Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	20Lab
Platform:	Binance Smart Chain
Language:	Solidity
Date:	May 27th, 2024

Table of contents

`

Introduction

EtherAuthority was contracted by the 20Lab team to perform the Security audit of the 20Lab smart contracts 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 May 27th, 2024.

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 20Lab token contract defines a comprehensive ERC20 token with additional functionalities such as burning, ownership control, fee management, dividend tracking, and liquidity management. Here’s an in-depth breakdown of the contract's key components and functionalities:
- Here's a brief overview of the key components and functionalities of the provided code:
    - **Token.sol(20lab-v1.9.0-1):** This contract is designed for a comprehensive token ecosystem with mechanisms for fee management, liquidity provision, and dividend distribution, making it suitable for use cases that require complex tokenomics and user interactions.
    - **Token.sol(20lab-v1.9.0-2):** This contract is designed for a comprehensive token management system with advanced features like dynamic fee management, liquidity provision, and dividend distribution. It includes robust mechanisms for handling various operational aspects, ensuring security and flexibility for the token holders and the contract owner.

Audit scope

Name	**Code Review and Security Analysis Report for 20Lab Smart Contracts**
Platform	**Binance Smart Chain**
Language	**Solidity**
File 1	Token.sol(20lab-v1.9.0-1)
File 1 Smart Contract Code	0xbf752138328562c717f840468014500b6Ebf7500
File 2	Token.sol(20lab-v1.9.0-2)
File 2 Smart Contract Code	0xf43f1B7c53b35297201Cf779c606456966f9D070
Audit Date	May 27th, 2024

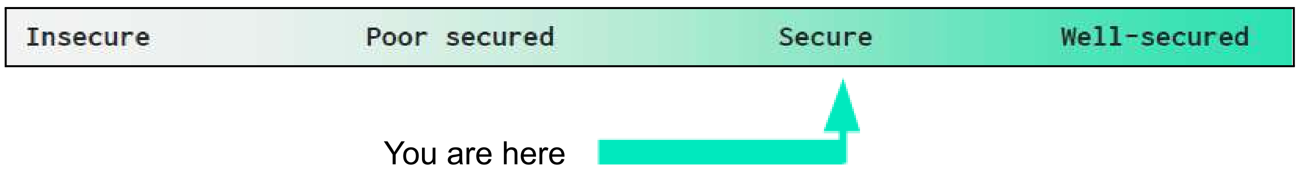
Claimed Smart Contract Features

Claimed Feature Details	Our Observation
File 1: Token.sol(20lab-v1.9.0-1) **Tokenomics:** • Name: 20lab-v1.9.0-1 • Symbol: 20lab-v1.9.0-1 • Decimals: 18 • Total Supply: 1 million 20lab-v1.9.0-1 • Gas For Processing: 0.3 million 20lab-v1.9.0-1 • Maximum buy amount: 0.05 million 20lab-v1.9.0-1 • Maximum sell amount: 0.05 million 20lab-v1.9.0-1 • Maximum Supply: 1.23 million 20lab-v1.9.0-1 • Maximum wallet amount: 0.1 million 20lab-v1.9.0-1 • Swap Threshold Ratio: 0.5 % of the balance of pairV2 contract **Ownership Control:** • Recover tokens. • Recover foreign erc20 tokens. • Add addresses in the blacklist. • Update the swap threshold. • Update the marketing address. • Update marketing fees like buy fee, sell fee, and transfer fee. • Update liquidity fees like buy fee, sell fee, and transfer fee. • Manage exclusion from dividends. • Enable trading. • Manage exclusion from trading restrictions. • Set up the rewards fees. • Manage exclusion from fees. • Set an AMM pair address.	**YES, This is valid. We suggest renouncing ownership once the ownership functions are not needed. This is to make the smart contract 100% decentralized.**

● Manage exclusion from transaction limits. ● Update the maximum wallet amount. ● Update the maximum Buy/Sell amount	
File 2: Token.sol(20lab-v1.9.0-2) **Tokenomics:** ● Name: 20lab-v1.9.0-2 ● Symbol: 20lab-v1.9.0-2 ● Decimals: 18 ● Total Supply: 1 million 20lab-v1.9.0-2 ● Gas For Processing: 0.3 million 20lab-v1.9.0-2 ● Maximum buy amount: 0.05 million 20lab-v1.9.0-2 ● Maximum sell amount: 0.05 million 20lab-v1.9.0-2 ● Maximum wallet amount: 0.1 million 20lab-v1.9.0-2 ● Swap Threshold Ratio: 0.5 % of balance of pairV2 contract **Ownership Control:** ● Recover tokens. ● Recover foreign erc20 tokens. ● Update the swap threshold. ● Update the marketing address. ● Update marketing fees like buy fee, sell fee, and transfer fee. ● Update liquidity fees like buy fee, sell fee, and transfer fee. ● Manage exclusion from dividends. ● Set up the rewards fees. ● Manage exclusion from fees. ● Set an AMM pair address. ● Manage exclusion from transaction limits. ● Update the maximum wallet amount. ● Update the maximum Buy/Sell amount	**YES, This is valid. We suggest renouncing ownership once the ownership functions are not needed. This is to make the smart contract 100% decentralized.**

Audit Summary

According to the standard audit assessment, the 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 a 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. A general overview is presented in the AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium, 0 low, and 2 very low-level issues.

Investor Advice: A 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	The solidity version is not specified	Passed
	Solidity version is too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack 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	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	"Out of Gas" Issue	Passed
	High consumption 'for/while' loop	Passed
	High consumption 'storage' storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage is not set	Passed
	"Short Address" Attack	Passed
	"Double Spend" Attack	Passed

Overall Audit Result: **PASSED**

Business Risk Analysis

Category	Result
● Buy Tax	2%
● Sell Tax	2%
● Transfer Tax	0%
● Cannot Buy	No
● Cannot Sell	No
● Max Tax	25%
● Modify Tax	Yes
● Fee Check	Not Detected
● Is Honeypot	Not Detected
● Trading Cooldown	Not Detected
● Can Pause Trade?	Not Detected
● Pause Transfer?	Not Detected
● Max Tax?	No
● Is it Anti-whale?	Not Detected
● Is Anti-bot?	Not Detected
● Is it a Blacklist?	No
● Blacklist Check	No
● Can Mint?	No
● Is it a Proxy?	No
● Can Take Ownership?	Yes
● Hidden Owner?	Not Detected
● Self Destruction?	Not Detected
● Auditor Confidence	High

Overall Audit Result: PASSED

Code Quality

This audit scope has 2 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits, and Interfaces. This is a compact and well-written smart contract.

The libraries in 20Lab 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 20Lab.

The 20Lab 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 **well** commented on in the smart contracts. Ethereum's NatSpec commenting style is recommended.

Documentation

We were given a 20Lab smart contract code in the form of a 20lab-v1.9.0-1 and 20lab-v1.9.0-2 web link.

As mentioned above, code parts are **well-commented**. And the logic is straightforward. So it is easy to quickly understand the programming flow as well as the complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Use of Dependencies

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

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

AS-IS overview

Token.sol(20lab-v1.9.0-1)

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	afterConstructor	external	initializer	No Issue
3	decimals	write	Passed	No Issue
4	recoverToken	external	access only Owner	No Issue
5	recoverForeignERC20	external	access only Owner	No Issue
6	blacklist	external	access only Owner	No Issue
7	_updateFeeToken	write	Passed	No Issue
8	_sendInOtherTokens	write	Passed	No Issue
9	_swapTokensForOtherTokens	write	Passed	No Issue
10	updateSwapThreshold	write	access only Owner	No Issue
13	getSwapThresholdAmount	read	Passed	No Issue
14	getAllPending	read	Passed	No Issue
15	marketingAddressSetup	write	access only Owner	No Issue
16	marketingFeesSetup	write	access only Owner	No Issue
17	receive	external	Passed	No Issue
18	_swapTokensForCoin	write	Passed	No Issue
19	_swapAndLiquify	write	Passed	No Issue
20	_addLiquidity	write	Passed	No Issue
21	addLiquidityFromLeftoverTokens	external	Passed	No Issue
22	liquidityFeesSetup	write	access only Owner	No Issue
23	_sendDividends	write	Passed	No Issue
24	excludeFromDividends	external	access only Owner	No Issue
25	_excludeFromDividends	internal	Passed	No Issue
26	rewardsFeesSetup	write	access only Owner	No Issue
27	excludeFromFees	write	access only Owner	No Issue
28	_updateRouterV2	write	Passed	No Issue
29	setAMMPair	external	access only Owner	No Issue
30	_setAMMPair	write	Passed	No Issue
31	excludeFromLimits	external	access only Owner	No Issue
32	_excludeFromLimits	internal	Passed	No Issue
33	updateMaxWalletAmount	write	access only Owner	No Issue
34	_maxWalletSafeLimit	read	Passed	No Issue
35	_maxTxSafeLimit	read	Passed	No Issue
36	updateMaxBuyAmount	write	access only Owner	No Issue
37	updateMaxSellAmount	write	access only Owner	No Issue
38	enableTrading	external	access only Owner	No Issue
39	excludeFromTradingRestriction	write	access only Owner	No Issue
40	_update	internal	Passed	No Issue
41	_beforeTokenUpdate	internal	Warning: Function state mutability can	Refer Audit Findings

			be restricted to view	
42	_afterTokenUpdate	internal	Warning: Unused function parameter. Remove or comment out the variable name to silence this warning	Refer Audit Findings
43	name	read	Passed	No Issue
44	symbol	read	Passed	No Issue
45	decimals	read	Passed	No Issue
46	totalSupply	read	Passed	No Issue
47	balanceOf	read	Passed	No Issue
48	transfer	write	Passed	No Issue
49	allowance	read	Passed	No Issue
50	approve	write	Passed	No Issue
51	transferFrom	write	Passed	No Issue
52	_transfer	internal	Passed	No Issue
53	_update	internal	Passed	No Issue
54	_mint	internal	Passed	No Issue
55	_burn	internal	Passed	No Issue
56	_approve	internal	Passed	No Issue
57	_approve	internal	Passed	No Issue
58	_spendAllowance	internal	Passed	No Issue
59	burn	write	Passed	No Issue
60	burnFrom	write	Passed	No Issue
61	pendingOwner	read	Passed	No Issue
62	transferOwnership	write	access only Owner	No Issue
63	_transferOwnership	internal	Passed	No Issue
64	acceptOwnership	write	Passed	No Issue
65	mint	write	access only Owner	No Issue
66	permit	write	Passed	No Issue
67	nonces	read	Passed	No Issue
68	DOMAIN_SEPARATOR	external	Passed	No Issue
69	_deployDividendTracker	internal	Passed	No Issue
70	gasForProcessingSetup	write	access only Owner	No Issue
71	callbackGasSetup	external	access only Owner	No Issue
72	claimWaitSetup	external	access only Owner	No Issue
73	_excludeFromDividends	internal	Passed	No Issue
74	isExcludedFromDividends	read	Passed	No Issue
75	claim	external	Passed	No Issue
76	getClaimWait	external	Passed	No Issue
77	getTotalDividendsDistributed	external	Passed	No Issue
78	withdrawableDividendOf	read	Passed	No Issue
79	dividendTokenBalanceOf	read	Passed	No Issue
80	dividendTokenTotalSupply	read	Passed	No Issue
81	getAccountDividendsInfo	external	Passed	No Issue
82	getAccountDividendsInfoAtIndex	external	Passed	No Issue

