



# SMART CONTRACT AUDIT

 interfinetwork

 hello@interfi.network

 <https://interfi.network>

PREPARED FOR

**LENDORA PROTOCOL**



# INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	Lendora Protocol
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Contract	Multiple contracts
Source	<a href="https://github.com/LendoraProtocol/lendora-protocol">https://github.com/LendoraProtocol/lendora-protocol</a>
Blockchain	Not available
Centralization	Active ownership
Commit	9648eb73a8f0680c09b8cb675ac139e256829567
Website	<a href="http://lendora.xyz/">http://lendora.xyz/</a>
X (Twitter)	<a href="https://twitter.com/LendoraProtocol/">https://twitter.com/LendoraProtocol/</a>
Report Date	November 02, 2023


 Verify the authenticity of this report on our website: <https://www.github.com/interfinetwork>





## EXECUTIVE SUMMARY

InterFi has performed the automated and manual analysis of solidity codes. Solidity codes were reviewed for common contract vulnerabilities and centralized exploits. Here's a quick audit summary:

Status	Critical <span style="color: red;">●</span>	Major <span style="color: orange;">●</span>	Medium <span style="color: yellow;">●</span>	Minor <span style="color: green;">●</span>	Unknown <span style="color: brown;">●</span>
Open	1	2	3	9	1
Acknowledged	0	0	0	0	1
Resolved	0	0	0	0	0
Important Privileges	Review PAGE 09 for noteworthy centralized and access restricted functions				

 Please note that smart contracts deployed on blockchains aren't resistant to exploits, vulnerabilities and/or hacks. Blockchain and cryptography assets utilize new and emerging technologies. These technologies present a high level of ongoing risks. For a detailed understanding of risk severity, source code vulnerability, and audit limitations, kindly review the audit report thoroughly.

 Please note smart contract deployment status is available. Contract files can be modified by project team before deployment.

 Please note that centralization privileges regardless of their inherited risk status – constitute an elevated impact on smart contract safety and security.



# TABLE OF CONTENTS

TABLE OF CONTENTS .....	4
SCOPE OF WORK.....	5
AUDIT METHODOLOGY .....	6
RISK CATEGORIES .....	9
MANUAL REVIEW.....	10
AUTOMATED ANALYSIS.....	31
DISCLAIMERS .....	47
ABOUT INTERFI NETWORK .....	50

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL



## SCOPE OF WORK

InterFi was consulted by Lendora Protocol to conduct the smart contract audit of their solidity source codes. **The audit scope of work is strictly limited to mentioned solidity file(s) only:**

BasePriceOracle.sol	CToken.sol	ErrorReporter.sol
CErc20.sol	CTokenInterface.sol	Exponential.sol
CErc20Delegate.sol	CarefulMath.sol	ExponentialNoError.sol
CErc20Delegator.sol	ChainlinkPriceOracle.sol	InterestRateModel.sol
CErc20Immutable.sol	Comptroller.sol	JumpRateModelV2.sol
CEther.sol	ComptrollerStorage.sol	PriceOracle.sol
CEtherDelegate.sol	ComptrollerInterface.sol	RewardDistributor.sol
CEtherDelegator.sol	EIP20NonStandardInterface.sol	Unitroller.sol

## ASSUMPTIONS

Following trust assumptions are made to complete this audit report. Please note – any of the following may introduce critical vulnerabilities that could impact the entire protocol:

- CToken admin are trusted entities.
- Unitroller admin are trusted entities.
- Comptroller admin are trusted entities.
- CToken comptroller is an instance of Unitroller.
- Unitroller comptrollerImplementation is an instance of Comptroller.
- It is assumed that administrator and price feeds are available, honest and not compromised.
- Price feeds are used to retrieve prices of different assets. Well-known oracle attacks include price manipulation and denial of service (DoS) attacks that make price unavailable.
- Every cToken is coupled with an underlying asset. Note that the contract code of new underlying assets may introduce vulnerabilities that could impact the entire protocol.



## SOURCE

<https://github.com/LendoraProtocol/lendora-protocol>

## SHA AT SOURCE

f273ab2e2ca052ba82ce1daca526246365da45db

## AUDIT COMMIT (FOR INTERNAL USE)

9648eb73a8f0680c09b8cb675ac139e256829567

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL



# AUDIT METHODOLOGY

Smart contract audits are conducted using a set of standards and procedures. Mutual collaboration is essential to performing an effective smart contract audit. Here's a brief overview of InterFi's auditing process and methodology:

## CONNECT

- The onboarding team gathers source codes, and specifications to make sure we understand the size, and scope of the smart contract audit.

## AUDIT

- Automated analysis is performed to identify common contract vulnerabilities. We may use the following third-party frameworks and dependencies to perform the automated analysis:
  - Remix IDE Developer Tool
  - Open Zeppelin Code Analyzer
  - SWC Vulnerabilities Registry
  - DEX Dependencies, e.g., Pancakeswap, Uniswap
- Simulations are performed to identify centralized exploits causing contract and/or trade locks.
- A manual line-by-line analysis is performed to identify contract issues and centralized privileges.

