ASSURE DEFI®
THE VERIFICATION GOLD STANDARD

Security Assessment:

Good Trouble TOKEN

June 1, 2024

- Audit Status: **Fail**
- Audit Edition: **Advance**

VERIFIED BY ASSURE DEFI®
INTEGRITY • TRUST • CREDIBILITY

Risk Analysis

Classifications of Manual Risk Results

Classification	Description
Critical	Danger or Potential Problems.
High	Be Careful or Fail test.
Medium	Pass, Not-Detected or Safe Item.
Low	Function Detected

Manual Code Review Risk Results

Contract Privilege	Description
Buy Tax	5%
Sale Tax	5%
Cannot Buy	Pass
Cannot Sale	Pass
Max Tax	40%
Modify Tax	Yes
Fee Check	Pass
Is Honeypot?	Not Detected
Trading Cooldown	Not Detected
Can Pause Trade?	Pass
Pause Transfer?	Not Detected
Max Tx?	Fail
Is Anti Whale?	Detected
Is Anti Bot?	Not Detected

Contract Privilege	Description
Is Blacklist?	Not Detected
Blacklist Check	Pass
is Whitelist?	Detected
Can Mint?	Pass
Is Proxy?	Not Detected
Can Take Ownership?	Not Detected
Hidden Owner?	Not Detected
Owner	no
Self Destruct?	Not Detected
External Call?	Detected
Other?	Not Detected
Holders	0
Auditor Confidence	Medium Risk
KYC Present	No
KYC URL	N/A

The following quick summary it's added to the project overview; however, there are more details about the audit and its results. Please read every detail.

Project Overview

Token Summary

Parameter	Result
Address	0x
Name	Good Trouble
Token Tracker	Good Trouble (GTRB)
Decimals	18
Supply	100,000,000
Platform	ETHEREUM
compiler	v0.8.20+commit.a1b79de6
Contract Name	GTRB
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	
Payment Tx	Corporate

Main Contract Assessed Contract Name

Name	Contract	Live
Good Trouble	0x	Yes

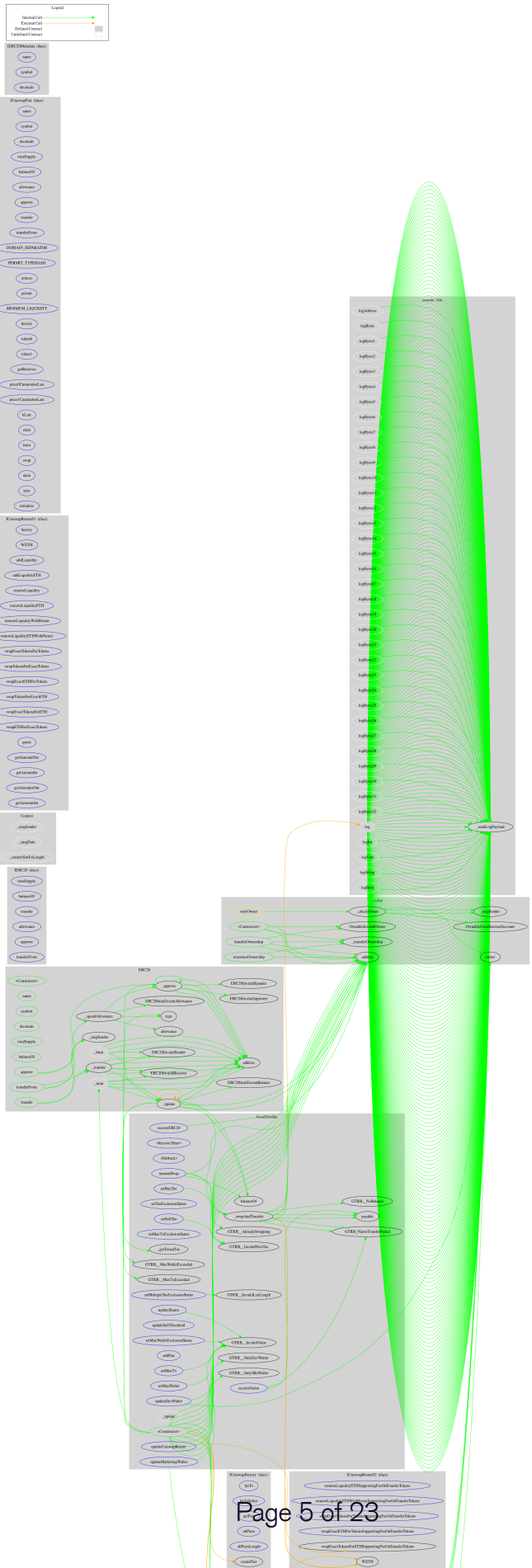
TestNet Contract was Not Assessed

Solidity Code Provided

SolID	File Sha-1	FileName
GTRB	e97c3d685bbca27c9c0103e7904dcbe7	GoodTrouble.sol
GTRB		
GTRB		
GTRB		
GTRB		
GTRB		