83	getLastProcessedIndex	external	Passed	No Issue
84	getNumberOfDividendTokenHolders	read	Passed	No Issue
85	process	external	Passed	No Issue
86	initializer	modifier	Passed	No Issue

Token.sol(20lab-v1.9.0-2)

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	afterConstructor	external	initializer	No Issue
3	decimals	write	Passed	No Issue
4	recoverToken	external	access only Owner	No Issue
5	recoverForeignERC20	external	access only Owner	No Issue
6	receive	external	Passed	No Issue
7	_swapTokensForCoin	write	Passed	No Issue
8	updateSwapThreshold	write	access only Owner	No Issue
9	getSwapThresholdAmount	read	Passed	No Issue
10	getAllPending	read	Passed	No Issue
13	marketingAddressSetup	write	access only Owner	No Issue
14	marketingFeesSetup	write	access only Owner	No Issue
15	_swapAndLiquify	write	Passed	No Issue
16	_addLiquidity	write	Passed	No Issue
17	addLiquidityFromLeftoverTokens	external	Passed	No Issue
18	liquidityFeesSetup	write	access only Owner	No Issue
19	_swapTokensForOtherRewardTokens	write	Passed	No Issue
20	_sendDividends	write	Passed	No Issue
21	excludeFromDividends	external	access only Owner	No Issue
22	_excludeFromDividends	internal	Passed	No Issue
23	rewardsFeesSetup	write	access only Owner	No Issue
24	excludeFromFees	write	access only Owner	No Issue
25	_updateRouterV2	write	Passed	No Issue
26	setAMMPair	external	access only Owner	No Issue
27	_setAMMPair	write	Passed	No Issue

28	excludeFromLimits	external	access only Owner	No Issue
29	_excludeFromLimits	internal	Passed	No Issue
30	updateMaxWalletAmount	write	access only Owner	No Issue
31	_maxWalletSafeLimit	read	Passed	No Issue
32	_maxTxSafeLimit	read	Passed	No Issue
33	updateMaxBuyAmount	write	access only Owner	No Issue
34	updateMaxSellAmount	write	access only Owner	No Issue
35	_update	internal	Passed	No Issue
36	_beforeTokenUpdate	internal	Warning: Function state mutability can be restricted to view	Refer Audit Findings
37	_afterTokenUpdate	internal	Warning: Unused function parameter. Remove or comment out the variable name to silence this warning	Refer Audit Findings
38	name	read	Passed	No Issue
39	symbol	read	Passed	No Issue
40	decimals	read	Passed	No Issue
41	totalSupply	read	Passed	No Issue
42	balanceOf	read	Passed	No Issue
43	transfer	write	Passed	No Issue
44	allowance	read	Passed	No Issue
45	approve	write	Passed	No Issue
46	transferFrom	write	Passed	No Issue
47	_transfer	internal	Passed	No Issue
48	_update	internal	Passed	No Issue
49	_mint	internal	Passed	No Issue
50	_burn	internal	Passed	No Issue
51	_approve	internal	Passed	No Issue
52	_approve	internal	Passed	No Issue
53	_spendAllowance	internal	Passed	No Issue
54	burn	write	Passed	No Issue
55	burnFrom	write	Passed	No Issue
56	pendingOwner	read	Passed	No Issue
57	transferOwnership	write	access only Owner	No Issue
58	_transferOwnership	internal	Passed	No Issue
59	acceptOwnership	write	Passed	No Issue
60	_deployDividendTracker	internal	Passed	No Issue

61	_setRewardToken	internal	Passed	No Issue
62	gasForProcessingSetup	write	access only Owner	No Issue
63	claimWaitSetup	external	access only Owner	No Issue
64	_excludeFromDividends	internal	Passed	No Issue
65	isExcludedFromDividends	read	Passed	No Issue
66	claim	external	Passed	No Issue
67	getClaimWait	external	Passed	No Issue
68	getTotalDividendsDistributed	external	Passed	No Issue
69	withdrawableDividendOf	read	Passed	No Issue
70	dividendTokenBalanceOf	read	Passed	No Issue
71	dividendTokenTotalSupply	read	Passed	No Issue
72	getAccountDividendsInfo	external	Passed	No Issue
73	getAccountDividendsInfoAtIndex	external	Passed	No Issue
74	getLastProcessedIndex	external	Passed	No Issue
75	getNumberOfDividendTokenHolders	read	Passed	No Issue
76	process	external	Passed	No Issue
77	initializer	modifier	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

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

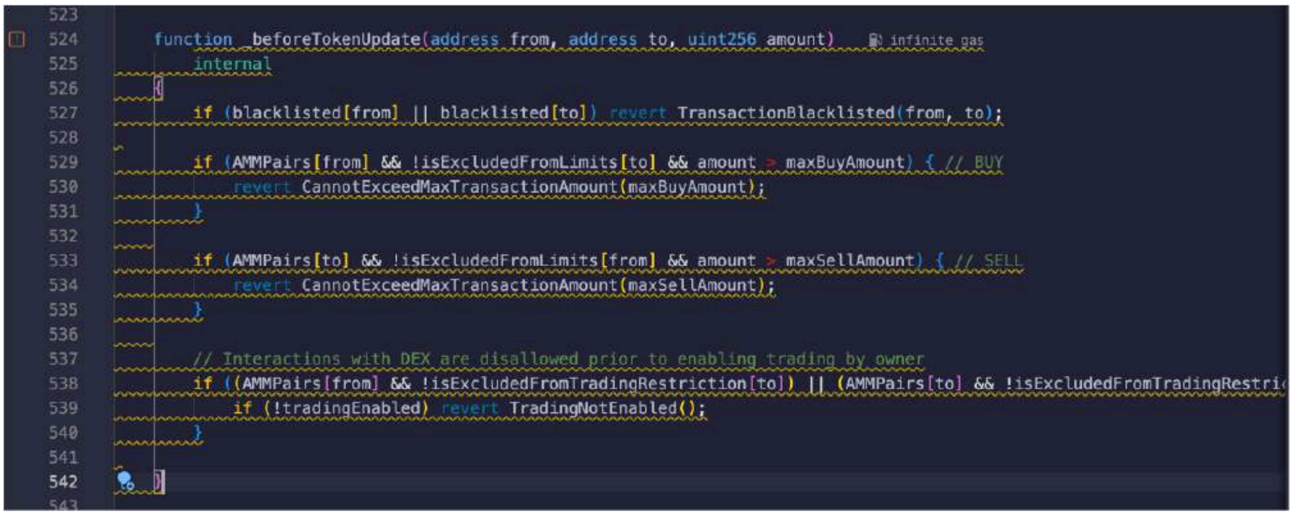
Low

No Low severity vulnerabilities were found.

Very Low / Informational / Best practices:

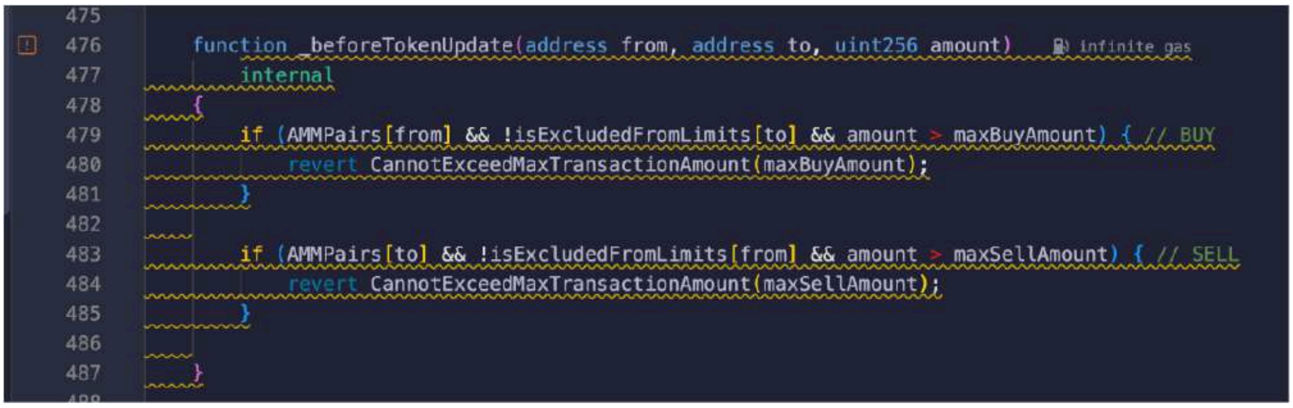
(1) Warning: Function state mutability can be restricted to view:

Token.sol(20lab-v1.9.0-1)



```
    function _beforeTokenUpdate(address from, address to, uint256 amount)   infinite gas
        internal
    {
        if (blacklisted[from] || blacklisted[to]) revert TransactionBlacklisted(from, to);

        if (AMMPairs[from] && !isExcludedFromLimits[to] && amount > maxBuyAmount) { // BUY
            revert CannotExceedMaxTransactionAmount(maxBuyAmount);
        }

        if (AMMPairs[to] && !isExcludedFromLimits[from] && amount > maxSellAmount) { // SELL
            revert CannotExceedMaxTransactionAmount(maxSellAmount);
        }

        // Interactions with DEX are disallowed prior to enabling trading by owner
        if ((AMMPairs[from] && !isExcludedFromTradingRestriction[to]) || (AMMPairs[to] && !isExcludedFromTradingRestri
            if (!tradingEnabled) revert TradingNotEnabled();
        }

    }
```

Token.sol(20lab-v1.9.0-2)



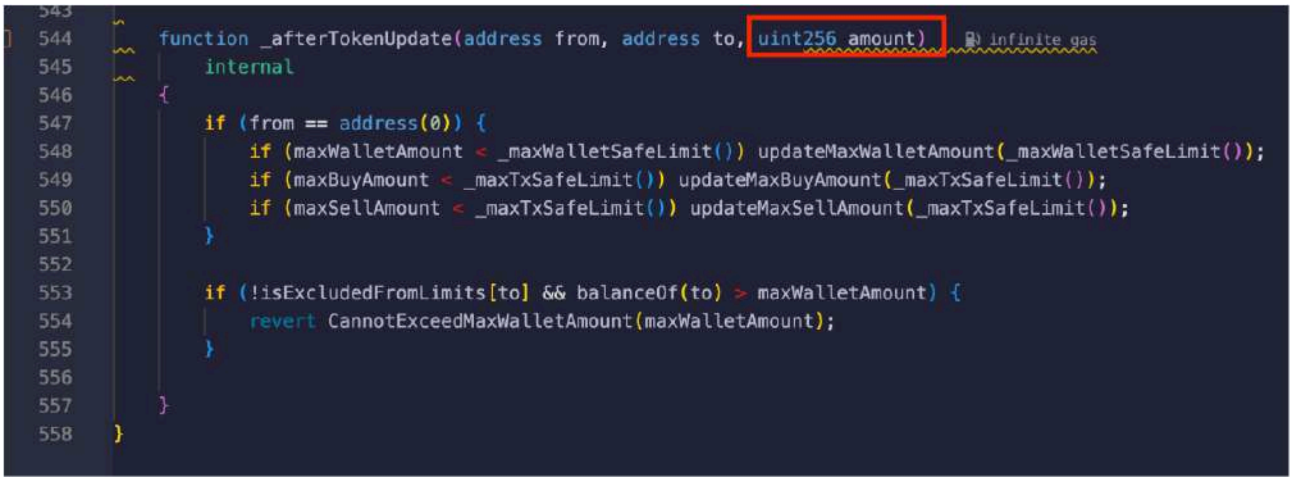
```
function _beforeTokenUpdate(address from, address to, uint256 amount)   infinite gas
    internal
{
    if (AMMPairs[from] && !isExcludedFromLimits[to] && amount > maxBuyAmount) { // BUY
        revert CannotExceedMaxTransactionAmount(maxBuyAmount);
    }

    if (AMMPairs[to] && !isExcludedFromLimits[from] && amount > maxSellAmount) { // SELL
        revert CannotExceedMaxTransactionAmount(maxSellAmount);
    }
}
```

Warning: Function state mutability can be restricted to view since it's just reading the state variables not updating.

