimDividend()
- deposit() - onlyToken
- process() - onlyToken
- setShare() - onlyToken

Lynx

- setBuyFees() - authorized
- setSellFees() - authorized
- setTransferFees() - authorized

Resolution: Please write an event log for listed events.

Status: This issue is fixed in the revised contract code.

Very Low / Informational / Best practices:

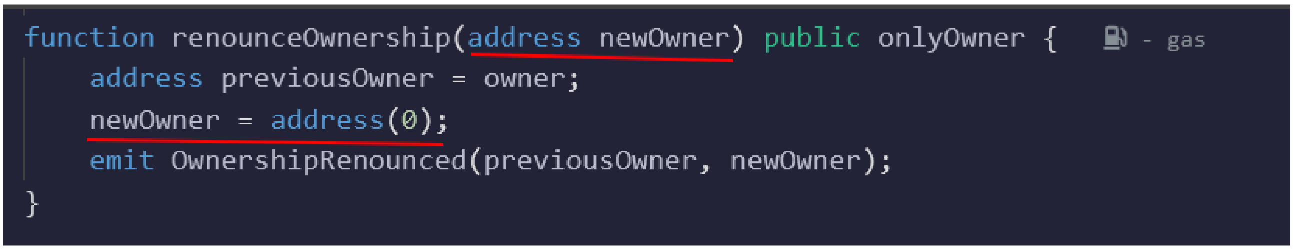
(1) SafeMath Library:

SafeMath Library is used in this contract code, but the compiler version is greater than or equal to 0.8.0, Then it will be not required to use, solidity automatically handles overflow/underflow.

Resolution: Remove the SafeMath library and use normal math operators, It will improve code size, and less gas consumption.

Status: This issue is fixed in the revised contract code.

(2) Unused function parameter: **- LynxAuthorization**



```
function renounceOwnership(address newOwner) public onlyOwner {
    address previousOwner = owner;
    newOwner = address(0);
    emit OwnershipRenounced(previousOwner, newOwner);
}
```

There is a function renounceOwnership() that asks the parameter "newOwner" but it's not required and not used in this function, because in this function newOwner will always be address(0).

Resolution: We suggest removing parameters from these function calls.

Status: This issue is fixed in the revised contract code.

(3) Unused modifier / Internal function / variable:

LynxDividendDistributor

The initialization() modifier is defined but not used.

The initialized variable has been set but not used anywhere.

Lynx

There are some internal functions defined but not used:

- launched()

Resolution: We suggest removing unused modifier / Internal function / variable.

Status: This issue is fixed in the revised contract code.

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:

- getIsFree: Owner can check if the holder address is free or not.
- launch: Owner can launch only be called 1 time.
- swapBackManual: Authorized can swap back manual amount.
- setReflectionToken: Authorized can set reflection token address.
- setTransferEnabled: Owner can set transfer enabled status.
- setMaxWallet: Authorized can set maximum wallet amount.
- setTxLimit: Authorized can set transaction limit amount.
- setBuyFees: Authorized can set buy reflection fee, buy burn fee, buy treasury fee, buy developer fee.
- setSellFees: Authorized can set sell reflection fee, sell burn fee, sell treasury fee, sell developer fee.
- setTransferFees: Authorized can set transfer reflection fee, transfer burn fee, transfer treasury fee, transfer developer fee.
- setFeeReceivers: Authorized can set treasury fee receiver, developer fee receiver.
- setFree: Owner can set free holder address.

- unSetFree: Owner can unset free holder address.
- setIsDividendExempt: Authorized can set if it is dividend exempt status.
- setIsFeeExempt: Authorized can set if it is fee exempt status.
- setDexPair: Authorized can set dex pair address.
- setDexPair2: Authorized can set dex pair 2 address.
- setDexPair3: Authorized can set dex pair 3 address.
- setDistributionCriteria: Authorized can set distribution criteria.
- setDistributorAddress: Authorized can set distribution address.
- setNewDistributor: Authorized can set new distributor.
- setDistributorSettings: Authorized can set distributor gas settings.
- setSwapBackSettings: Authorized can set swap settings.

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

Conclusion

We were given a contract code in the form of a Github weblink. And we have used all possible tests based on given objects as files. We have observed 1 critical, 3 low severity issues and some informational issues in the smart contracts. **All the issues have been resolved** in the revised code. **So the smart contract is ready for the mainnet deployment**.

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.

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

The 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).

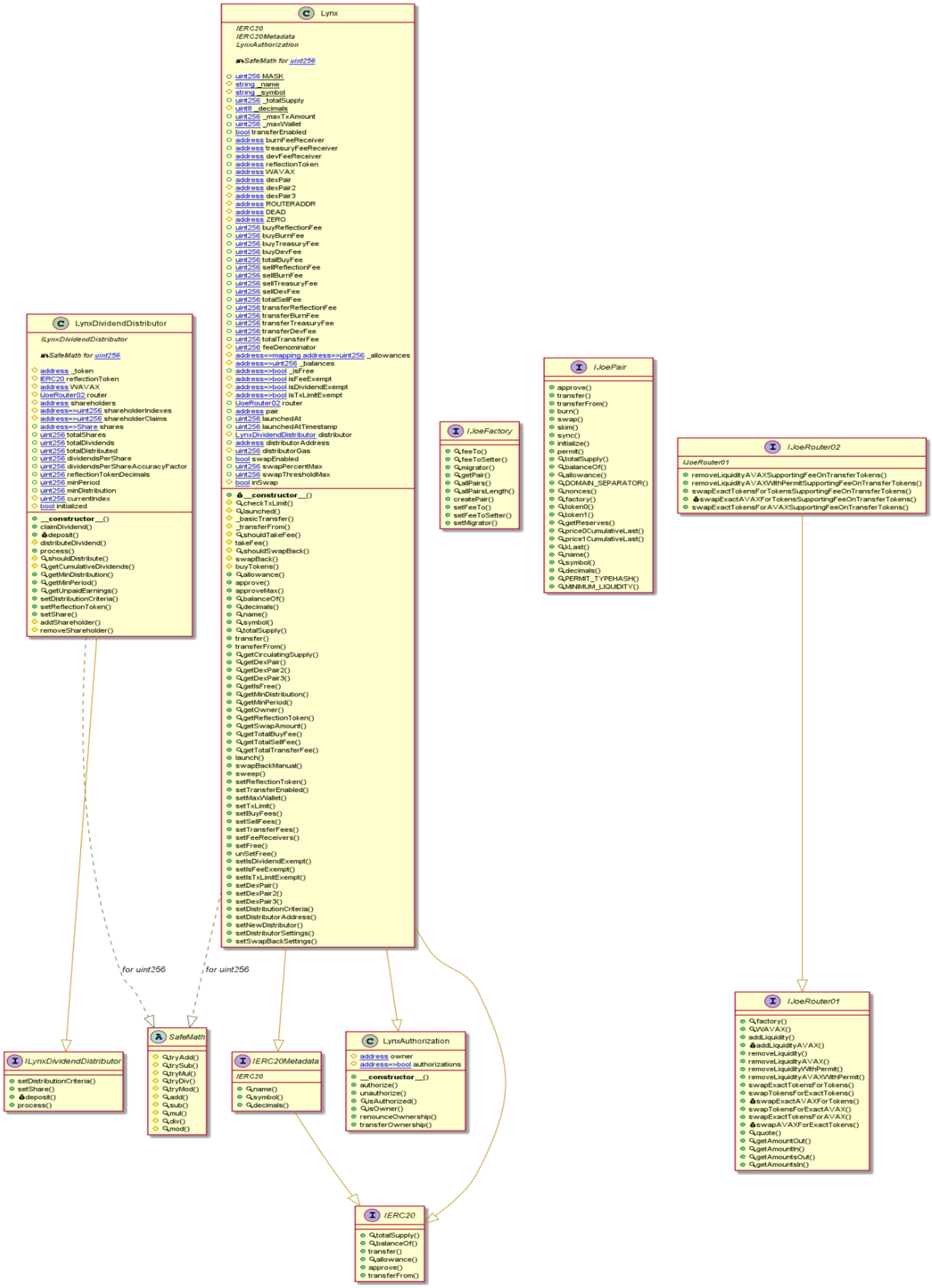
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we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