We may inspect below mentioned common contract vulnerabilities, and centralized exploits:

Centralized Exploits	<ul style="list-style-type: none"><li>○ Token Supply Manipulation</li><li>○ Access Control and Authorization</li><li>○ Assets Manipulation</li><li>○ Ownership Control</li><li>○ Liquidity Access</li><li>○ Stop and Pause Trading</li><li>○ Ownable Library Verification</li></ul>
----------------------	---



## Common Contract Vulnerabilities


- Integer Overflow
- Lack of Arbitrary limits
- Incorrect Inheritance Order
- Typographical Errors
- Requirement Violation
- Gas Optimization
- Coding Style Violations
- Re-entrancy
- Third-Party Dependencies
- Potential Sandwich Attacks
- Irrelevant Codes
- Divide before multiply
- Conformance to Solidity Naming Guides
- Compiler Specific Warnings
- Language Specific Warnings

**REPORT**

- The auditing team provides a preliminary report specifying all the checks which have been performed and the findings thereof.
- The client's development team reviews the report and makes amendments to solidity codes.
- The auditing team provides the final comprehensive report with open and unresolved issues.

**PUBLISH**

- The client may use the audit report internally or disclose it publicly.

 It is important to note that there is no pass or fail in the audit, it is recommended to view the audit as an unbiased assessment of the safety of solidity codes.





## RISK CATEGORIES

Smart contracts are generally designed to hold, approve, and transfer tokens. This makes them very tempting attack targets. A successful external attack may allow the external attacker to directly exploit. A successful centralization-related exploit may allow the privileged role to directly exploit. All risks which are identified in the audit report are categorized here for the reader to review:

Risk Type	Definition
Critical 🛑	These risks could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
Major 🟡	These risks are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to high-risk severity.
Medium 🟡	These risks should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution. Low-risk re-entrancy-related vulnerabilities should be fixed to deter exploits.
Minor 🟢	These risks do not pose a considerable risk to the contract or those who interact with it. They are code-style violations and deviations from standard practices. They should be highlighted and fixed nonetheless.
Unknown 🟤	These risks pose uncertain severity to the contract or those who interact with it. They should be fixed immediately to mitigate the risk uncertainty.

All statuses which are identified in the audit report are categorized here for the reader to review:

Status Type	Definition
Open	Risks are open.
Acknowledged	Risks are acknowledged, but not fixed.
Resolved	Risks are acknowledged and fixed.



## MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Centralized privileges	Critical ●
CEN-07	Authorizations and access control	
CEN-09	Contract upgradeability and re-initialization	

Important onlyProvider privileges are listed below:

```

setProvider
setUnderlyingPrice
setDirectPrice
setFeed

```

Important admin privileges are listed below:

```

initialize
_delegateCompLikeTo
adminOrInitializing
addRewardAddress
setRewardAddress
setComptroller
setAdmin
_setCloseFactor
_setMarketBorrowCaps
_setBorrowCapGuardian
_setMintPaused
_setBorrowPaused
_setTransferPaused
_setSeizePaused
_becomeImplementation
_resignImplementation
_setImplementation
_resignImplementation
_setImplementation

```



pauseGuardian can pause state of following functions. pauseGuardian privileges are listed below

```
_setMintPaused
_setBorrowPaused
_setTransferPaused
_setSeizePaused
```

Important adminOrInitializing privileges are listed below:

```
_setRewardSpeed
updateAndDistributeSupplierRewardsForToken
updateAndDistributeBorrowerRewardsForToken
updateAndDistributeBorrowerRewardsForToken
_grantReward
```

Important borrowCapGuardian privileges are listed below:

```
_setMarketBorrowCaps()
```

Important Unitroller admin privileges are listed below:

```
_become
```

## RECOMMENDATION

Deployers', owners', administrators', and all other privileged roles' private-keys/access-keys/admin-keys should be secured carefully. These entities can have a single point of failure that compromises the security of the project. Manage centralized and privileged roles carefully.

- Implement multi-signature wallets: For wallets with admin privileges and that act as pauseGuardian, borrowCapGuardian, consider converting them into multi-signature wallets. Multi-sig wallet requires multiple parties to agree on transactions before execution, reducing the risk that a single compromised account could perform unauthorized actions.
- Time-lock sensitive functions: Introduce time-locks for critical functions so that any changes have a mandatory delay period during which users can review and react to proposed actions.



Identifier	Definition	Severity
LEN-00	Note regarding possible insolvency	Unknown 🟡

Here are some scenarios which could cause a market to become insolvent:

- If protocol experiences a sudden withdrawal of a significant portion of its liquidity, it may not have enough assets on hand to cover all withdrawals, especially if assets are locked up in loans or other illiquid positions.
- If a malicious actor gains control over a DeFi protocol's governance mechanism (which could be through acquiring a majority of governance tokens), they could make decisions that deplete the protocol's funds or direct assets to themselves.
- If the asset used as collateral loses its value rapidly (due to hacks, or scams), it might become impossible to maintain solvency, especially if the protocol is overly reliant on a particular asset.
- Front-running users' transactions to profit from trades or liquidations can create unfavorable market conditions. If miners or MEV bots consistently extract value from users, it could lead to a loss of trust and a withdrawal of capital.
- The price of the underlying (or borrowed) asset makes a big, quick move during a time of high network congestion — resulting in the market becoming insolvent before enough liquidation transactions can be mined.
- The price oracle temporarily goes offline during a time of high market volatility. This could result in the oracle not updating the asset prices until after the market has become insolvent.
- If price reported by the price oracle such as ChainlinkPriceOracle is inaccurate, it may lead to price manipulation, and liquidation.
- If project team accumulates or buys back underlying asset, and uses it to manipulate price, it may lead to liquidation of collateralized funds. The admin or oracle steals enough collateral that the market becomes insolvent.
- Miners receive enough funds that the market eventually becomes insolvent. (Read COD-11)



- Administrators list a buggy ERC20 token that allows minting of arbitrarily many tokens. This bad token is used as collateral to borrow funds that it never intends to repay.

## RECOMMENDATION

Due to inherent interconnectedness of DeFi systems, complete security for any protocol remains unattainable. There is no foolproof method to entirely prevent aforementioned events from occurring. When such events do occur, features like contract upgradeability and circuit breakers prove invaluable.


Circuit breakers should be implemented to temporarily suspend specific functions in the face of an emergency. However, it is crucial to ensure that these safety mechanisms are governed by reliable and trustworthy entities to maintain the integrity and trust in the protocol.

## ACKNOWLEDGEMENT

Lendora team acknowledged to these possible insolvency scenarios present in protocol, and provided mentioned comments:

- Governance control will be put in place to prevent any single actor from unilaterally controlling the protocol's decisions.
- Price oracle such as ChainlinkPriceOracle is trusted, and utilized by many DeFi protocols.
- Regular audits, bug bounties, continuous monitoring of price oracles and dependencies will be done to ensure the integrity of Lendora Protocol.



Identifier	Definition	Severity
LOG-01	Lack of adequate checks	Minor 

Below mentioned functions are set without any arbitrary boundaries.

`_addReserves` – Allows any arbitrary user to add to the reserves, which may affect interest rate calculations.

Currently cTokens cannot be transferred if they are required to collateralize a loan, but there are no functions to check if users' cTokens are locked. Considering adding a related function to check if users' cTokens are transferable.


`sweepToken` – This function is public. It should be callable by admin only.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL

## RECOMMENDATION

Add appropriate checks to mentioned functions.



Identifier	Definition	Severity
LOG-03	Re-entrancy	Minor 

In functions like `doTransferIn` and `doTransferOut` - non-standard ERC-20 token handling does not revert on false return, however, both functions have require statements that may guarantee the transaction success.

**Scenario:**

`doTransferIn` makes external call `token.transfer`, which may be a point of re-entrancy if token contract is malicious or poorly implemented. Since, there are no state changes being made after external calls, re-entrancy risk is low. Despite reduced risk, the potential for re-entrancy is not completely dismissed, and it is recommended to implement standard re-entrancy protection.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL

**RECOMMENDATION**

Implement standard re-entrancy protection, since these functions do not conform to the ERC-20 return value norms.



Identifier	Definition	Severity
LOG-04	Race condition in the ERC20 approve function	Medium 🟡

CToken contracts, are ERC20 compliant, and are vulnerable to known race condition in the ERC20 approve function. ERC20 approve function race condition vulnerability arises when a user tries to change an already set allowance.

#### Scenario:

An attacker can monitor the pending transactions and, upon noticing an attempt to reduce their allowance, quickly initiate a transaction to spend the remaining allowance before the change is confirmed. If the attacker's transaction is confirmed first, they can spend the initial allowance and then spend the new allowance once the change is confirmed, potentially doubling their spending and stealing tokens.

#### RECOMMENDATION

Before updating the allowance to new amount, first reset it to zero. This should be done even if the new allowance is lower than the previous one. This should stop possible race condition.





Identifier	Definition	Severity
COD-02	Miner manipulation via <code>block.timestamp</code> , <code>block.number</code>	Minor ●
LOG-05	Potential front-running	

**Miner Manipulation:** Miners have control over the order of transactions in a block. They could potentially reorder transactions to benefit themselves or third parties. For instance, if a miner sees a transaction that will significantly affect the price of an asset on a DeFi platform, they could insert their own transaction ahead of it to capitalize on the price movement.

**Front-Running:** Front-running is when an entity sees a pending transaction and quickly gets a similar transaction into a block beforehand to profit from the knowledge of the upcoming trade.

#### Scenario:

When transaction to liquidate an under-collateralized loan is broadcasted, a miner or observer could attempt to front-run this transaction with their own liquidation, profiting from the liquidation bonus.

Actions like changing the admin or implementation can have significant consequences. A front-runner may acquire or dump tokens based on the anticipated reaction to a governance change.

Protocol contract uses price feed oracles, miners or other users could manipulate transaction order to benefit from trades contingent on the timing of the price updates.

#### RECOMMENDATION

Use commit-reveal or similar scheme to hide transactions until validated.