Call Graph

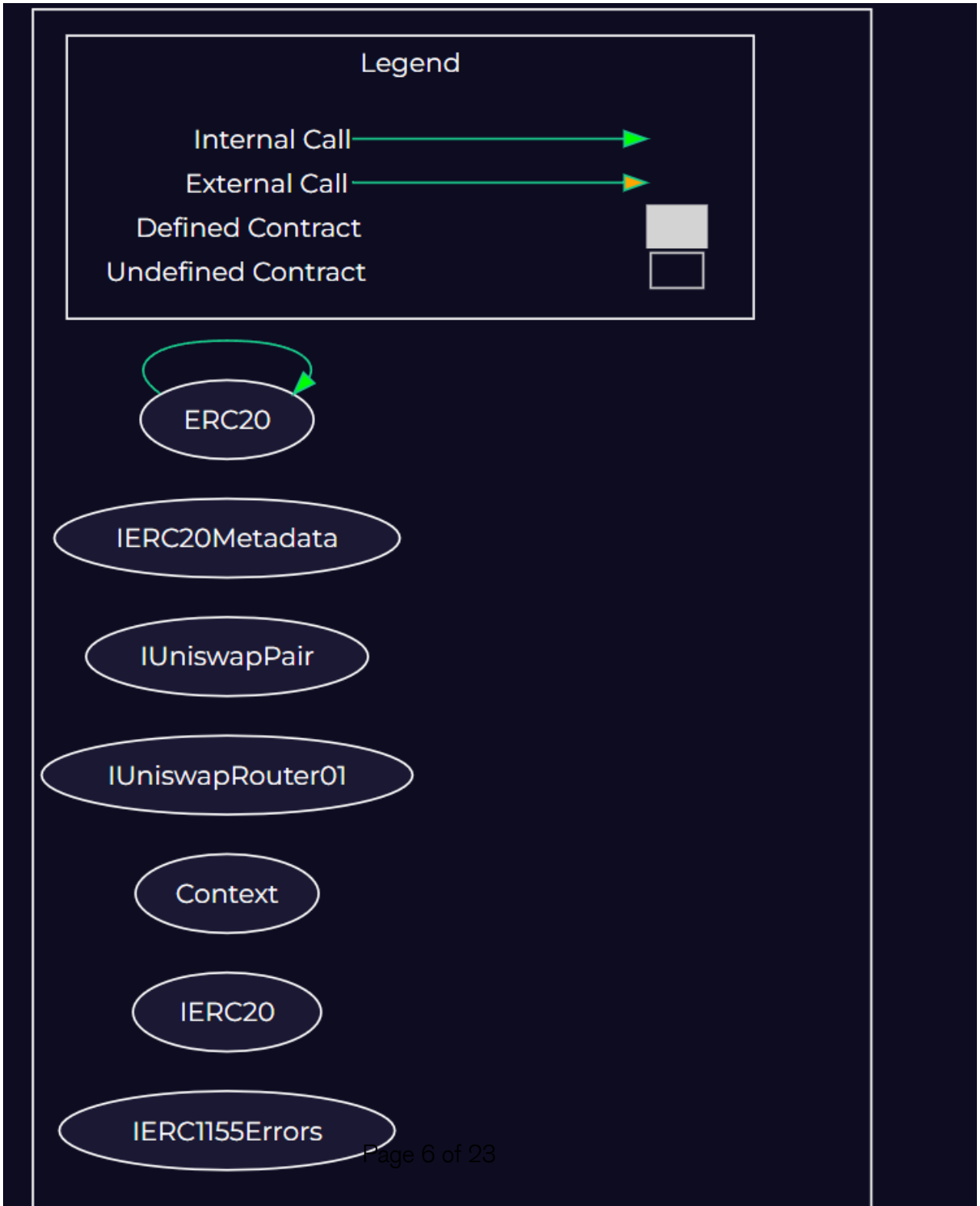
The contract for Good Trouble has the following call graph structure.



Inheritance

The contract for Good Trouble has the following inheritance structure.