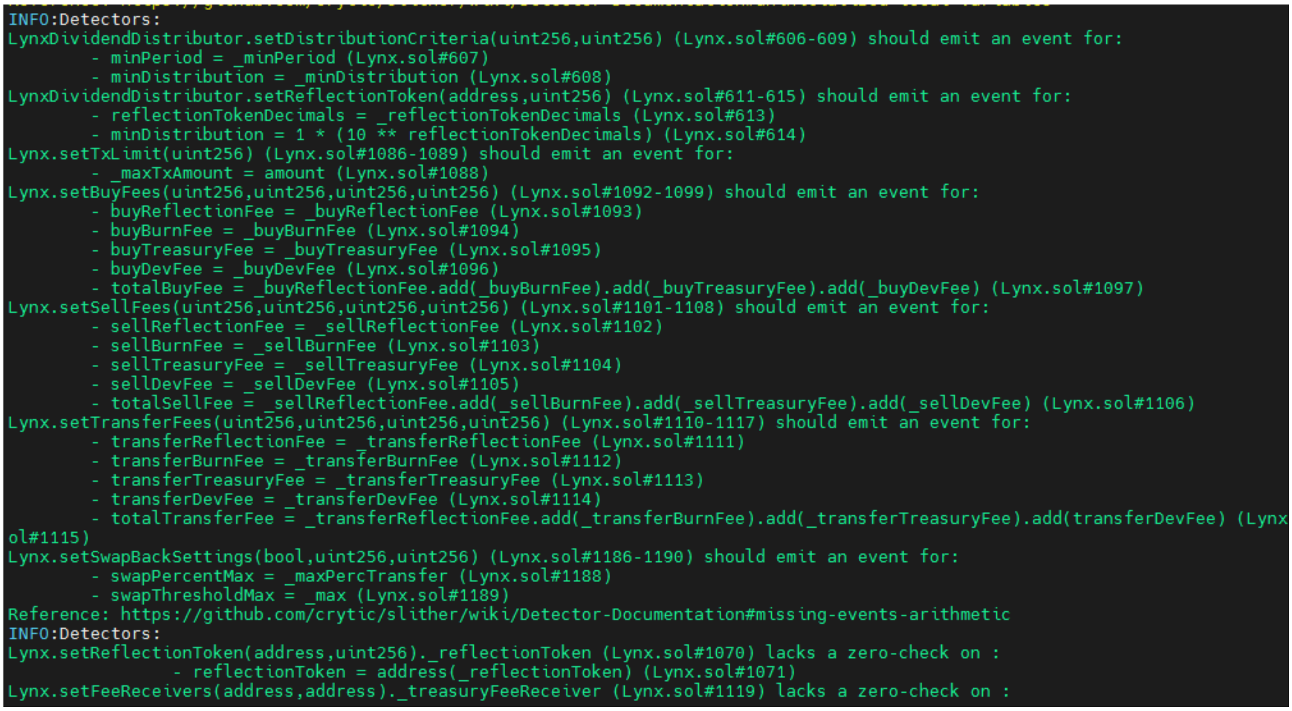
Appendix

Code Flow Diagram - Lynx Finance

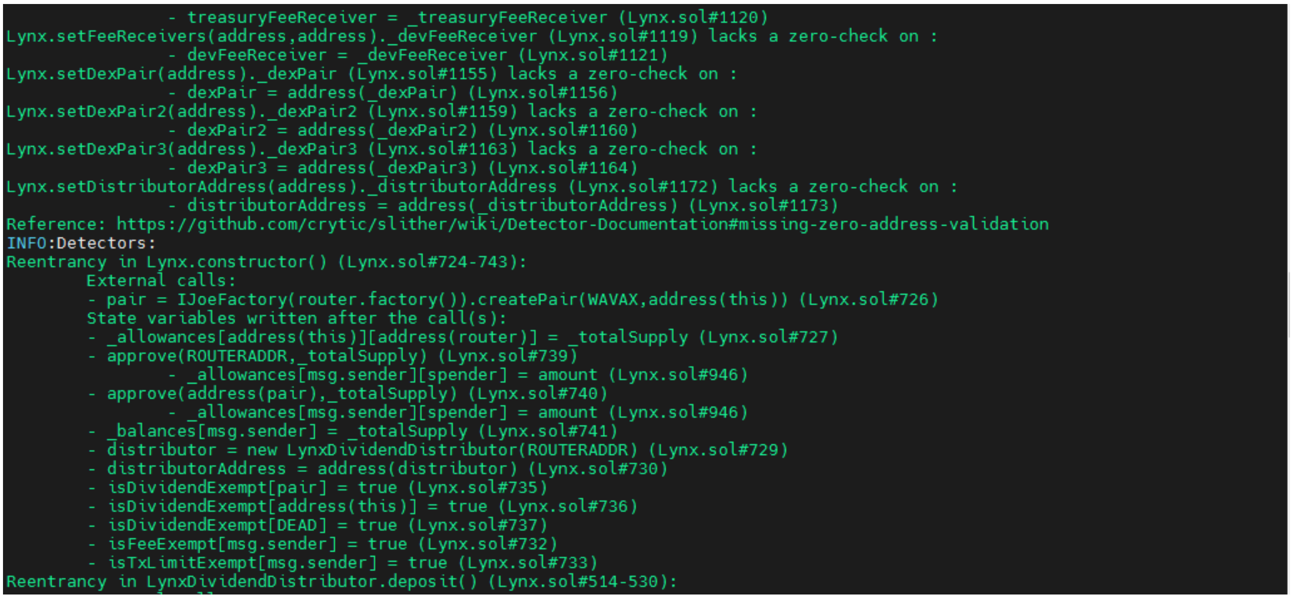


Slither Results Log

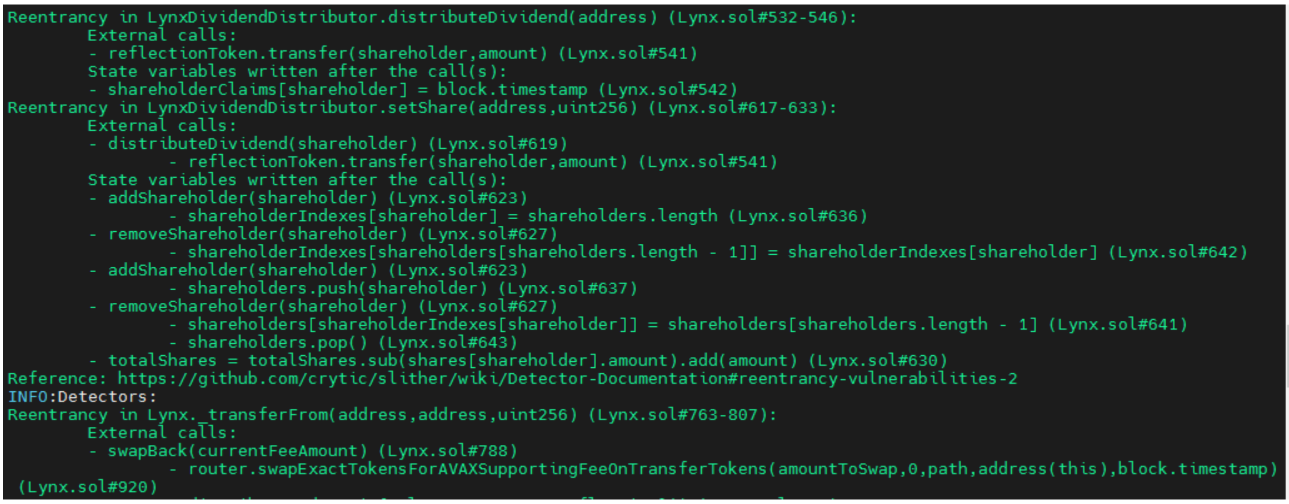
Slither Log >> Lynx.sol



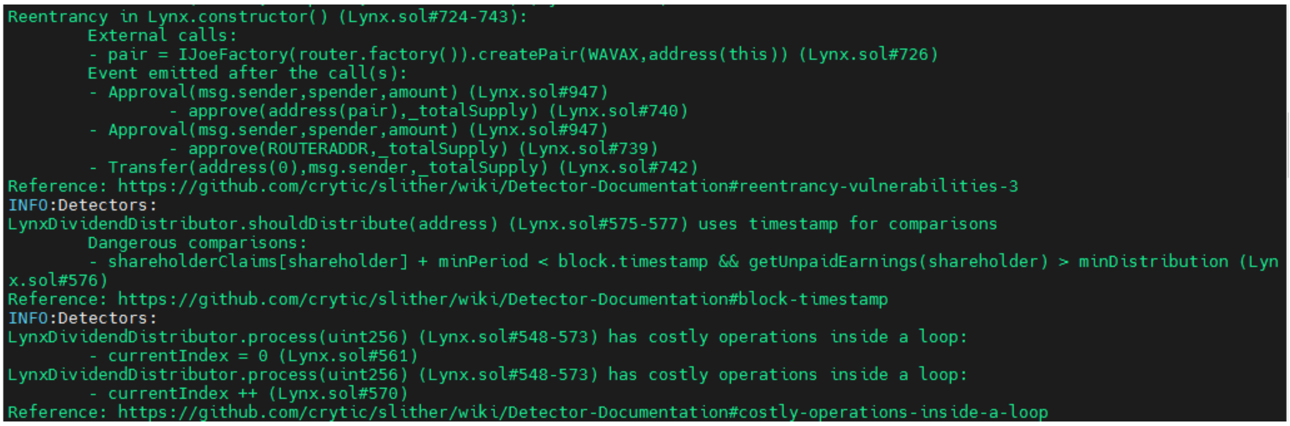
```
INFO:Detectors:
LynxDividendDistributor.setDistributionCriteria(uint256,uint256) (Lynx.sol#606-609) should emit an event for:
        - minPeriod = _minPeriod (Lynx.sol#607)
        - minDistribution = _minDistribution (Lynx.sol#608)
LynxDividendDistributor.setReflectionToken(address,uint256) (Lynx.sol#611-615) should emit an event for:
        - reflectionTokenDecimals = _reflectionTokenDecimals (Lynx.sol#613)
        - minDistribution = 1 * (10 ** reflectionTokenDecimals) (Lynx.sol#614)
Lynx.setTxLimit(uint256) (Lynx.sol#1086-1089) should emit an event for:
        - _maxTxAmount = amount (Lynx.sol#1088)
Lynx.setBuyFees(uint256,uint256,uint256,uint256) (Lynx.sol#1092-1099) should emit an event for:
        - buyReflectionFee = _buyReflectionFee (Lynx.sol#1093)
        - buyBurnFee = _buyBurnFee (Lynx.sol#1094)
        - buyTreasuryFee = _buyTreasuryFee (Lynx.sol#1095)
        - buyDevFee = _buyDevFee (Lynx.sol#1096)
        - totalBuyFee = _buyReflectionFee.add(_buyBurnFee).add(_buyTreasuryFee).add(_buyDevFee) (Lynx.sol#1097)
Lynx.setSellFees(uint256,uint256,uint256,uint256) (Lynx.sol#1101-1108) should emit an event for:
        - sellReflectionFee = _sellReflectionFee (Lynx.sol#1102)
        - sellBurnFee = _sellBurnFee (Lynx.sol#1103)
        - sellTreasuryFee = _sellTreasuryFee (Lynx.sol#1104)
        - sellDevFee = _sellDevFee (Lynx.sol#1105)
        - totalSellFee = _sellReflectionFee.add(_sellBurnFee).add(_sellTreasuryFee).add(_sellDevFee) (Lynx.sol#1106)
Lynx.setTransferFees(uint256,uint256,uint256,uint256) (Lynx.sol#1110-1117) should emit an event for:
        - transferReflectionFee = _transferReflectionFee (Lynx.sol#1111)
        - transferBurnFee = _transferBurnFee (Lynx.sol#1112)
        - transferTreasuryFee = _transferTreasuryFee (Lynx.sol#1113)
        - transferDevFee = _transferDevFee (Lynx.sol#1114)
        - totalTransferFee = _transferReflectionFee.add(_transferBurnFee).add(_transferTreasuryFee).add(transferDevFee) (Lynx
ol#1115)
Lynx.setSwapBackSettings(bool,uint256,uint256) (Lynx.sol#1186-1190) should emit an event for:
        - swapPercentMax = _maxPercTransfer (Lynx.sol#1188)
        - swapThresholdMax = _max (Lynx.sol#1189)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
Lynx.setReflectionToken(address,uint256)._reflectionToken (Lynx.sol#1070) lacks a zero-check on :
                - reflectionToken = address(_reflectionToken) (Lynx.sol#1071)
Lynx.setFeeReceivers(address,address)._treasuryFeeReceiver (Lynx.sol#1119) lacks a zero-check on :
```