Identifier	Definition	Severity
COD-03	Lack of price update validation	Medium 🟡
COD-04	Oracle manipulation risk	

Lendora protocol allows users to borrow funds, by depositing underlying asset as collateral. The borrowed value is always lower than the amount in collateral. If the collateral value declines, the protocol may sell it to recover the loaned amount.

`setUnderlyingPrice` and `setDirectPrice` functions accept any uint as the new price. There should be sanity checks to prevent erroneous or malicious price updates.


Smart contract allows provider to update the price without any checks against sudden spikes or drops.

## RECOMMENDATION

Implementing a maximum percentage change between price updates can mitigate price manipulation risks.

Trust in price oracles such as `ChainlinkPriceOracle` is required for the protocol to work. Make sure reported price is accurate.



Identifier	Definition	Severity
COD-05	Ether handling	Minor 

doTransferIn function has a sanity check that could potentially be bypassed if from parameter is not properly validated. doTransferIn checks for `msg.value >= amount` instead of strict equality, which may be correct, but CEther does not follow this pattern, potentially allowing discrepancies between the expected and actual transferred value.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL

## RECOMMENDATION

Fix non-conforming logic.



Identifier	Definition
COD-06	Note regarding keccak256 secure hashing

Note that the keccak256 function is not collision-resistant, and therefore there is a possibility of two different messages producing the same hash. Generating strong random input data, and properly securing and managing keys is recommended for fortification of keccak256.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL



Identifier	Definition	Severity
COD-07	Unknown address	Minor ●
COD-08	Hardcoded address	

0xc00e94Cb662C3520282E6f5717214004A7f26888 – Comptroller

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL

## RECOMMENDATION

Interact with trusted contracts and accounts only.



Identifier	Definition	Severity
COD-09	Utilization rate calculation	Medium 🟡


In utilizationRate, if cash + borrows - reserves is less than borrows due to a very low amount of cash or high reserves, this can lead to unexpected behavior because sub will revert if it results in a negative number.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL

## RECOMMENDATION

Ensure that this case is handled or can never occur.



Identifier	Definition	Severity
COD-10	Use of <code>delegatecall</code>	Minor 

`delegatecall` is present, and is used to pass execution to the `comptrollerImplementation`. If there are vulnerabilities in `implementation contract`, a malicious actor can potentially utilize these functions to destroy the logic implementation or execute custom logic.

## RECOMMENDATION

Verify the user input and do not allow contract to perform `delegatecall` calls to untrusted contracts. Use of `delegatecall` in the contract is not recommended, as managing the storage layout in multiple contracts during logic upgrade can be disruptive.



Identifier	Definition	Severity
COD-11	Interest free loan	Major 🟡

In CToken contract, it calculates the borrow balance of an account in function `borrowBalanceStoredInternal`. The balance is the principal scaled by the ratio of the borrow indices

However, when principal and ratio of borrow indices are both small – the result can equal the principal, due to automatic truncation of division within solidity. Meaning a loan could accrue no actual interest, yet still be factored into calculations of `totalReserves`, `totalBorrows`, and `exchangeRates`. In case of many small loans being taken out, the associated interest calculated for that market may not match the amount actually received when users pay off those loans.


It is possible for users to take out small, short-term, interest-free loans. Additionally, users can resupply the borrowed assets back into protocol to receive interest.

## RECOMMENDATION

Consider the case of automatic truncation of division in this `borrowBalanceStoredInternal` function. Fix non-conforming logic.





Identifier	Definition	Severity
COD-12	Lack of event-driven architecture	Minor 


Smart contracts use function calls to update state, which can make it difficult to track and analyze changes to the contract over time.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT

## RECOMMENDATION

Use events to track state changes. Events improve transparency and provide a more granular view of contract activity.



Identifier	Definition	Severity
COD-13	Assembly code	Minor 

Inline assembly is a way to access the Ethereum Virtual Machine (EVM) at low level. This bypasses several important safety features and checks of Solidity. Moreover, automated and manual checks are not confidently possible for inline assembly codes.

Below mentioned functions use inline assembly codes:

```
doTransferIn
doTransferOut
delegateTo
delegateToViewImplementation
function () external payable
```

## RECOMMENDATION

Use high level Solidity constructs instead.



Identifier	Definition	Severity
COD-14	Note regarding flash loan vulnerabilities	Unknown 🟤


Smart contracts are not directly susceptible to flash loan attacks, which usually exploit some form of arbitrage opportunity. However, when smart contracts interact with malicious contracts, technically flash loan vulnerabilities can be introduced. For example, when “approved” underlying asset contract turns out to be a malicious contract, it can be used to introduce flash-loan vulnerabilities.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT

## NOTE

Due to the interconnected nature of DeFi contracts, smart contracts can be exploited using another malicious contract. Be cautious while interacting with third-party contracts, tokens, and protocols.



Identifier	Definition	Severity
COD-15	Excessive indirection	Minor 

Exponential contract represents a fixed-size decimal number with a `uint` that is scaled up so the smallest non-zero decimal value is internally represented by the number 1. However, it is also unnecessarily wrapped in a struct. Many operations are complicated by mapping back and forth between the two representations.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL

## NOTE

Consider using the `uint256` type to internally represent the fixed-size decimal numbers.