The Project has a Total Supply of 100,000,000



Privileged Functions (onlyOwner)

Please Note if the contract is Renounced none of this functions can be executed.

Function Name	Parameters	Visibility
setBuyTax	uint8 _buyTax	external
setSellTax	uint8 _sellTax	external
setTaxExclusionStatus	address _address, bool _status	external
setMultipleTaxExclusionStatus	address[] calldata addresses, bool _status	external
setMaxTxExclusionStatus	address _address, bool _status	external
setMaxWalletExclusionStatus	address _address, bool _status	external
setMaxTx	uint _maxTx	external
setMaxWallet	uint _maxWallet	external
updateDevWallet	address _devWallet	external
updateMarketingWallet	address _marketingWallet	external
updateSellThreshold	uint _sellThreshold	external
manualSwap		external
addPair	address _pair	external

Function Name	Parameters	Visibility
updateShares	uint16 _mktShare, uint16 _devShare	external
updateUniswapRouter	address _router	external
recoverNative	address _to, uint _amount	external
recoverERC20	address _token, address _to	external

GTRB-01 | Potential Sandwich Attacks.

Category	Severity	Location	Status
Security	Low	GoodTrouble.sol: L: 2950	Unresolved

Description

A sandwich attack might happen when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (before the transaction being attacked) a transaction to purchase one of the assets and make profits by back running (after the transaction being attacked) a transaction to sell the asset. The following functions are called without setting restrictions on slippage or minimum output amount, so transactions triggering these functions are vulnerable to sandwich attacks, especially when the input amount is large:

- swapExactTokensForETHSupportingFeeOnTransferTokens()
- addLiquidityETH()

Remediation

We recommend setting reasonable minimum output amounts, instead of 0, based on token prices when calling the aforementioned functions.

Referrences:

What Are Sandwich Attacks in DeFi — and How Can You Avoid Them?.

GTRB-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	Low	GoodTrouble.sol: L: 2819 C: 14, L: 2831 C: 14, L: 2842 C: 14, L: 2860 C: 14, L: 2831 C: 14	Detected

Description

The given input is missing the check for the non-zero address.

The given input is missing the check for the onlyOwners need to be revisited for require..

Remediation

We advise the client to add the check for the passed-in values to prevent unexpected errors as below:

```
...
 require(receiver != address(0), "Receiver is the zero address");
...
...
require(value X limitation, "Your not able to do this function");
...
```

We also recommend customer to review the following function that is missing a required validation. onlyOwners need to be revisited for require..

GTRB-07 | State Variables could be Declared Constant.

Category	Severity	Location	Status
Coding Style	Low	GoodTrouble.sol: L: 2612, L:	Detected

Description

Constant state variables should be declared constant to save gas.

Remediation

Add the constant attribute to state variables that never changes.

https://docs.soliditylang.org/en/latest/contracts.html#constant-state-variables

GTRB-08 | Dead Code Elimination.

Category	Severity	Location	Status
Coding Style	Low	GoodTrouble.sol: L: 1649, L: 277	Detected

Description

Functions that are not used in the contract, and make the code s size bigger.

Remediation

Remove unused functions. dead-code elimination (also known as DCE, dead-code removal, dead-code stripping, or dead-code strip) is a compiler optimization to remove code which does not affect the program results. Removing such code has several benefits: it shrinks program size, an important consideration in some contexts, and it allows the running program to avoid executing irrelevant operations, which reduces its running time. It can also enable further optimizations by simplifying program structure.

https://docs.soliditylang.org/en/latest/cheatsheet.html

GTRB-10 | Initial Token Distribution.

Category	Severity	Location	Status
Centralization / Privilege	High	GoodTrouble.sol: L: 2693	Detected

Description

All of the Good Trouble tokens are sent to the contract deployer when deploying the contract. This could be a centralization risk as the deployer can distribute tokens without obtaining the consensus of the community.

Remediation

We recommend the team to be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key.

Project Action

GTRB-11 | Potential Reentrancy in _swapAndTransfer.

Category	Severity	Location	Status
Optimization	High	GoodTrouble.sol: L: 2950 C: 14	Detected

Description

The _swapAndTransfer function involves external calls which could be exploited for reentrancy.

Remediation

Use reentrancy guards or check-effects-interactions pattern.

Project Action

GTRB-20 | Transfer Delay Mechanism.

Category	Severity	Location	Status
Optimization	Low	GoodTrouble.sol: L: 989	Detected

Description

The transfer delay mechanism (isTransferDelayActive) can only be disabled and not re-enabled, which might be restrictive.