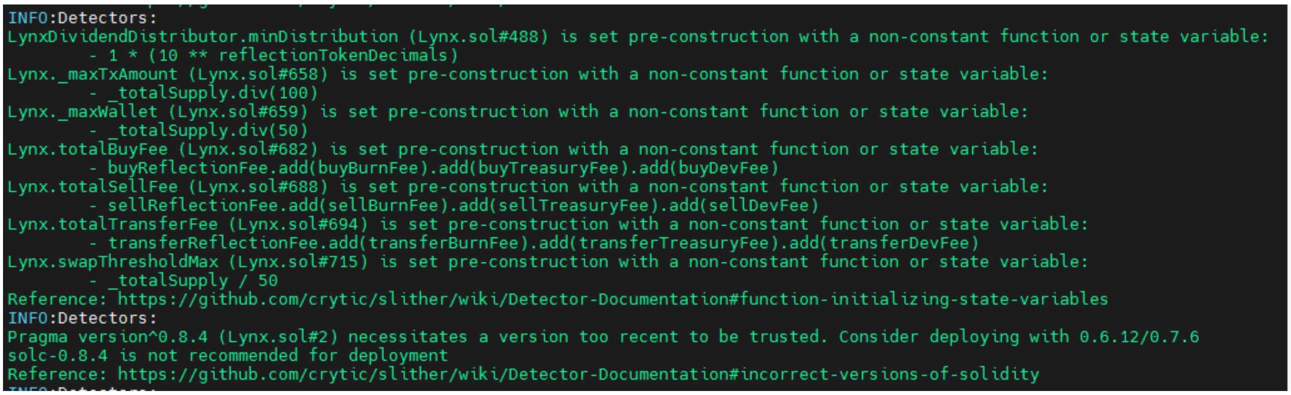
```
                - treasuryFeeReceiver = _treasuryFeeReceiver (Lynx.sol#1120)
Lynx.setFeeReceivers(address,address)._devFeeReceiver (Lynx.sol#1119) lacks a zero-check on :
                - devFeeReceiver = _devFeeReceiver (Lynx.sol#1121)
Lynx.setDexPair(address)._dexPair (Lynx.sol#1155) lacks a zero-check on :
                - dexPair = address(_dexPair) (Lynx.sol#1156)
Lynx.setDexPair2(address)._dexPair2 (Lynx.sol#1159) lacks a zero-check on :
                - dexPair2 = address(_dexPair2) (Lynx.sol#1160)
Lynx.setDexPair3(address)._dexPair3 (Lynx.sol#1163) lacks a zero-check on :
                - dexPair3 = address(_dexPair3) (Lynx.sol#1164)
Lynx.setDistributorAddress(address)._distributorAddress (Lynx.sol#1172) lacks a zero-check on :
                - distributorAddress = address(_distributorAddress) (Lynx.sol#1173)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Lynx.constructor() (Lynx.sol#724-743):
        External calls:
        - pair = IJoeFactory(router.factory()).createPair(WAVAX,address(this)) (Lynx.sol#726)
        State variables written after the call(s):
        - _allowances[address(this)][address(router)] = _totalSupply (Lynx.sol#727)
        - approve(ROUTERADDR,_totalSupply) (Lynx.sol#739)
                - _allowances[msg.sender][spender] = amount (Lynx.sol#946)
        - approve(address(pair),_totalSupply) (Lynx.sol#740)
                - _allowances[msg.sender][spender] = amount (Lynx.sol#946)
        - _balances[msg.sender] = _totalSupply (Lynx.sol#741)
        - distributor = new LynxDividendDistributor(ROUTERADDR) (Lynx.sol#729)
        - distributorAddress = address(distributor) (Lynx.sol#730)
        - isDividendExempt[pair] = true (Lynx.sol#735)
        - isDividendExempt[address(this)] = true (Lynx.sol#736)
        - isDividendExempt[DEAD] = true (Lynx.sol#737)
        - isFeeExempt[msg.sender] = true (Lynx.sol#732)
        - isTxLimitExempt[msg.sender] = true (Lynx.sol#733)
Reentrancy in LynxDividendDistributor.deposit() (Lynx.sol#514-530):
```



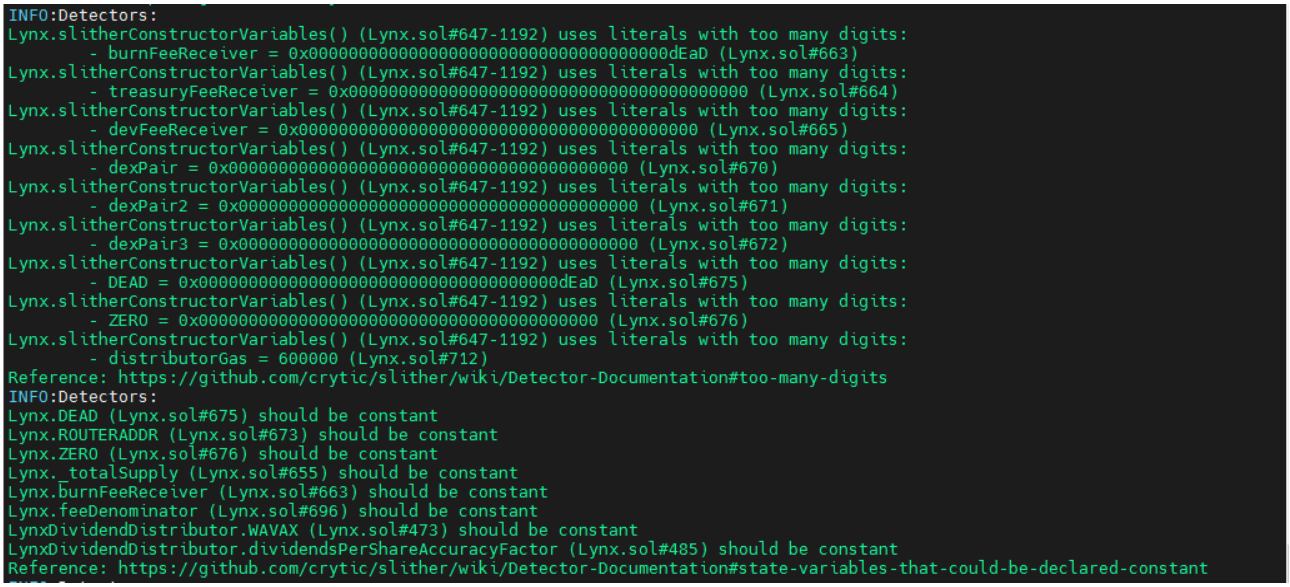
```
Reentrancy in LynxDividendDistributor.distributeDividend(address) (Lynx.sol#532-546):
        External calls:
        - reflectionToken.transfer(shareholder,amount) (Lynx.sol#541)
        State variables written after the call(s):
        - shareholderClaims[shareholder] = block.timestamp (Lynx.sol#542)
Reentrancy in LynxDividendDistributor.setShare(address,uint256) (Lynx.sol#617-633):
        External calls:
        - distributeDividend(shareholder) (Lynx.sol#619)
                - reflectionToken.transfer(shareholder,amount) (Lynx.sol#541)
        State variables written after the call(s):
        - addShareholder(shareholder) (Lynx.sol#623)
                - shareholderIndexes[shareholder] = shareholders.length (Lynx.sol#636)
        - removeShareholder(shareholder) (Lynx.sol#627)
                - shareholderIndexes[shareholders[shareholders.length - 1]] = shareholderIndexes[shareholder] (Lynx.sol#642)
        - addShareholder(shareholder) (Lynx.sol#623)
                - shareholders.push(shareholder) (Lynx.sol#637)
        - removeShareholder(shareholder) (Lynx.sol#627)
                - shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length - 1] (Lynx.sol#641)
                - shareholders.pop() (Lynx.sol#643)
        - totalShares = totalShares.sub(shares[shareholder].amount).add(amount) (Lynx.sol#630)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in Lynx._transferFrom(address,address,uint256) (Lynx.sol#763-807):
        External calls:
        - swapBack(currentFeeAmount) (Lynx.sol#788)
                - router.swapExactTokensForAVAXSupportingFeeOnTransferTokens(amountToSwap,0,path,address(this),block.timestamp)
 (Lynx.sol#920)
```