Consider enforcing Mantissa suffix convention to consistently indicate whether a given variable is scaled.



Identifier	Definition	
COM-01	Multiple pragma directives	
COM-02	Floating pragma	

Various compilers and floating pragma are used across all contracts.

TERFI  
CONFIDENTIALINTERFI  
CONFIDENTIAL

## RECOMMENDATION

Pragma should be fixed to the version that you're indenting to deploy your contracts with.



Identifier	Definition	Severity
COM-02	Outdated compiler version	Major 🟡

Compilers are set to outdated version. Using an outdated Solidity compilers can lead to security vulnerabilities due to unpatched bugs, higher gas costs from less optimized code, and potential compatibility issues.

```
pragma solidity ^0.5.16;
```






INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL

## RECOMMENDATION

Set Compiler to version 0.8.12 or above.



## AUTOMATED ANALYSIS

Symbol	Definition
	Function modifies state
	Function is payable
	Function is internal
	Function is private
	Function is important

```

| **ControllerInterface** | Implementation |   | | |
| L | enterMarkets | External ! |  | NO ! |
| L | exitMarket | External ! |  | NO ! |
| L | mintAllowed | External ! |  | NO ! |
| L | mintVerify | External ! |  | NO ! |
| L | redeemAllowed | External ! |  | NO ! |
| L | redeemVerify | External ! |  | NO ! |
| L | borrowAllowed | External ! |  | NO ! |
| L | borrowVerify | External ! |  | NO ! |
| L | repayBorrowAllowed | External ! |  | NO ! |
| L | repayBorrowVerify | External ! |  | NO ! |
| L | liquidateBorrowAllowed | External ! |  | NO ! |
| L | liquidateBorrowVerify | External ! |  | NO ! |
| L | seizeAllowed | External ! |  | NO ! |
| L | seizeVerify | External ! |  | NO ! |
| L | transferAllowed | External ! |  | NO ! |
| L | transferVerify | External ! |  | NO ! |
| L | liquidateCalculateSeizeTokens | External ! |   | NO ! |
| | | | |
| **CTokenStorage** | Implementation |   |

```



|||||

| **\*\*CTokenInterface\*\*** | Implementation | CTokenStorage |||| <sup>L</sup> | transfer | External ! | ● |NO! || <sup>L</sup> | transferFrom | External ! | ● |NO! || <sup>L</sup> | approve | External ! | ● |NO! || <sup>L</sup> | allowance | External ! | |NO! || <sup>L</sup> | balanceOf | External ! | |NO! || <sup>L</sup> | balanceOfUnderlying | External ! | ● |NO! || <sup>L</sup> | getAccountSnapshot | External ! | |NO! || <sup>L</sup> | borrowRatePerBlock | External ! | |NO! || <sup>L</sup> | supplyRatePerBlock | External ! | |NO! || <sup>L</sup> | totalBorrowsCurrent | External ! | ● |NO! || <sup>L</sup> | borrowBalanceCurrent | External ! | ● |NO! || <sup>L</sup> | borrowBalanceStored | Public ! | |NO! || <sup>L</sup> | exchangeRateCurrent | Public ! | ● |NO! || <sup>L</sup> | exchangeRateStored | Public ! | |NO! || <sup>L</sup> | getCash | External ! | |NO! || <sup>L</sup> | accrueInterest | Public ! | ● |NO! || <sup>L</sup> | seize | External ! | ● |NO! || <sup>L</sup> | \_setPendingAdmin | External ! | ● |NO! || <sup>L</sup> | \_acceptAdmin | External ! | ● |NO! || <sup>L</sup> | \_setComptroller | Public ! | ● |NO! || <sup>L</sup> | \_setReserveFactor | External ! | ● |NO! || <sup>L</sup> | \_reduceReserves | External ! | ● |NO! || <sup>L</sup> | \_setInterestRateModel | Public ! | ● |NO! |

|||||

| **\*\*CErc20Storage\*\*** | Implementation | |||

|||||

| **\*\*CErc20Interface\*\*** | Implementation | CErc20Storage |||| <sup>L</sup> | mint | External ! | ● |NO! || <sup>L</sup> | redeem | External ! | ● |NO! || <sup>L</sup> | redeemUnderlying | External ! | ● |NO! |



```

|  L | borrow | External ! | ● |NO! |
|  L | repayBorrow | External ! | ● |NO! |
|  L | repayBorrowBehalf | External ! | ● |NO! |
|  L | liquidateBorrow | External ! | ● |NO! |
|  L | _addReserves | External ! | ● |NO! |
|||||
| **CDelegationStorage** | Implementation | |||
|||||
| **CDelegatorInterface** | Implementation | CDelegationStorage |||
|  L | _setImplementation | Public ! | ● |NO! |
|||||
| **CDelegateInterface** | Implementation | CDelegationStorage |||
|  L | _becomeImplementation | Public ! | ● |NO! |
|  L | _resignImplementation | Public ! | ● |NO! |
|||||
| **ComptrollerErrorReporter** | Implementation | |||
|  L | fail | Internal 🔒 | ● | |
|  L | failOpaque | Internal 🔒 | ● | |
|||||
| **TokenErrorReporter** | Implementation | |||
|  L | fail | Internal 🔒 | ● | |
|  L | failOpaque | Internal 🔒 | ● | |
|||||
| **CarefulMath** | Implementation | |||
|  L | mulUInt | Internal 🔒 | | |
|  L | divUInt | Internal 🔒 | | |
|  L | subUInt | Internal 🔒 | | |
|  L | addUInt | Internal 🔒 | | |
|  L | addThenSubUInt | Internal 🔒 | | |
|||||
| **ExponentialNoError** | Implementation | |||
|  L | truncate | Internal 🔒 | | |