Resolution: Warning: Function state mutability can be restricted to view since its just reading the state variables not updating.

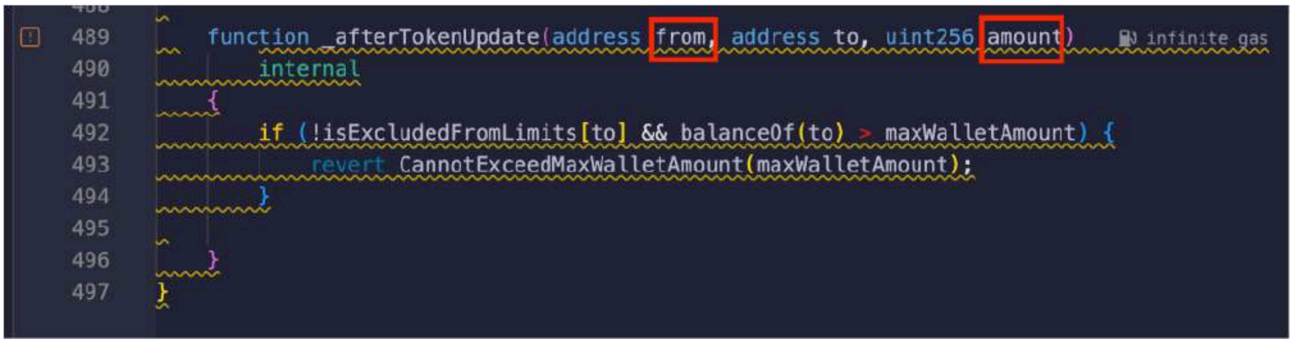
(2) Warning: Unused function parameter. Remove or comment out the variable name to silence this warning:

Token.sol(20lab-v1.9.0-1)



```
function _afterTokenUpdate(address from, address to, uint256 amount)   infinite gas
    internal
{
    if (from == address(0)) {
        if (maxWalletAmount < _maxWalletSafeLimit()) updateMaxWalletAmount(_maxWalletSafeLimit());
        if (maxBuyAmount < _maxTxSafeLimit()) updateMaxBuyAmount(_maxTxSafeLimit());
        if (maxSellAmount < _maxTxSafeLimit()) updateMaxSellAmount(_maxTxSafeLimit());
    }

    if (!isExcludedFromLimits[to] && balanceOf(to) > maxWalletAmount) {
        revert CannotExceedMaxWalletAmount(maxWalletAmount);
    }

}
}
```

Token.sol(20lab-v1.9.0-2)



```
function _afterTokenUpdate(address from, address to, uint256 amount)   infinite gas
    internal
{
    if (!isExcludedFromLimits[to] && balanceOf(to) > maxWalletAmount) {
        revert CannotExceedMaxWalletAmount(maxWalletAmount);
    }

}
}
```

Unused parameter in the _afterTokenUpdate.

Resolution: Please remove it if unused.

Centralization

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

Token.sol(20lab-v1.9.0-1)

- recoverToken: The owner can recover tokens.
- recoverForeignERC20: The owner can recover foreign erc20 tokens.
- blacklist: The owner can add addresses to the blacklist.
- updateSwapThreshold: The owner can update the swap threshold.
- marketingAddressSetup: The owner can update the marketing address.
- marketingFeesSetup: The owner can update marketing fees like buy fee, sell fee and transfer fee.
- liquidityFeesSetup: The owner can update liquidity fees like buy fee, sell fee and transfer fee.
- excludeFromDividends: The owner can manage exclusion from dividends.
- rewardsFeesSetup: The owner can set up the rewards fees.
- excludeFromFees: The owner can manage exclusion from fees.
- setAMMPair: The owner can set an AMM pair address.
- excludeFromLimits: The owner can manage exclusion from transaction limits.
- updateMaxWalletAmount: The owner can update the maximum wallet amount.
- updateMaxBuyAmount: The owner can update the maximum Buy amount.
- updateMaxSellAmount: The owner can update the maximum sell amount.
- enableTrading: The owner can enable trading.
- excludeFromTradingRestriction: The owner can manage exclusion from trading restrictions.

DividendTrackerFunctions.sol

- gasForProcessingSetup: The owner can set the gas for processing dividends.

- callbackGasSetup: The owner can set the gas for a callback.
- claimWaitSetup: The owner can set claim wait time.

DividendTracker.sol

- callbackGasSetup: The owner can set the gas for the callback.
- excludeFromDividends: The owner can manage exclusion from dividends.
- claimWaitSetup: The owner can set claim wait time.
- claim: The owner can claim.
- setBalance: The owner can set the balance.
- process: The owner can process.

Mintable.sol

- mint: The owner can mint new tokens.

Ownable2Step.sol

- transferOwnership: Transfers ownership of the contract to a new account (`newOwner`).

Ownable.sol

- renounceOwnership: Leaves the contract without the owner. It will not be possible to call
  onlyOwner functions.
- transferOwnership: Transfers ownership of the contract to a new account (`newOwner`).

Token.sol(20lab-v1.9.0-2)

- recoverToken: The owner can recover tokens.
- recoverForeignERC20: The owner can recover foreign erc20 tokens.
- updateSwapThreshold: The owner can update the swap threshold.
- marketingAddressSetup: The owner can update the marketing address.
- marketingFeesSetup: The owner can update marketing fees like buy fee, sell fee and transfer fee.

- liquidityFeesSetup: The owner can update liquidity fees like buy fee, sell fee and transfer fee.
- excludeFromDividends: The owner can manage exclusion from dividends.
- rewardsFeesSetup: The owner can set up the rewards fees.
- excludeFromFees: The owner can manage exclusion from fees.
- setAMMPair: The owner can set an AMM pair address.
- excludeFromLimits: The owner can manage exclusion from transaction limits.
- updateMaxWalletAmount: The owner can update the maximum wallet amount.
- updateMaxBuyAmount: The owner can update the maximum Buy amount.
- updateMaxSellAmount: The owner can update the maximum sell amount.

Token.sol→DividendTrackerFunctions.sol(20lab-v1.9.0-2)

- gasForProcessingSetup: The owner can set the gas for processing dividends.
- claimWaitSetup: The owner can set claim wait time.

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 20lab-v1.9.0-1 and 20lab-v1.9.0-2 web link. And we have used all possible tests based on given objects as files. We observed 2 informational issues in the smart contracts. but those are not critical. **So, the smart contracts are 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 the 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 the 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 white box penetration testing. We look at the project's website 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, and 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.

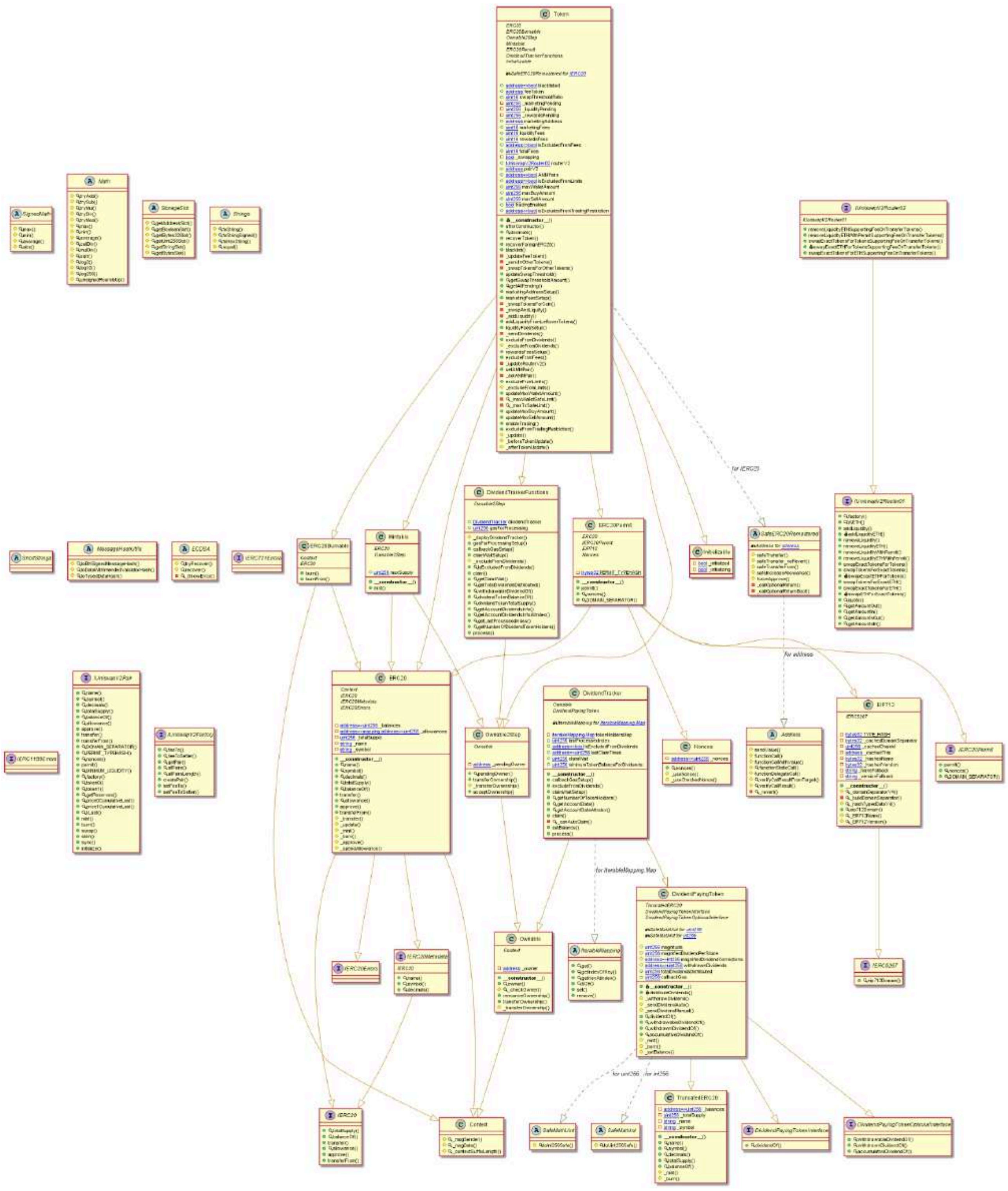
Technical Disclaimer

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.