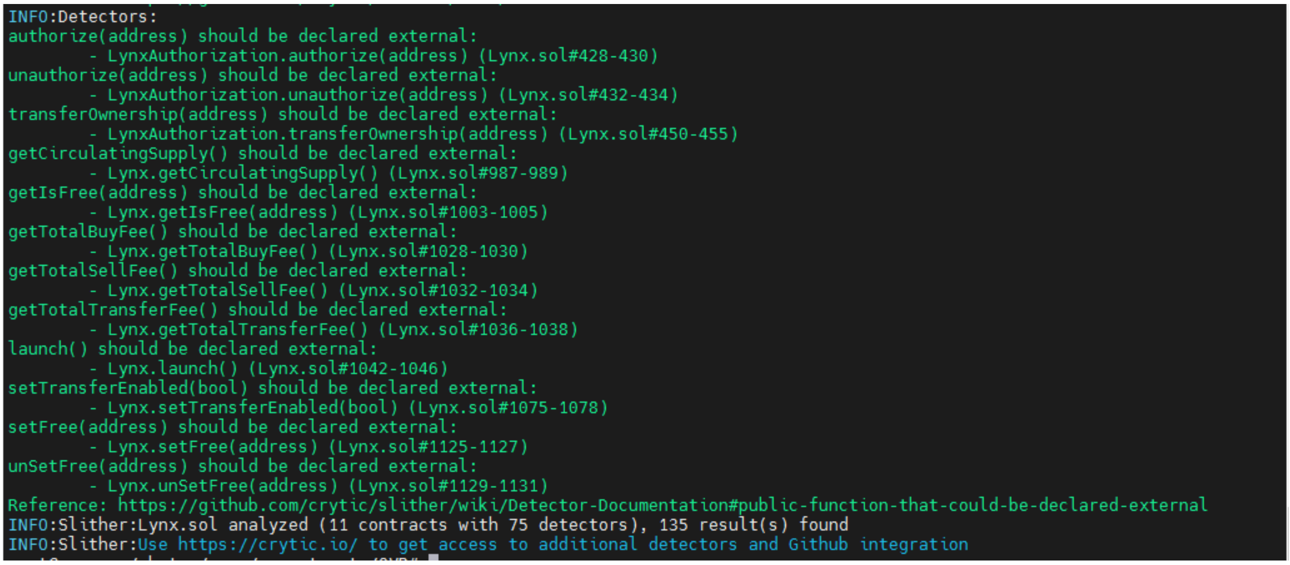
```
Reentrancy in Lynx.constructor() (Lynx.sol#724-743):
        External calls:
        - pair = IJoeFactory(router.factory()).createPair(WAVAX,address(this)) (Lynx.sol#726)
        Event emitted after the call(s):
        - Approval(msg.sender,spender,amount) (Lynx.sol#947)
                - approve(address(pair),_totalSupply) (Lynx.sol#740)
        - Approval(msg.sender,spender,amount) (Lynx.sol#947)
                - approve(ROUTERADDR,_totalSupply) (Lynx.sol#739)
        - Transfer(address(0),msg.sender,_totalSupply) (Lynx.sol#742)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
LynxDividendDistributor.shouldDistribute(address) (Lynx.sol#575-577) uses timestamp for comparisons
        Dangerous comparisons:
        - shareholderClaims[shareholder] + minPeriod < block.timestamp && getUnpaidEarnings(shareholder) > minDistribution (Lyn
x.sol#576)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
LynxDividendDistributor.process(uint256) (Lynx.sol#548-573) has costly operations inside a loop:
        - currentIndex = 0 (Lynx.sol#561)
LynxDividendDistributor.process(uint256) (Lynx.sol#548-573) has costly operations inside a loop:
        - currentIndex ++ (Lynx.sol#570)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
```



```
INFO:Detectors:
LynxDividendDistributor.minDistribution (Lynx.sol#488) is set pre-construction with a non-constant function or state variable:
        - 1 * (10 ** reflectionTokenDecimals)
Lynx._maxTxAmount (Lynx.sol#658) is set pre-construction with a non-constant function or state variable:
        - _totalSupply.div(100)
Lynx._maxWallet (Lynx.sol#659) is set pre-construction with a non-constant function or state variable:
        - _totalSupply.div(50)
Lynx.totalBuyFee (Lynx.sol#682) is set pre-construction with a non-constant function or state variable:
        - buyReflectionFee.add(buyBurnFee).add(buyTreasuryFee).add(buyDevFee)
Lynx.totalSellFee (Lynx.sol#688) is set pre-construction with a non-constant function or state variable:
        - sellReflectionFee.add(sellBurnFee).add(sellTreasuryFee).add(sellDevFee)
Lynx.totalTransferFee (Lynx.sol#694) is set pre-construction with a non-constant function or state variable:
        - transferReflectionFee.add(transferBurnFee).add(transferTreasuryFee).add(transferDevFee)
Lynx.swapThresholdMax (Lynx.sol#715) is set pre-construction with a non-constant function or state variable:
        - _totalSupply / 50
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state-variables
INFO:Detectors:
Pragma version^0.8.4 (Lynx.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```



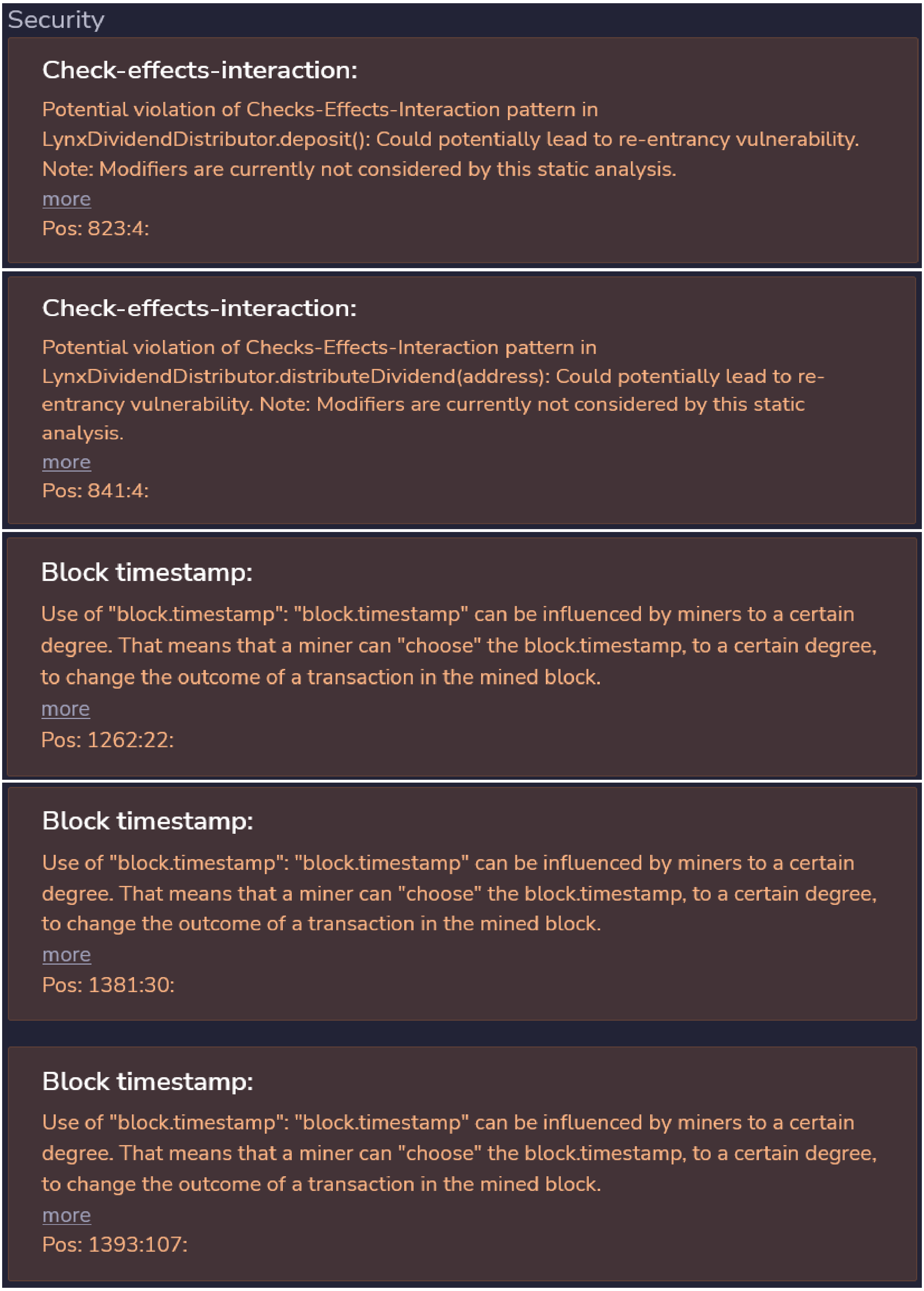
```
INFO:Detectors:
Lynx.slitherConstructorVariables() (Lynx.sol#647-1192) uses literals with too many digits:
        - burnFeeReceiver = 0x000000000000000000000000000000000000dEaD (Lynx.sol#663)
Lynx.slitherConstructorVariables() (Lynx.sol#647-1192) uses literals with too many digits:
        - treasuryFeeReceiver = 0x0000000000000000000000000000000000000000 (Lynx.sol#664)
Lynx.slitherConstructorVariables() (Lynx.sol#647-1192) uses literals with too many digits:
        - devFeeReceiver = 0x0000000000000000000000000000000000000000 (Lynx.sol#665)
Lynx.slitherConstructorVariables() (Lynx.sol#647-1192) uses literals with too many digits:
        - dexPair = 0x0000000000000000000000000000000000000000 (Lynx.sol#670)
Lynx.slitherConstructorVariables() (Lynx.sol#647-1192) uses literals with too many digits:
        - dexPair2 = 0x0000000000000000000000000000000000000000 (Lynx.sol#671)
Lynx.slitherConstructorVariables() (Lynx.sol#647-1192) uses literals with too many digits:
        - dexPair3 = 0x0000000000000000000000000000000000000000 (Lynx.sol#672)
Lynx.slitherConstructorVariables() (Lynx.sol#647-1192) uses literals with too many digits:
        - DEAD = 0x000000000000000000000000000000000000dEaD (Lynx.sol#675)
Lynx.slitherConstructorVariables() (Lynx.sol#647-1192) uses literals with too many digits:
        - ZERO = 0x0000000000000000000000000000000000000000 (Lynx.sol#676)
Lynx.slitherConstructorVariables() (Lynx.sol#647-1192) uses literals with too many digits:
        - distributorGas = 600000 (Lynx.sol#712)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
Lynx.DEAD (Lynx.sol#675) should be constant
Lynx.ROUTERADDR (Lynx.sol#673) should be constant
Lynx.ZERO (Lynx.sol#676) should be constant
Lynx._totalSupply (Lynx.sol#655) should be constant
Lynx.burnFeeReceiver (Lynx.sol#663) should be constant