```





```

| L | div_ | Internal 🔒 | | |
| L | fraction | Internal 🔒 | | |
|||||
| **Exponential** | Implementation | CarefulMath, ExponentialNoError |||
| L | getExp | Internal 🔒 | | |
| L | addExp | Internal 🔒 | | |
| L | subExp | Internal 🔒 | | |
| L | mulScalar | Internal 🔒 | | |
| L | mulScalarTruncate | Internal 🔒 | | |
| L | mulScalarTruncateAddUInt | Internal 🔒 | | |
| L | divScalar | Internal 🔒 | | |
| L | divScalarByExp | Internal 🔒 | | |
| L | divScalarByExpTruncate | Internal 🔒 | | |
| L | mulExp | Internal 🔒 | | |
| L | mulExp | Internal 🔒 | | |
| L | mulExp3 | Internal 🔒 | | |
| L | divExp | Internal 🔒 | | |
|||||
| **EIP20Interface** | Interface | |||
| L | name | External ! | |NO! |
| L | symbol | External ! | |NO! |
| L | decimals | External ! | |NO! |
| L | totalSupply | External ! | |NO! |
| L | balanceOf | External ! | |NO! |
| L | transfer | External ! | 🚫 |NO! |
| L | transferFrom | External ! | 🚫 |NO! |
| L | approve | External ! | 🚫 |NO! |
| L | allowance | External ! | |NO! |
|||||
| **EIP20NonStandardInterface** | Interface | |||
| L | totalSupply | External ! | |NO! |
| L | balanceOf | External ! | |NO! |

```



```

| L | transfer | External ! | ● | NO ! |
| L | transferFrom | External ! | ● | NO ! |
| L | approve | External ! | ● | NO ! |
| L | allowance | External ! | | NO ! |
|||||
| **InterestRateModel** | Implementation | |||
| L | getBorrowRate | External ! | | NO ! |
| L | getSupplyRate | External ! | | NO ! |
|||||
| **CToken** | Implementation | CTokenInterface, Exponential, TokenErrorReporter |||
| L | initialize | Public ! | ● | NO ! |
| L | transferTokens | Internal 🔒 | ● | |
| L | transfer | External ! | ● | nonReentrant |
| L | transferFrom | External ! | ● | nonReentrant |
| L | approve | External ! | ● | NO ! |
| L | allowance | External ! | | NO ! |
| L | balanceOf | External ! | | NO ! |
| L | balanceOfUnderlying | External ! | ● | NO ! |
| L | getAccountSnapshot | External ! | | NO ! |
| L | getBlockNumber | Internal 🔒 | | |
| L | borrowRatePerBlock | External ! | | NO ! |
| L | supplyRatePerBlock | External ! | | NO ! |
| L | totalBorrowsCurrent | External ! | ● | nonReentrant |
| L | borrowBalanceCurrent | External ! | ● | nonReentrant |
| L | borrowBalanceStored | Public ! | | NO ! |
| L | borrowBalanceStoredInternal | Internal 🔒 | | |
| L | exchangeRateCurrent | Public ! | ● | nonReentrant |
| L | exchangeRateStored | Public ! | | NO ! |
| L | exchangeRateStoredInternal | Internal 🔒 | | |
| L | getCash | External ! | | NO ! |
| L | accrueInterest | Public ! | ● | NO ! |