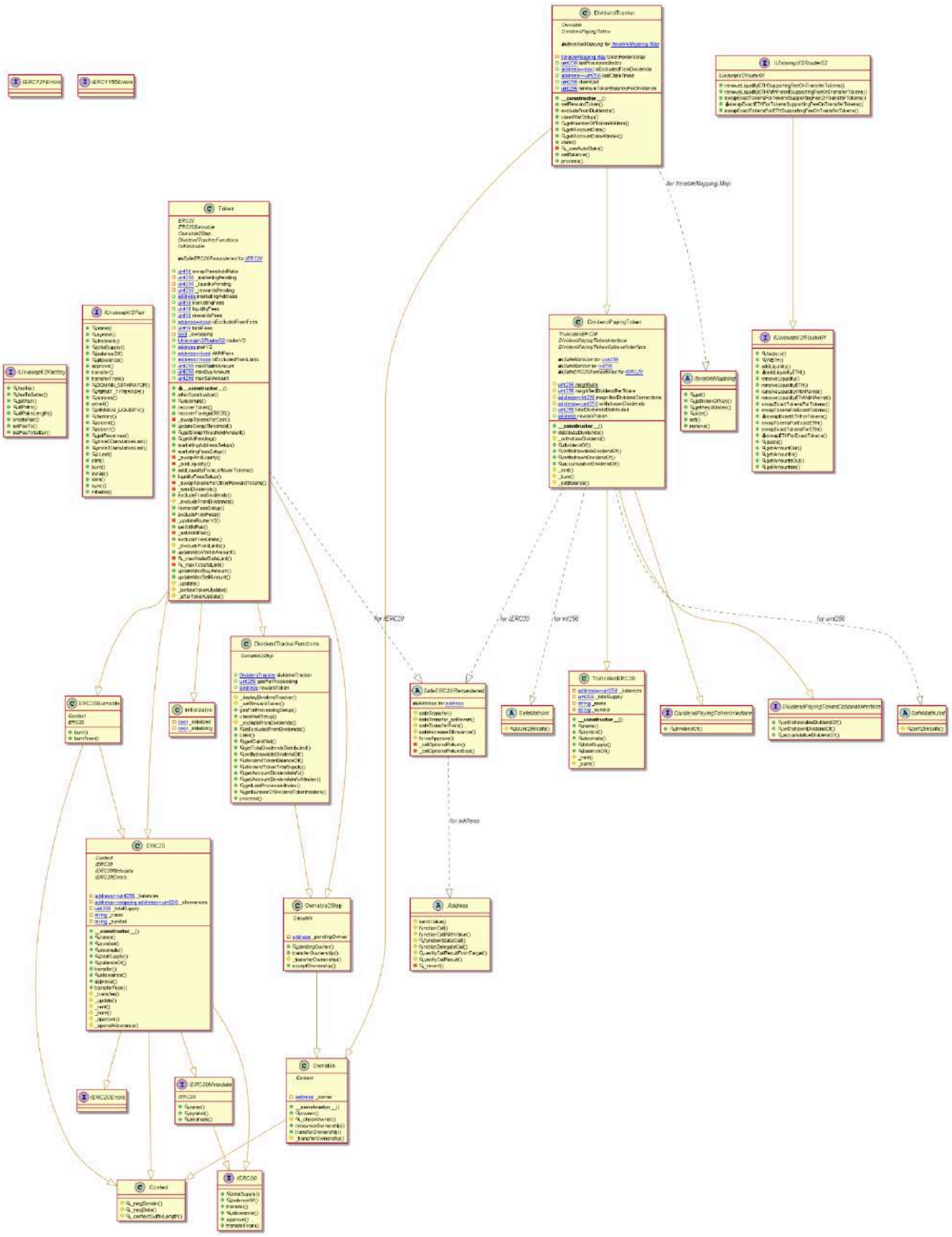
Appendix

Code Flow Diagram - 20Lab

Token Diagram(20lab-v1.9.0-1)



Token Diagram(20lab-v1.9.0-2)



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.

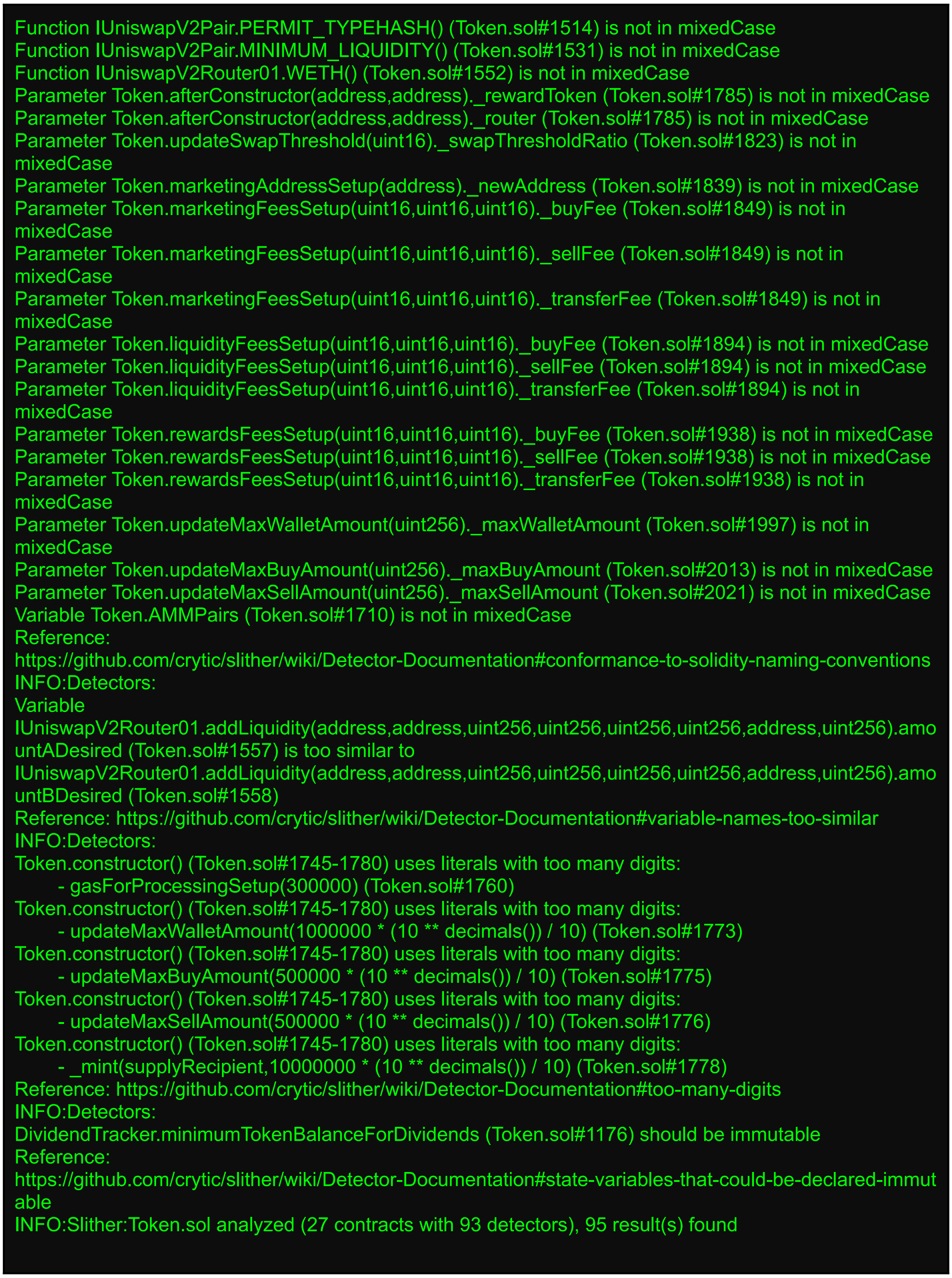
We did the analysis of the project altogether. Below are the results.

Slither Log >> Token.sol(20lab-v1.9.0-1)

```
INFO:Detectors:
DividendTracker.getAccountData(address) (Token.sol#2711-2740) uses timestamp for comparisons
	Dangerous comparisons:
	- nextClaimTime > block.timestamp (Token.sol#2739)
DividendTracker._canAutoClaim(uint256) (Token.sol#2770-2774) uses timestamp for comparisons
	Dangerous comparisons:
	- block.timestamp < lastClaimTime (Token.sol#2771)
	- block.timestamp - lastClaimTime >= claimWait (Token.sol#2773)
ERC20Permit.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (Token.sol#2953-2976)
uses timestamp for comparisons
	Dangerous comparisons:
	- block.timestamp > deadline (Token.sol#2962)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Token._update(address,address,uint256) (Token.sol#3380-3465) has a high cyclomatic complexity (21).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-complexity
INFO:Detectors:
Pragma version^0.8.20 (Token.sol#4) necessitates a version too recent to be trusted. Consider deploying
with 0.8.18.
solc-0.8.20 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
Parameter DividendTracker.getAccountData(address)._account (Token.sol#2711) is not in mixedCase
Function ERC20Permit.DOMAIN_SEPARATOR() (Token.sol#2989-2991) is not in mixedCase
Parameter Token.afterConstructor(address,address)._feeToken (Token.sol#3111) is not in mixedCase
Parameter Token.afterConstructor(address,address)._router (Token.sol#3111) is not in mixedCase
Parameter Token.updateSwapThreshold(uint16)._swapThresholdRatio (Token.sol#3158) is not in
mixedCase
Parameter Token.marketingAddressSetup(address)._newAddress (Token.sol#3174) is not in mixedCase
Parameter Token.marketingFeesSetup(uint16,uint16,uint16)._buyFee (Token.sol#3184) is not in
mixedCase
Parameter Token.marketingFeesSetup(uint16,uint16,uint16)._sellFee (Token.sol#3184) is not in
mixedCase
Parameter Token.marketingFeesSetup(uint16,uint16,uint16)._transferFee (Token.sol#3184) is not in
mixedCase
Parameter Token.liquidityFeesSetup(uint16,uint16,uint16)._buyFee (Token.sol#3244) is not in mixedCase
Parameter Token.liquidityFeesSetup(uint16,uint16,uint16)._sellFee (Token.sol#3244) is not in mixedCase
Parameter Token.liquidityFeesSetup(uint16,uint16,uint16)._transferFee (Token.sol#3244) is not in
mixedCase
Parameter Token.rewardsFeesSetup(uint16,uint16,uint16)._buyFee (Token.sol#3274) is not in mixedCase
Parameter Token.rewardsFeesSetup(uint16,uint16,uint16)._sellFee (Token.sol#3274) is not in mixedCase
Parameter Token.rewardsFeesSetup(uint16,uint16,uint16)._transferFee (Token.sol#3274) is not in
mixedCase
Parameter Token.updateMaxWalletAmount(uint256)._maxWalletAmount (Token.sol#3333) is not in
mixedCase
Parameter Token.updateMaxBuyAmount(uint256)._maxBuyAmount (Token.sol#3349) is not in mixedCase
Parameter Token.updateMaxSellAmount(uint256)._maxSellAmount (Token.sol#3357) is not in mixedCase
Variable Token.AMMPairs (Token.sol#3023) is not in mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable
IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amo
untADesired (Token.sol#1521) is too similar to
IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amo
untBDesired (Token.sol#1522)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
ShortStrings.slitherConstructorConstantVariables() (Token.sol#921-1001) uses literals with too many
digits:
        - FALLBACK_SENTINEL =
0x00000000000000000000000000000000000000000000000000000000000000FF (Token.sol#923)
Token.constructor() (Token.sol#3066-3106) uses literals with too many digits:
        - gasForProcessingSetup(300000) (Token.sol#3083)
Token.constructor() (Token.sol#3066-3106) uses literals with too many digits:
        - updateMaxWalletAmount(1000000 * (10 ** decimals()) / 10) (Token.sol#3096)
Token.constructor() (Token.sol#3066-3106) uses literals with too many digits:
        - updateMaxBuyAmount(500000 * (10 ** decimals()) / 10) (Token.sol#3098)
Token.constructor() (Token.sol#3066-3106) uses literals with too many digits:
        - updateMaxSellAmount(500000 * (10 ** decimals()) / 10) (Token.sol#3099)
Token.constructor() (Token.sol#3066-3106) uses literals with too many digits:
        - _mint(supplyRecipient,10000000 * (10 ** decimals()) / 10) (Token.sol#3104)
Token.constructor() (Token.sol#3066-3106) uses literals with too many digits:
        - Mintable(12300000) (Token.sol#3069)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
DividendTracker.minimumTokenBalanceForDividends (Token.sol#2659) should be immutable
Mintable.maxSupply (Token.sol#2479) should be immutable
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immut
able
INFO:Slither:Token.sol analyzed (40 contracts with 93 detectors), 171 result(s) found
```