Lynx.feeDenominator (Lynx.sol#696) should be constant
LynxDividendDistributor.WAVAX (Lynx.sol#473) should be constant
LynxDividendDistributor.dividendsPerShareAccuracyFactor (Lynx.sol#485) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```



```
INFO:Detectors:
authorize(address) should be declared external:
        - LynxAuthorization.authorize(address) (Lynx.sol#428-430)
unauthorize(address) should be declared external:
        - LynxAuthorization.unauthorize(address) (Lynx.sol#432-434)
transferOwnership(address) should be declared external:
        - LynxAuthorization.transferOwnership(address) (Lynx.sol#450-455)
getCirculatingSupply() should be declared external:
        - Lynx.getCirculatingSupply() (Lynx.sol#987-989)
getIsFree(address) should be declared external:
        - Lynx.getIsFree(address) (Lynx.sol#1003-1005)
getTotalBuyFee() should be declared external:
        - Lynx.getTotalBuyFee() (Lynx.sol#1028-1030)
getTotalSellFee() should be declared external:
        - Lynx.getTotalSellFee() (Lynx.sol#1032-1034)
getTotalTransferFee() should be declared external:
        - Lynx.getTotalTransferFee() (Lynx.sol#1036-1038)
launch() should be declared external:
        - Lynx.launch() (Lynx.sol#1042-1046)
setTransferEnabled(bool) should be declared external:
        - Lynx.setTransferEnabled(bool) (Lynx.sol#1075-1078)
setFree(address) should be declared external:
        - Lynx.setFree(address) (Lynx.sol#1125-1127)
unSetFree(address) should be declared external:
        - Lynx.unSetFree(address) (Lynx.sol#1129-1131)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Lynx.sol analyzed (11 contracts with 75 detectors), 135 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

Lynx.sol



Security

Check-effects-interaction:

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

more

Pos: 823:4:

Check-effects-interaction:

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

more

Pos: 841:4:

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: 1262:22:

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

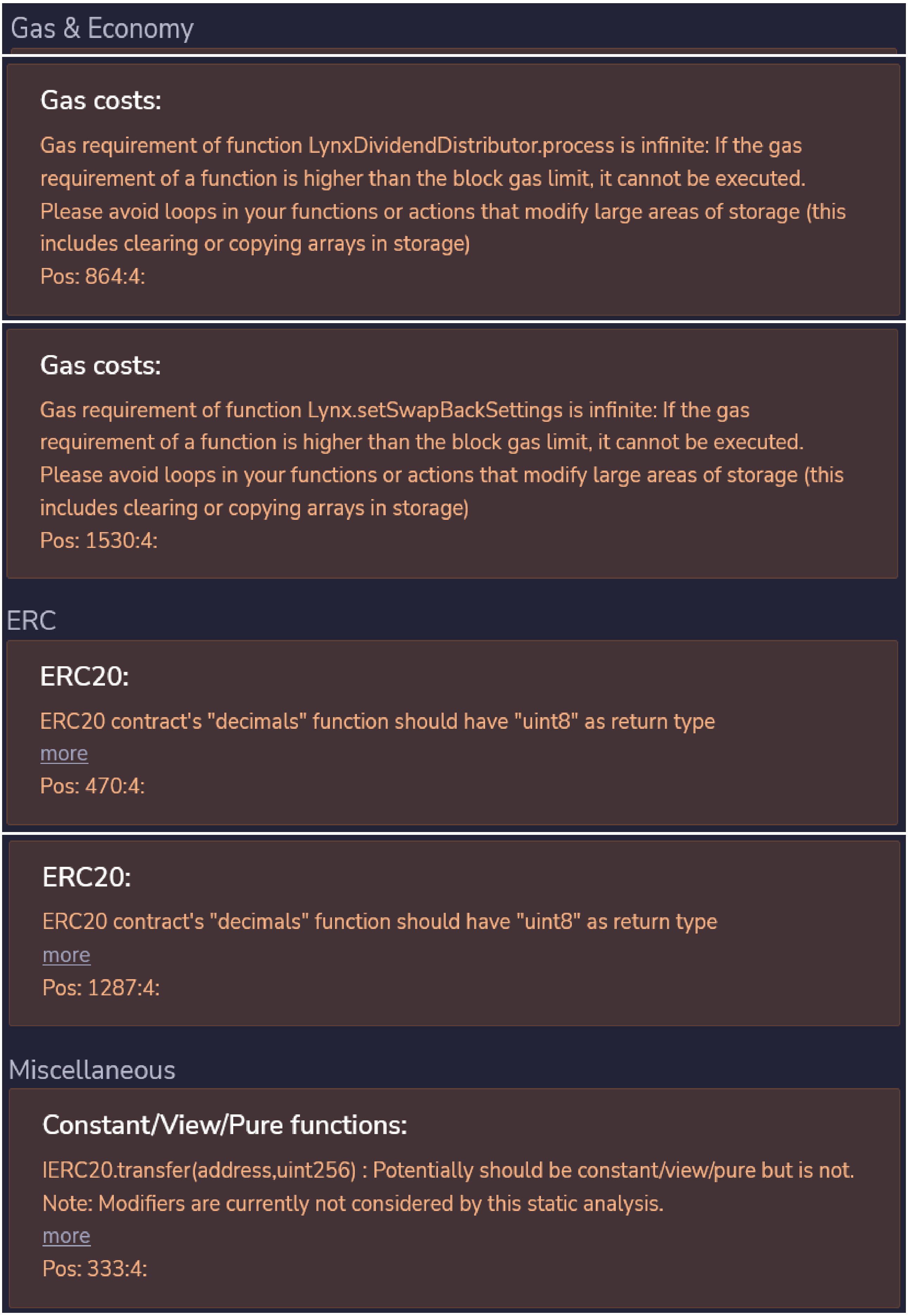
Pos: 1381:30:

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: 1393:107:



Gas & Economy

Gas costs:

Gas requirement of function LynxDividendDistributor.process 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: 864:4:

Gas costs:

Gas requirement of function Lynx.setSwapBackSettings 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: 1530:4:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 470:4:

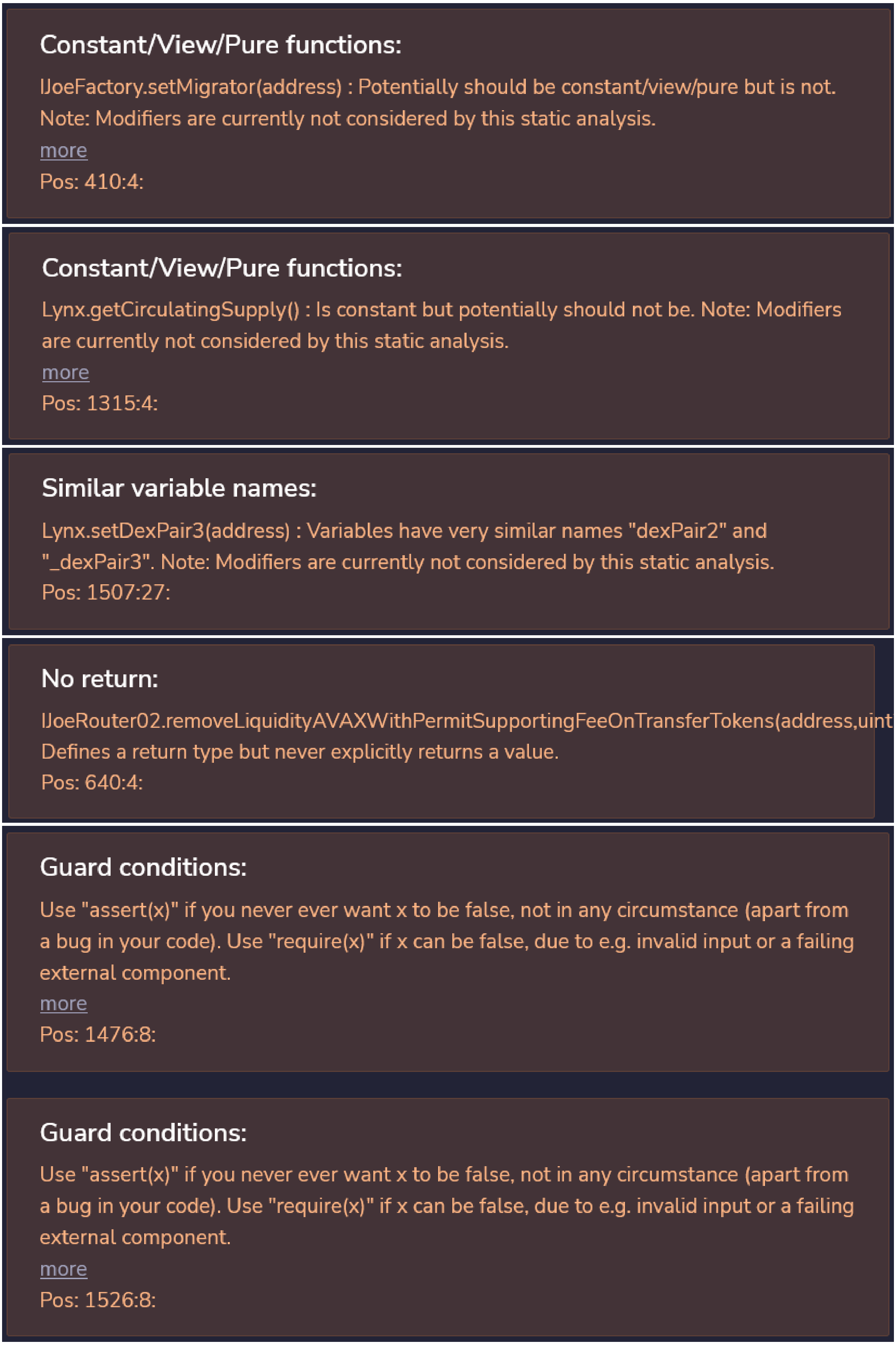
ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 1287:4:

Miscellaneous

Constant/View/Pure functions:

IERC20.transfer(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 333:4:



Constant/View/Pure functions:

IJoeFactory.setMigrator(address) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 410:4:

Constant/View/Pure functions:

Lynx.getCirculatingSupply() : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1315:4:

Similar variable names:

Lynx.setDexPair3(address) : Variables have very similar names "dexPair2" and "_dexPair3". Note: Modifiers are currently not considered by this static analysis.

Pos: 1507:27:

No return:

IJoeRouter02.removeLiquidityAVAXWithPermitSupportingFeeOnTransferTokens(address,uint

Defines a return type but never explicitly returns a value.

Pos: 640:4:

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

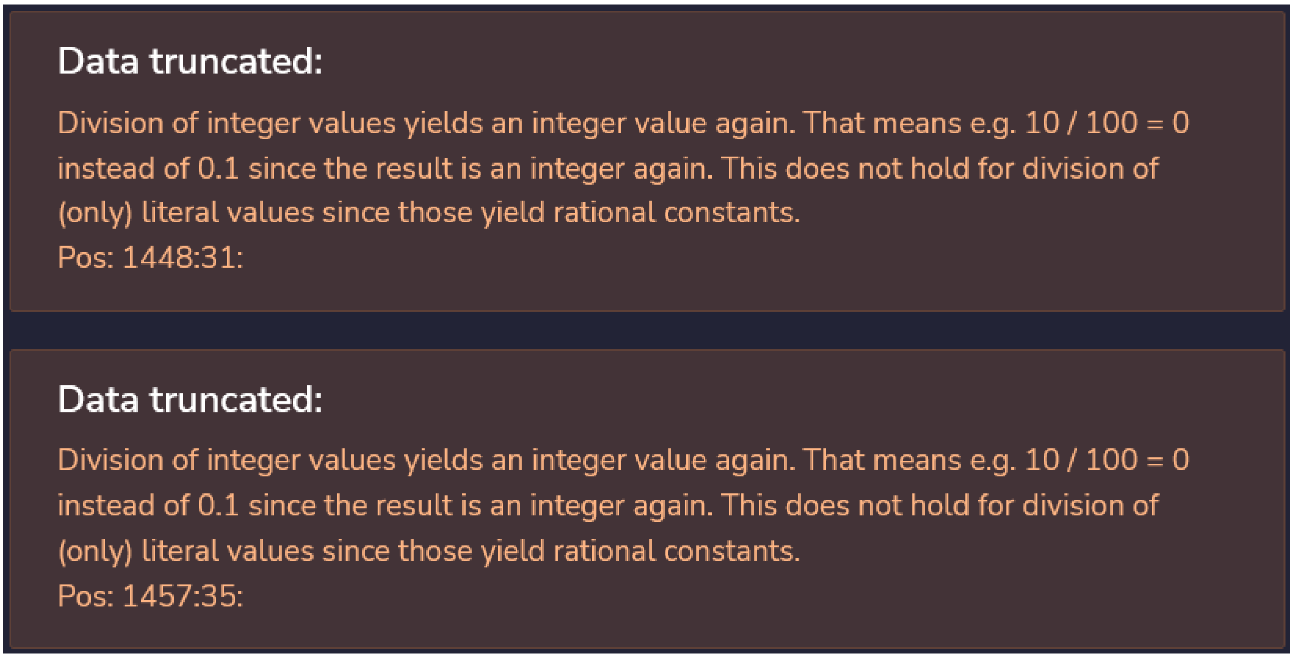
Pos: 1476:8:

Guard conditions:

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

more

Pos: 1526:8:



Data truncated:

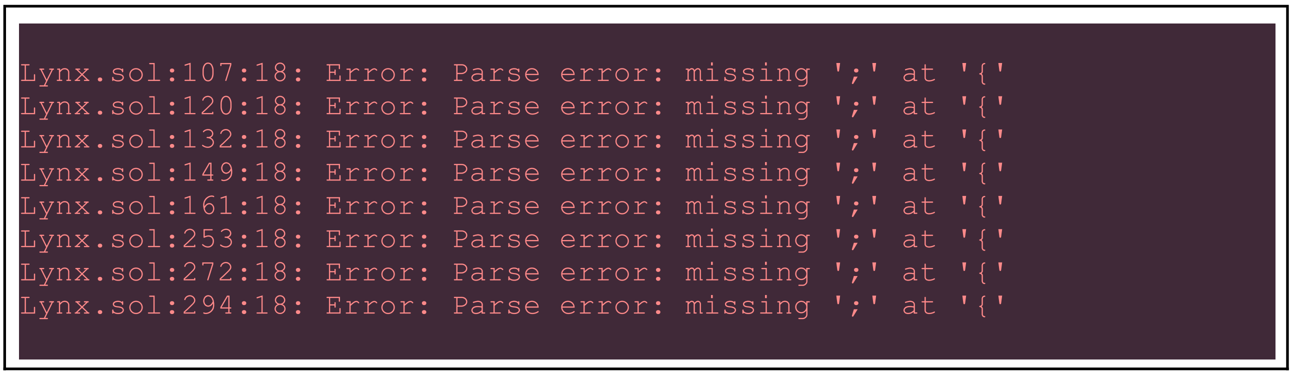
Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 1448:31:

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: 1457:35:

Solhint Linter

Lynx.sol



```
Lynx.sol:107:18: Error: Parse error: missing ';' at '{'
Lynx.sol:120:18: Error: Parse error: missing ';' at '{'
Lynx.sol:132:18: Error: Parse error: missing ';' at '{'
Lynx.sol:149:18: Error: Parse error: missing ';' at '{'
Lynx.sol:161:18: Error: Parse error: missing ';' at '{'
Lynx.sol:253:18: Error: Parse error: missing ';' at '{'
Lynx.sol:272:18: Error: Parse error: missing ';' at '{'
Lynx.sol:294:18: Error: Parse error: missing ';' at '{'
```

Software analysis result:

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

EtherAuthority

This is a private and confidential document. No part of this document should
be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io