```



```

| L | mintInternal | Internal | 🔒 | 🔴 | nonReentrant |
| L | mintFresh | Internal | 🔒 | 🔴 | |
| L | redeemInternal | Internal | 🔒 | 🔴 | nonReentrant |
| L | redeemUnderlyingInternal | Internal | 🔒 | 🔴 | nonReentrant |
| L | redeemFresh | Internal | 🔒 | 🔴 | |
| L | borrowInternal | Internal | 🔒 | 🔴 | nonReentrant |
| L | borrowFresh | Internal | 🔒 | 🔴 | |
| L | repayBorrowInternal | Internal | 🔒 | 🔴 | nonReentrant |
| L | repayBorrowBehalfInternal | Internal | 🔒 | 🔴 | nonReentrant |
| L | repayBorrowFresh | Internal | 🔒 | 🔴 | |
| L | liquidateBorrowInternal | Internal | 🔒 | 🔴 | nonReentrant |
| L | liquidateBorrowFresh | Internal | 🔒 | 🔴 | |
| L | seize | External | ! | 🔴 | nonReentrant |
| L | seizeInternal | Internal | 🔒 | 🔴 | |
| L | _setPendingAdmin | External | ! | 🔴 | NO! |
| L | _acceptAdmin | External | ! | 🔴 | NO! |
| L | _setComptroller | Public | ! | 🔴 | NO! |
| L | _setReserveFactor | External | ! | 🔴 | nonReentrant |
| L | _setReserveFactorFresh | Internal | 🔒 | 🔴 | |
| L | _addReservesInternal | Internal | 🔒 | 🔴 | nonReentrant |
| L | _addReservesFresh | Internal | 🔒 | 🔴 | |
| L | _reduceReserves | External | ! | 🔴 | nonReentrant |
| L | _reduceReservesFresh | Internal | 🔒 | 🔴 | |
| L | _setInterestRateModel | Public | ! | 🔴 | NO! |
| L | _setInterestRateModelFresh | Internal | 🔒 | 🔴 | |
| L | getCashPrior | Internal | 🔒 | | |
| L | doTransferIn | Internal | 🔒 | 🔴 | |
| L | doTransferOut | Internal | 🔒 | 🔴 | |
|||||
| **PriceOracle** | Implementation | |||
| L | getUnderlyingPrice | External | ! | | NO! |

```



|||||

| **\*\*CompLike\*\*** | Interface | |||| <sup>L</sup> | delegate | External ! | ● | NO ! |

|||||

| **\*\*CErc20\*\*** | Implementation | CToken, CErc20Interface |||| <sup>L</sup> | initialize | Public ! | ● | NO ! || <sup>L</sup> | mint | External ! | ● | NO ! || <sup>L</sup> | redeem | External ! | ● | NO ! || <sup>L</sup> | redeemUnderlying | External ! | ● | NO ! || <sup>L</sup> | borrow | External ! | ● | NO ! || <sup>L</sup> | repayBorrow | External ! | ● | NO ! || <sup>L</sup> | repayBorrowBehalf | External ! | ● | NO ! || <sup>L</sup> | liquidateBorrow | External ! | ● | NO ! || <sup>L</sup> | sweepToken | External ! | ● | NO ! || <sup>L</sup> | \_addReserves | External ! | ● | NO ! || <sup>L</sup> | getCashPrior | Internal 🔒 | | || <sup>L</sup> | doTransferIn | Internal 🔒 | ● | || <sup>L</sup> | doTransferOut | Internal 🔒 | ● | || <sup>L</sup> | \_delegateCompLikeTo | External ! | ● | NO ! |

|||||

| **\*\*BasePriceOracle\*\*** | Implementation | PriceOracle |||| <sup>L</sup> | <Constructor> | Public ! | ● | NO ! || <sup>L</sup> | setProvider | Public ! | ● | onlyProvider || <sup>L</sup> | \_getUnderlyingAddress | Internal 🔒 | | || <sup>L</sup> | getUnderlyingPrice | External ! | | NO ! || <sup>L</sup> | setUnderlyingPrice | Public ! | ● | onlyProvider || <sup>L</sup> | setDirectPrice | Public ! | ● | onlyProvider || <sup>L</sup> | assetPrices | External ! | | NO ! || <sup>L</sup> | compareStrings | Internal 🔒 | | |

|||||

| **\*\*IAggregatorV2V3Interface\*\*** | Interface | |||| <sup>L</sup> | decimals | External ! | | NO ! |

```

|  L | latestRoundData | External ! | |NO! |
|||||
| **ChainlinkPriceOracle** | Implementation | BasePriceOracle |||
|  L | getUnderlyingPrice | External ! | |NO! |
|  L | _getChainlinkPrice | Internal 🔒 | | |
|  L | getChainlinkPrice | External ! | |NO! |
|  L | setFeed | External ! | 🔴 | onlyProvider |

```

```

| **CEther** | Implementation | CToken ||| |
|  L | <Constructor> | Public ! | 🔴 |NO! |
|  L | mint | External ! | 🏧 |NO! |
|  L | redeem | External ! | 🔴 |NO! |
|  L | redeemUnderlying | External ! | 🔴 |NO! |
|  L | borrow | External ! | 🔴 |NO! |
|  L | repayBorrow | External ! | 🏧 |NO! |
|  L | repayBorrowBehalf | External ! | 🏧 |NO! |
|  L | liquidateBorrow | External ! | 🏧 |NO! |
|  L | <Fallback> | External ! | 🏧 |NO! |
|  L | getCashPrior | Internal 🔒 | | |
|  L | doTransferIn | Internal 🔒 | 🔴 | | |
|  L | doTransferOut | Internal 🔒 | 🔴 | | |
|  L | requireNoError | Internal 🔒 | | |

```

```

| **Unitroller** | Implementation | UnitrollerAdminStorage, ComptrollerErrorReporter |||
|  L | <Constructor> | Public ! | 🔴 |NO! |
|  L | _setPendingImplementation | Public ! | 🔴 |NO! |
|  L | _acceptImplementation | Public ! | 🔴 |NO! |
|  L | _setPendingAdmin | Public ! | 🔴 |NO! |
|  L | _acceptAdmin | Public ! | 🔴 |NO! |
|  L | <Fallback> | External ! | 🏧 |NO! |
|||||
| **IComptroller** | Interface | |||