Slither Log >> Token.sol(20lab-v1.9.0-2)

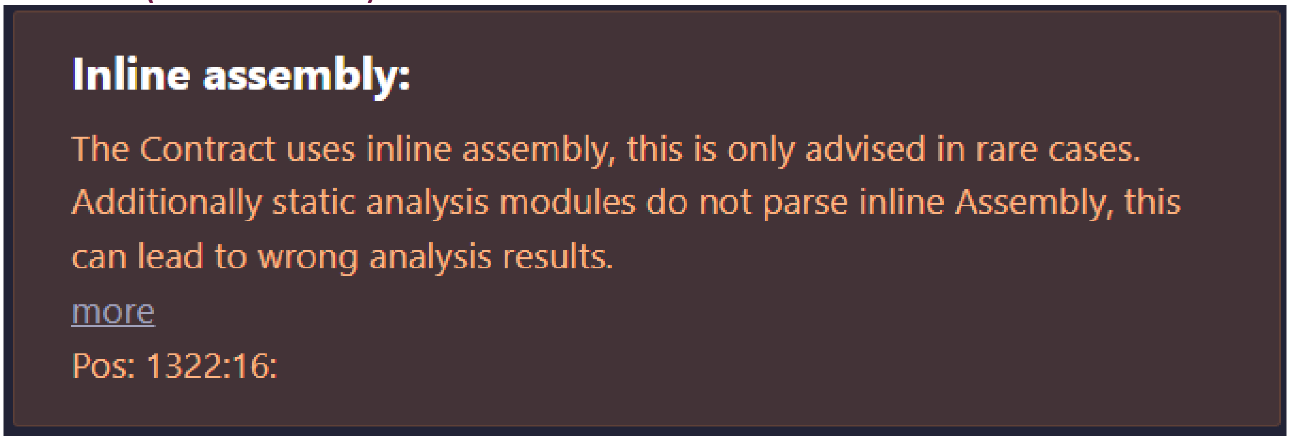
```
INFO:Detectors:
Token._update(address,address,uint256) (Token.sol#2030-2115) has a high cyclomatic complexity (21).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-complexity
INFO:Detectors:
Pragma version^0.8.20 (Token.sol#4) necessitates a version too recent to be trusted. Consider deploying
with 0.8.18.
solc-0.8.20 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function SafeERC20Remastered.safeTransfer_noRevert(IERC20,address,uint256) (Token.sol#250-252)
is not in mixedCase
Constant DividendPayingToken.magnitude (Token.sol#1022) is not in
UPPER_CASE_WITH_UNDERSCORES
Parameter DividendTracker.setRewardToken(address)._rewardToken (Token.sol#1187) is not in
mixedCase
Parameter DividendTracker.getAccountData(address)._account (Token.sol#1224) is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (Token.sol#1513) is not in mixedCase
```



```
Function IUniswapV2Pair.PERMIT_TYPEHASH() (Token.sol#1514) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (Token.sol#1531) is not in mixedCase
Function IUniswapV2Router01.WETH() (Token.sol#1552) is not in mixedCase
Parameter Token.afterConstructor(address,address)._rewardToken (Token.sol#1785) is not in mixedCase
Parameter Token.afterConstructor(address,address)._router (Token.sol#1785) is not in mixedCase
Parameter Token.updateSwapThreshold(uint16)._swapThresholdRatio (Token.sol#1823) is not in
mixedCase
Parameter Token.marketingAddressSetup(address)._newAddress (Token.sol#1839) is not in mixedCase
Parameter Token.marketingFeesSetup(uint16,uint16,uint16)._buyFee (Token.sol#1849) is not in
mixedCase
Parameter Token.marketingFeesSetup(uint16,uint16,uint16)._sellFee (Token.sol#1849) is not in
mixedCase
Parameter Token.marketingFeesSetup(uint16,uint16,uint16)._transferFee (Token.sol#1849) is not in
mixedCase
Parameter Token.liquidityFeesSetup(uint16,uint16,uint16)._buyFee (Token.sol#1894) is not in mixedCase
Parameter Token.liquidityFeesSetup(uint16,uint16,uint16)._sellFee (Token.sol#1894) is not in mixedCase
Parameter Token.liquidityFeesSetup(uint16,uint16,uint16)._transferFee (Token.sol#1894) is not in
mixedCase
Parameter Token.rewardsFeesSetup(uint16,uint16,uint16)._buyFee (Token.sol#1938) is not in mixedCase
Parameter Token.rewardsFeesSetup(uint16,uint16,uint16)._sellFee (Token.sol#1938) is not in mixedCase
Parameter Token.rewardsFeesSetup(uint16,uint16,uint16)._transferFee (Token.sol#1938) is not in
mixedCase
Parameter Token.updateMaxWalletAmount(uint256)._maxWalletAmount (Token.sol#1997) is not in
mixedCase
Parameter Token.updateMaxBuyAmount(uint256)._maxBuyAmount (Token.sol#2013) is not in mixedCase
Parameter Token.updateMaxSellAmount(uint256)._maxSellAmount (Token.sol#2021) is not in mixedCase
Variable Token.AMMPairs (Token.sol#1710) is not in mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable
IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amo
untADesired (Token.sol#1557) is too similar to
IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amo
untBDesired (Token.sol#1558)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
Token.constructor() (Token.sol#1745-1780) uses literals with too many digits:
        - gasForProcessingSetup(300000) (Token.sol#1760)
Token.constructor() (Token.sol#1745-1780) uses literals with too many digits:
        - updateMaxWalletAmount(1000000 * (10 ** decimals()) / 10) (Token.sol#1773)
Token.constructor() (Token.sol#1745-1780) uses literals with too many digits:
        - updateMaxBuyAmount(500000 * (10 ** decimals()) / 10) (Token.sol#1775)
Token.constructor() (Token.sol#1745-1780) uses literals with too many digits:
        - updateMaxSellAmount(500000 * (10 ** decimals()) / 10) (Token.sol#1776)
Token.constructor() (Token.sol#1745-1780) uses literals with too many digits:
        - _mint(supplyRecipient,10000000 * (10 ** decimals()) / 10) (Token.sol#1778)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
DividendTracker.minimumTokenBalanceForDividends (Token.sol#1176) should be immutable
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immut
able
INFO:Slither:Token.sol analyzed (27 contracts with 93 detectors), 95 result(s) found
```

Solidity Static Analysis

Token.sol(20lab-v1.9.0-1)

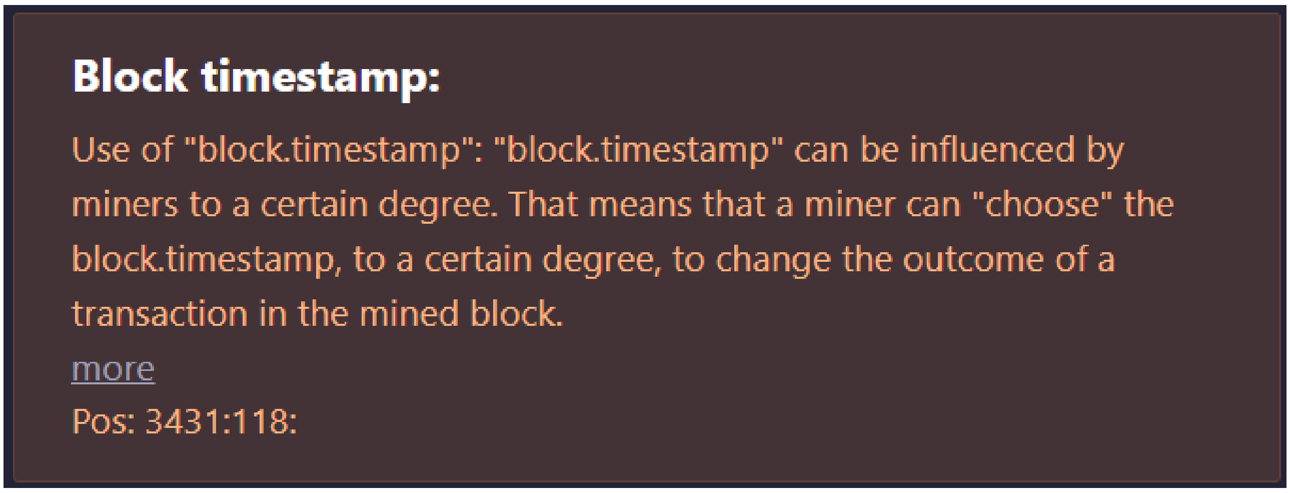


Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

more

Pos: 1322:16:

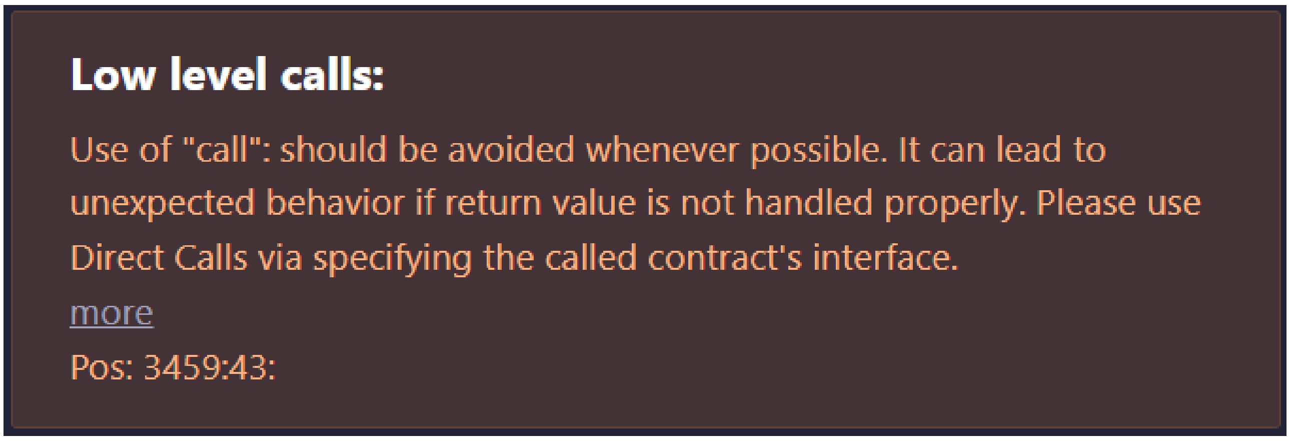


Block timestamp:

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: 3431:118:

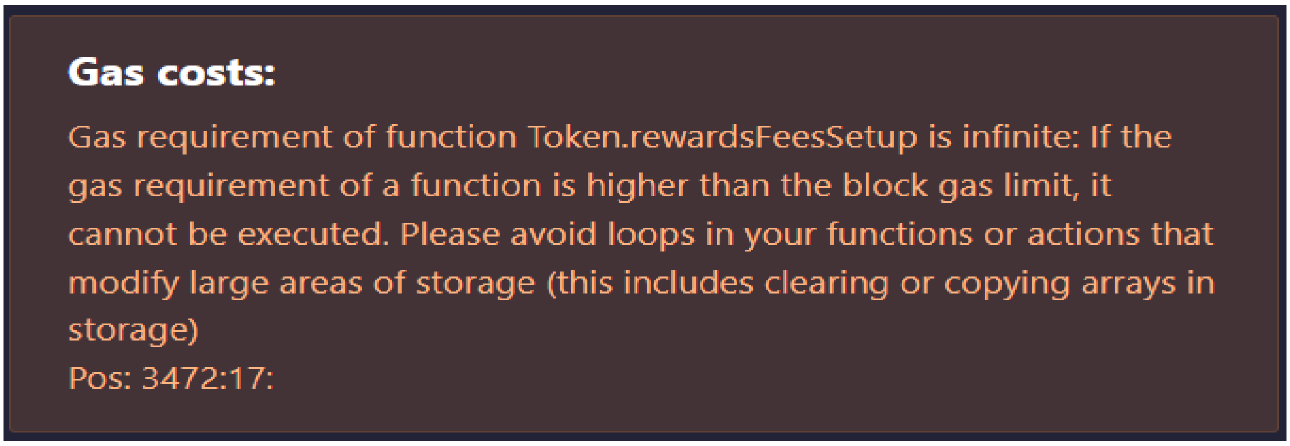


Low level calls:

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

more

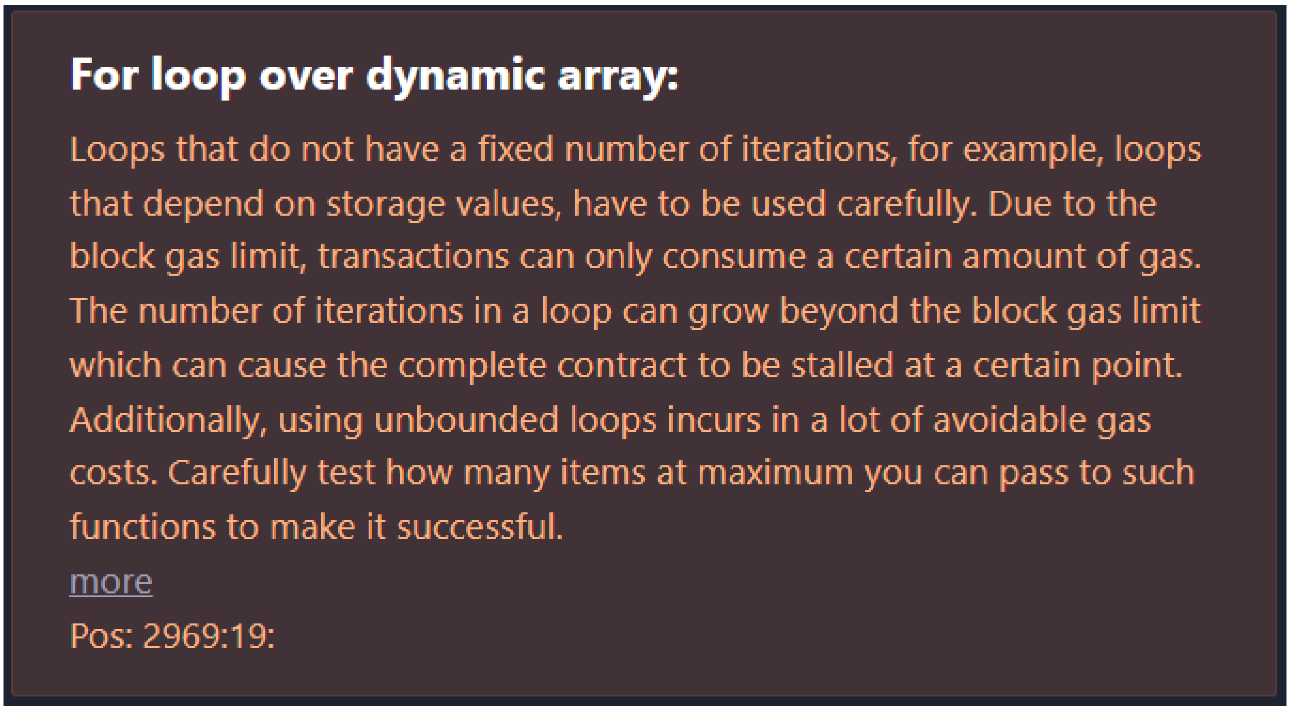
Pos: 3459:43:



Gas costs:

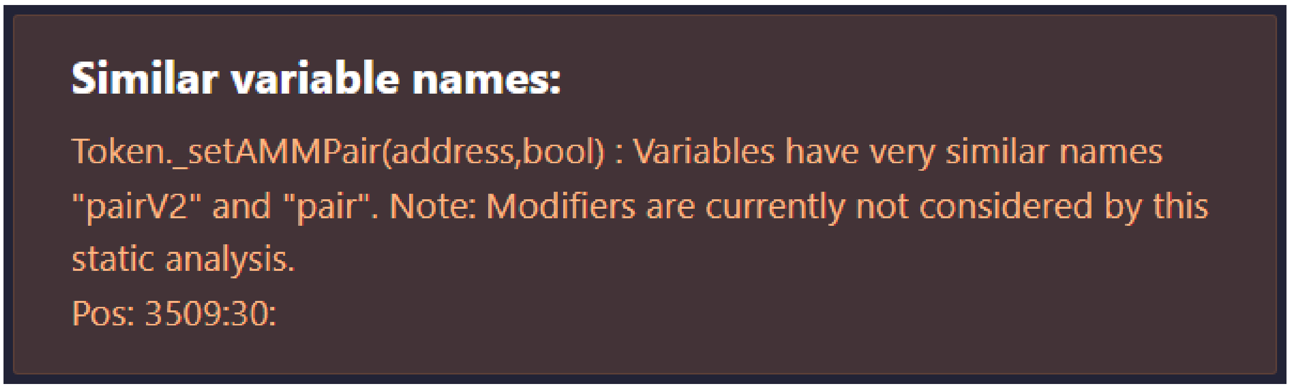
Gas requirement of function Token.rewardsFeesSetup is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 3472:17:



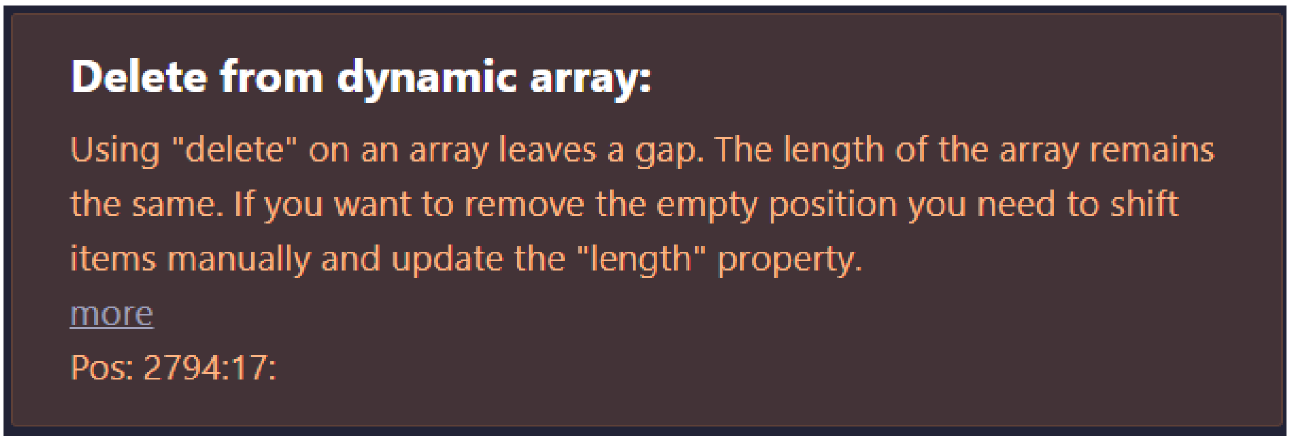
For loop over dynamic array:

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: 2969:19:



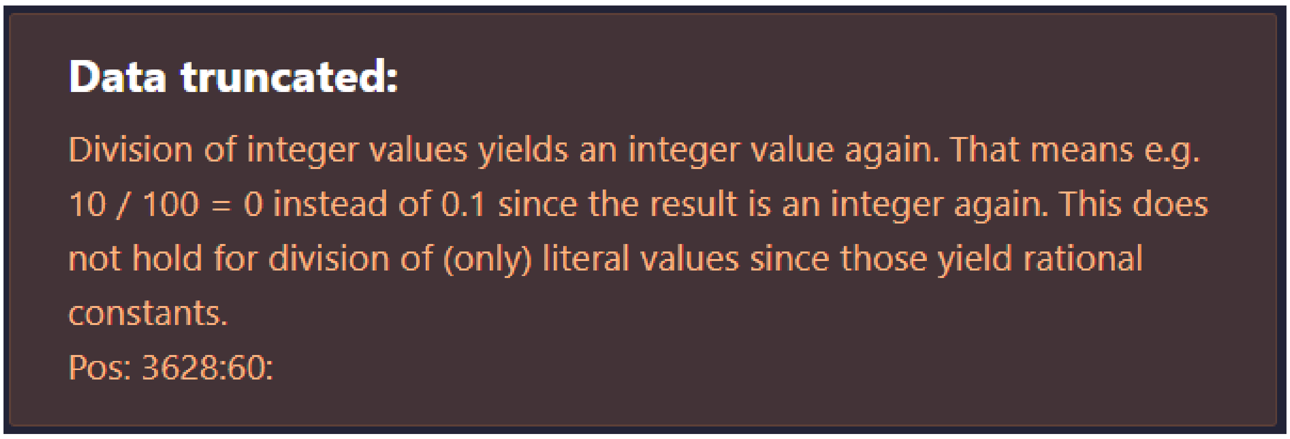
Similar variable names:

Token._setAMMPair(address,bool) : Variables have very similar names "pairV2" and "pair". Note: Modifiers are currently not considered by this static analysis.
Pos: 3509:30:



Delete from dynamic array:

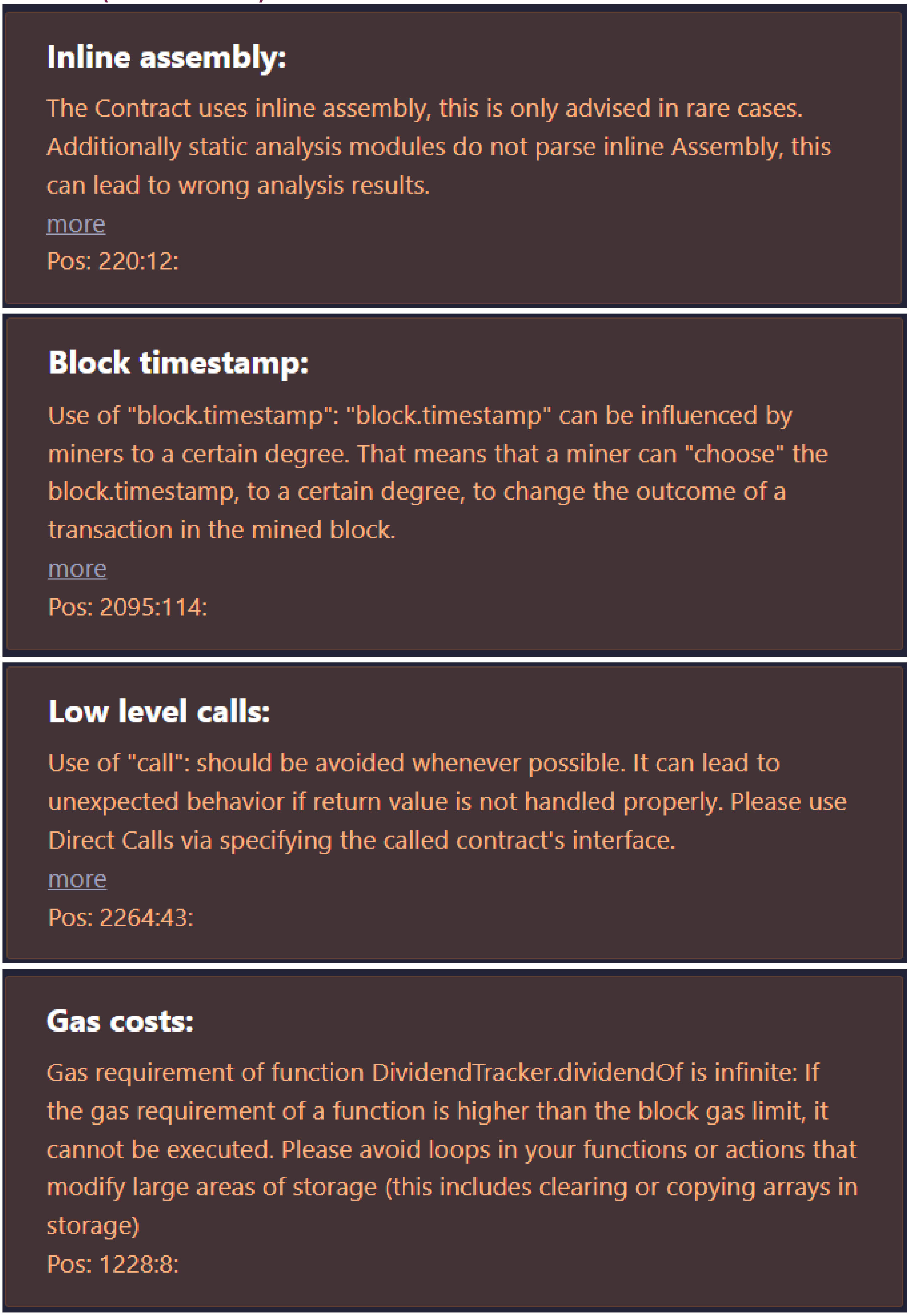
Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 2794:17:



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: 3628:60:

Token.sol(20lab-v1.9.0-2)



Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 220:12:

Block timestamp:

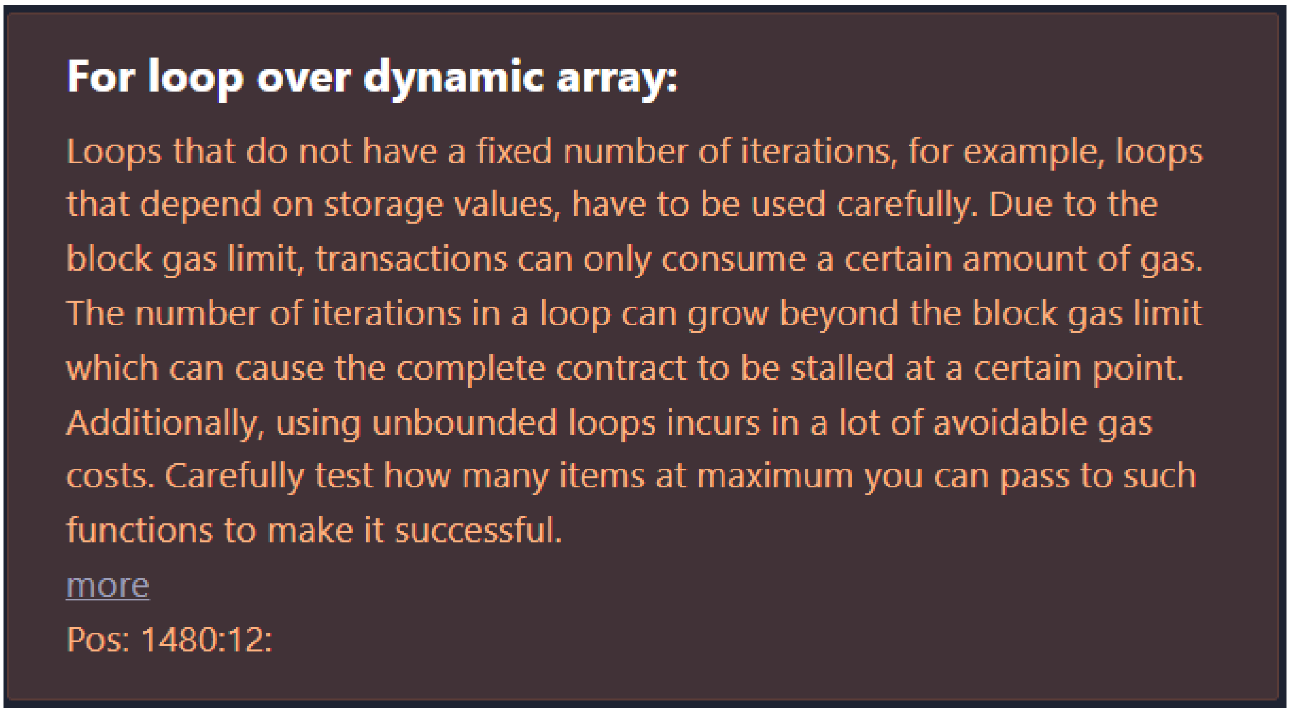
Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 2095:114:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.
more
Pos: 2264:43:

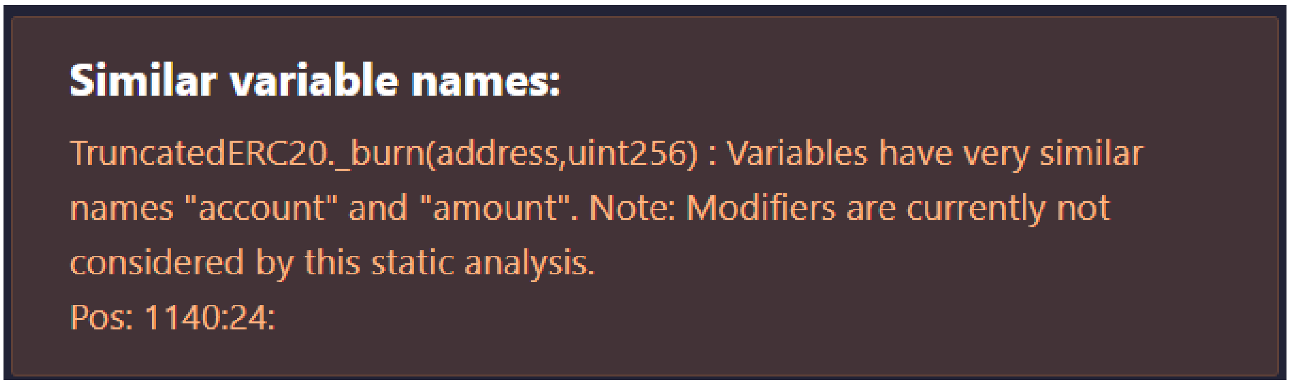
Gas costs:

Gas requirement of function DividendTracker.dividendOf 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: 1228:8:



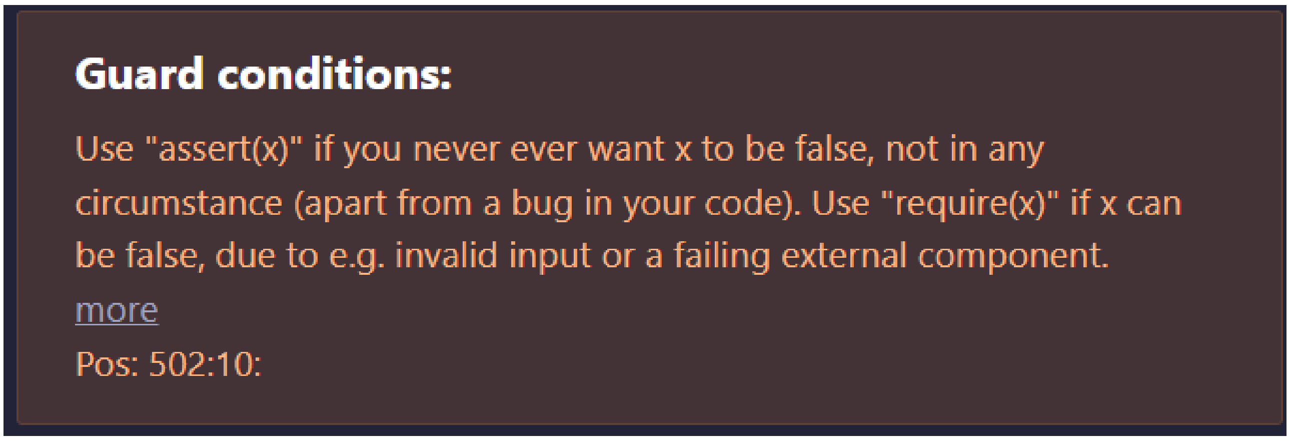
For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 1480:12:



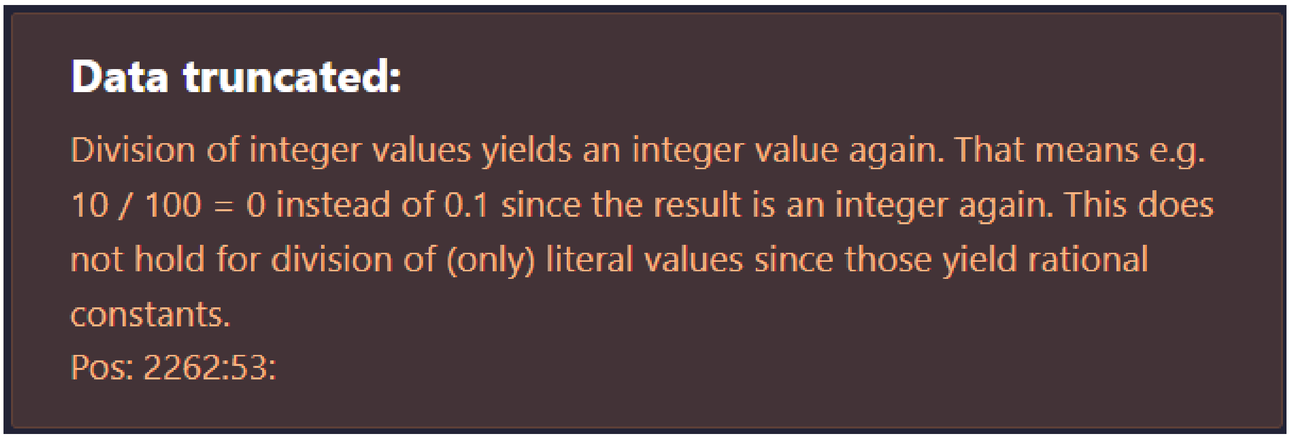
Similar variable names:

TruncatedERC20._burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 1140:24:



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: 502:10:

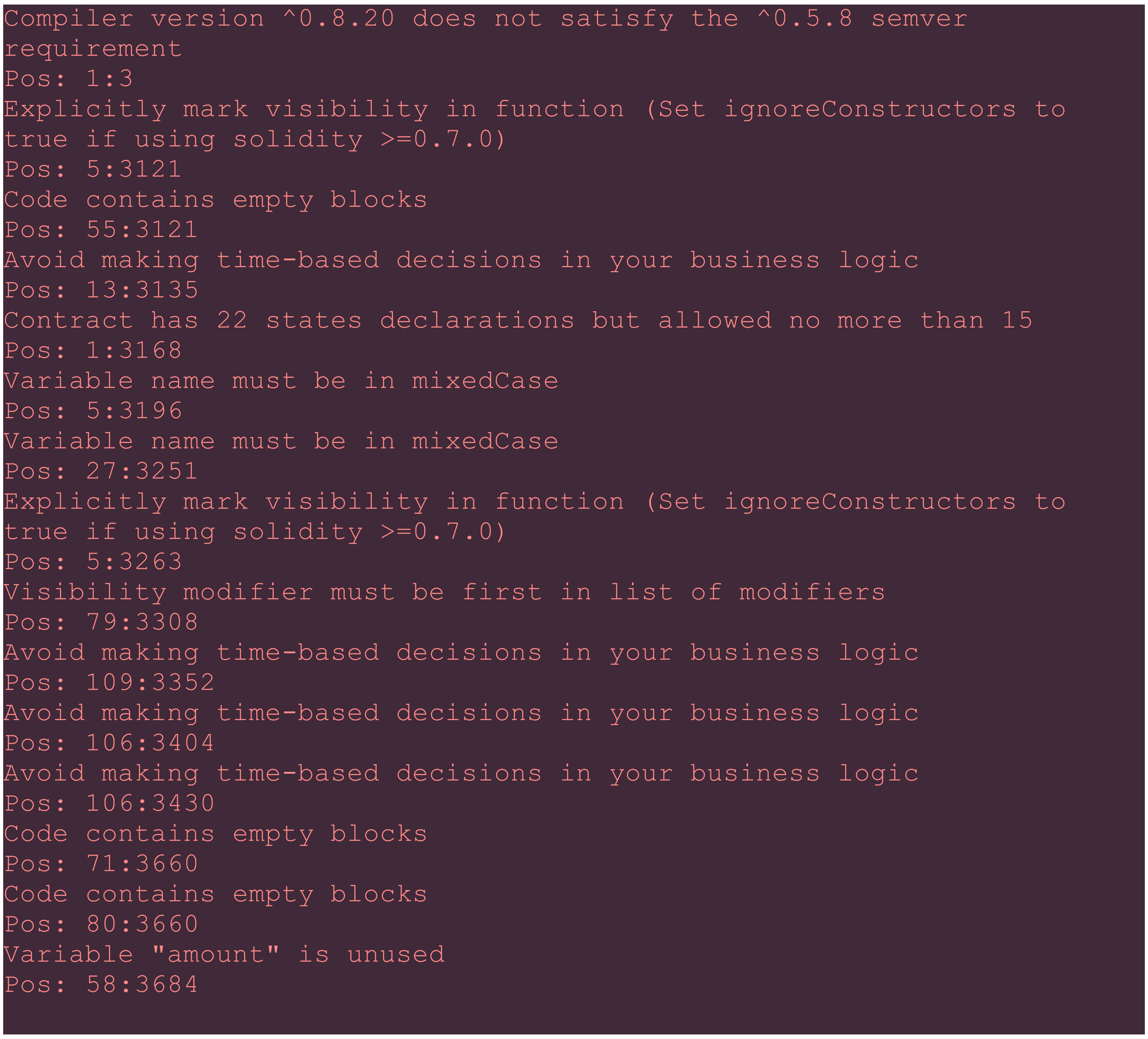


Data truncated:

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: 2262:53:

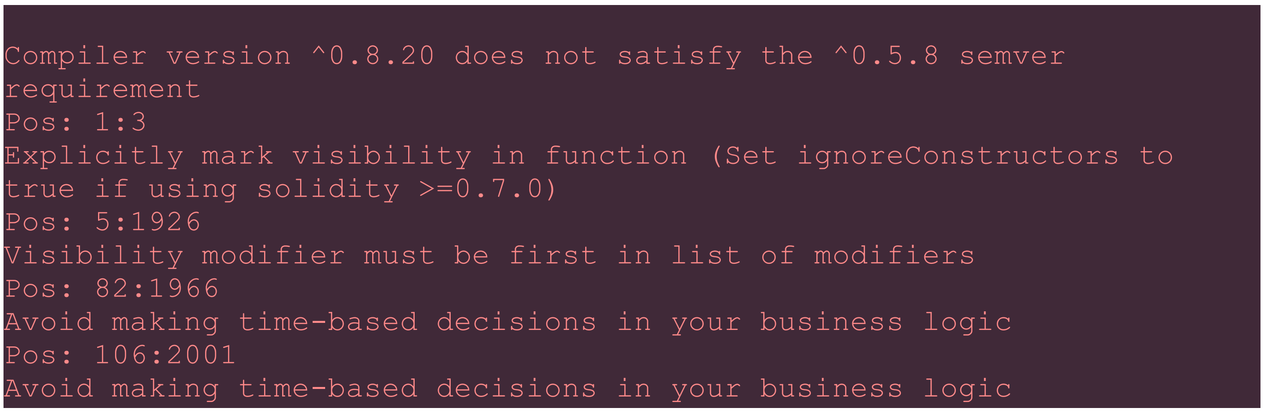
Solhint Linter

Token.sol(20lab-v1.9.0-1)

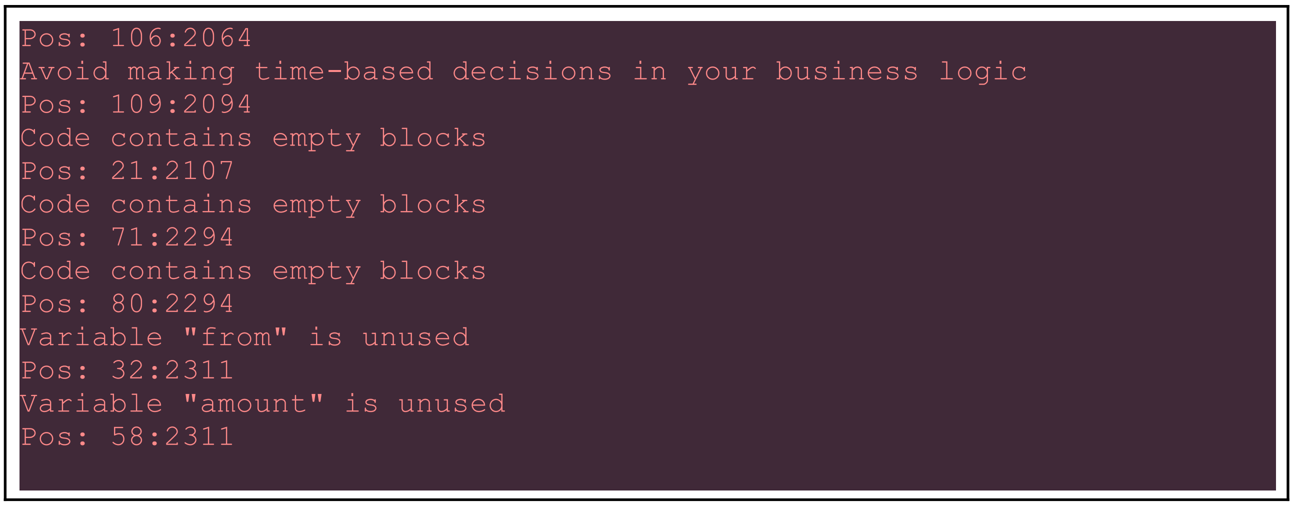


```
Compiler version ^0.8.20 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:3
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:3121
Code contains empty blocks
Pos: 55:3121
Avoid making time-based decisions in your business logic
Pos: 13:3135
Contract has 22 states declarations but allowed no more than 15
Pos: 1:3168
Variable name must be in mixedCase
Pos: 5:3196
Variable name must be in mixedCase
Pos: 27:3251
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:3263
Visibility modifier must be first in list of modifiers
Pos: 79:3308
Avoid making time-based decisions in your business logic
Pos: 109:3352
Avoid making time-based decisions in your business logic
Pos: 106:3404
Avoid making time-based decisions in your business logic
Pos: 106:3430
Code contains empty blocks
Pos: 71:3660
Code contains empty blocks
Pos: 80:3660
Variable "amount" is unused
Pos: 58:3684
```

Token.sol(20lab-v1.9.0-2)



```
Compiler version ^0.8.20 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:3
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:1926
Visibility modifier must be first in list of modifiers
Pos: 82:1966
Avoid making time-based decisions in your business logic
Pos: 106:2001
Avoid making time-based decisions in your business logic
```



```
Pos: 106:2064
Avoid making time-based decisions in your business logic
Pos: 109:2094
Code contains empty blocks
Pos: 21:2107
Code contains empty blocks
Pos: 71:2294
Code contains empty blocks
Pos: 80:2294
Variable "from" is unused
Pos: 32:2311
Variable "amount" is unused
Pos: 58:2311
```

Software analysis result:

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