Remediation

Consider allowing re-enabling of the transfer delay if needed.

Project Action

GTRB-21 | Max Transaction/Wallet Limits Control.

Category	Severity	Location	Status
Optimization	Medium	GoodTrouble.sol: L: 2797, L: 2808	Detected

Description

The contract allows the owner to set maximum transaction and wallet limits. While this provides flexibility, it also introduces the risk of unfair trading conditions if misused.

Remediation

Set Reasonable Limits: Ensure that the limits are set to reasonable values that do not hinder normal trading activities. Add Governance Mechanism: Introduce a governance mechanism to allow the community to vote on changes to these limits, reducing centralization risks. Emit Events: Ensure that any changes to these limits emit events for better transparency and tracking. Validation Checks: Add validation checks to ensure the new limits are within acceptable ranges.

Project Action

Technical Findings Summary

Classification of Risk

Severity	Description
Critical	Risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
High	Risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
Medium	Risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform
Low	Risks can be any of the above but on a smaller scale. They generally do not compromise the overall integrity of the Project, but they may be less efficient than other solutions.
Informational	Errors are often recommended to improve the code's style or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

Findings

Severity	Found	Pending	Resolved
Critical	0	0	0
High	2	2	0
Medium	2	2	0
Low	4	4	0
Informational	0	0	0
Total	8	8	0

Social Media Checks

Social Media	URL	Result
Twitter	https://x.com/goodtroubleent	Pass
Other		
Website	https://projectgoodtrouble.com	Pass
Telegram	https://t.me/GoodTroubleEnt	Pass

We recommend to have 3 or more social media sources including a completed working websites.

Social Media Information Notes:

Auditor Notes: undefined

Project Owner Notes:

Audit Result

Final Audit Score

Review	Score
Security Score	75
Auditor Score	75

The Following Score System Has been Added to this page to help understand the value of the audit, the maximum score is 100, however to attain that value the project most pass and provide all the data needed for the assessment. Our Passing Score has been changed to 85 Points for a higher standard, if a project does not attain 85% is an automatic failure. Read our notes and final assessment below.

Audit Fail

Assessment Results

Important Notes:

- Overall Classification:ı
- Classification: Medium Riskı
- Score: 70/100ı
- Conclusion:ı
- The GoodTrouble contract is a well-structured ERC20 token implementation with advanced features for tax management, liquidity provision, and fund distribution. However, several security concerns and areas for improvement have been identified, including reentrancy vulnerabilities, potential gas limit issues, susceptibility to front-running, and complex token distribution logic. Addressing these issues will enhance the contract's security, efficiency, and maintainability.ı
- Recommendations:ı
- Implement ReentrancyGuard to protect against reentrancy attacks.ı
- Optimize gas usage to prevent exceeding the block gas limit.ı
- Implement anti-front-running mechanisms.ı
- Ensure comprehensive checks for max transaction and wallet limits.ı
- Accurately manage and update startTaxTime to prevent manipulation.ı

- Review and enhance error handling for robustness.ı
- Simplify token distribution logic and thoroughly test it.ı
- Use configurable variables for addresses and provide update functions.ı
- Emit events for all critical actions to ensure transparency.ı
- Add comprehensive documentation for all functions and state variables.ı
- By addressing these recommendations, the overall security and functionality of the GoodTrouble contract can be significantly improved.

Auditor Score =75
Audit Fail

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that actagainst the nature of decentralization, such as explicit ownership or specialized access roles incombination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimalEVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on howblock.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functionsbeing invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that mayresult in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to makethe codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code,such as a constructor assignment imposing different require statements on the input variables than a setterfunction.

Coding Best Practices

ERC 20 Conding Standards are a set of rules that each developer should follow to ensure the code meet a set of creterias and is readable by all the developers.

Disclaimer

Assure Defi has conducted an independent security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocation for the Project, and users relying on this report should not consider this as having any merit for financial advice in any shape, form, or nature. The contracts audited do not account for any economic developments that the Project in question may pursue, and the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude, and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are entirely free of exploits, bugs, vulnerabilities or deprecation of technologies.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence, regardless of the findings presented. Information is provided ‘as is, and Assure Defi is under no covenant to audited completeness, accuracy, or solidity of the contracts. In no event will Assure Defi or its partners, employees, agents, or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions or actions with regards to the information provided in this audit report.

The assessment services provided by Assure Defi are subject to dependencies and are under continuing development. You agree that your access or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The services may access, and depend upon, multiple layers of third parties.