```



```

| L | isMarketListed | External ! | |NO! |
| L | getAllMarkets | External ! | |NO! |
|||||
| **RewardDistributorStorage** | Implementation | |||
|||||
| **RewardDistributor** | Implementation | RewardDistributorStorage, Exponential |||
| L | <Constructor> | Public ! | ● |NO! |
| L | initialize | Public ! | ● |NO! |
| L | adminOrInitializing | Internal 🔒 | | |
| L | _setRewardSpeed | Public ! | ● |NO! |
| L | setRewardSpeedInternal | Internal 🔒 | ● | |
| L | updateRewardSupplyIndex | Internal 🔒 | ● | |
| L | updateRewardBorrowIndex | Internal 🔒 | ● | |
| L | distributeSupplierReward | Internal 🔒 | ● | |
| L | distributeBorrowerReward | Internal 🔒 | ● | |
| L | updateAndDistributeSupplierRewardsForToken | External ! | ● |NO! |
| L | updateAndDistributeBorrowerRewardsForToken | External ! | ● |NO! |
| L | updateAndDistributeBorrowerRewardsForToken | External ! | ● |NO! |
| L | claimReward | Public ! | ● |NO! |
| L | claimReward | Public ! | ● |NO! |
| L | claimReward | Public ! | 🗑️ |NO! |
| L | grantRewardInternal | Internal 🔒 | ● | |
| L | _grantReward | Public ! | ● |NO! |
| L | addRewardAddress | Public ! | ● |NO! |
| L | getRewardAddress | Public ! | |NO! |
| L | getRewardAddressLength | External ! | |NO! |
| L | setRewardAddress | Public ! | ● |NO! |
| L | setComptroller | Public ! | ● |NO! |
| L | setAdmin | Public ! | ● |NO! |
| L | <Fallback> | External ! | 🗑️ |NO! |
| L | getBlockTimestamp | Public ! | |NO! |
|||||

```





| **\*\*Comptroller\*\*** | Implementation | ComptrollerV9Storage, ComptrollerInterface, ComptrollerErrorReporter, ExponentialNoError |||

L	<Constructor>	Public	!	●	NO!	
L	getAssetsIn	External	!		NO!	
L	checkMembership	External	!		NO!	
L	enterMarkets	Public	!	●	NO!	
L	addToMarketInternal	Internal		●		
L	exitMarket	External	!	●	NO!	
L	mintAllowed	External	!	●	NO!	
L	mintVerify	External	!	●	NO!	
L	redeemAllowed	External	!	●	NO!	
L	redeemAllowedInternal	Internal				
L	redeemVerify	External	!	●	NO!	
L	borrowAllowed	External	!	●	NO!	
L	borrowVerify	External	!	●	NO!	
L	repayBorrowAllowed	External	!	●	NO!	
L	repayBorrowVerify	External	!	●	NO!	
L	liquidateBorrowAllowed	External	!	●	NO!	
L	liquidateBorrowVerify	External	!	●	NO!	
L	seizeAllowed	External	!	●	NO!	
L	seizeVerify	External	!	●	NO!	
L	transferAllowed	External	!	●	NO!	
L	transferVerify	External	!	●	NO!	
L	getAccountLiquidity	Public	!		NO!	
L	getAccountLiquidityInternal	Internal				
L	getHypotheticalAccountLiquidity	Public	!		NO!	
L	getHypotheticalAccountLiquidityInternal	Internal				
L	liquidateCalculateSeizeTokens	External	!		NO!	
L	_setRewardDistributor	Public	!	●	NO!	
L	_setPriceOracle	Public	!	●	NO!	
L	_setCloseFactor	External	!	●	NO!	
L	_setCollateralFactor	External	!	●	NO!	



```

| L | _setLiquidationIncentive | External ! | 🔴 | NO ! | |
| L | _supportMarket | External ! | 🔴 | NO ! |
| L | _addMarketInternal | Internal 🔒 | 🔴 | | |
| L | _initializeMarket | Internal 🔒 | 🔴 | | |
| L | _setMarketBorrowCaps | External ! | 🔴 | NO ! |
| L | _setBorrowCapGuardian | External ! | 🔴 | NO ! |
| L | _setPauseGuardian | Public ! | 🔴 | NO ! |
| L | _setMintPaused | Public ! | 🔴 | NO ! |
| L | _setBorrowPaused | Public ! | 🔴 | NO ! |
| L | _setTransferPaused | Public ! | 🔴 | NO ! |
| L | _setSeizePaused | Public ! | 🔴 | NO ! |
| L | _become | Public ! | 🔴 | NO ! |
| L | adminOrInitializing | Internal 🔒 | | |
| L | getAllMarkets | Public ! | | NO ! |
| L | getBlockTimestamp | Public ! | | NO ! |
| L | getCompAddress | Public ! | | NO ! |
| L | isMarketListed | External ! | | NO ! |

```

```

| **SafeMath** | Library | |||
| L | add | Internal 🔒 | | |
| L | add | Internal 🔒 | | |
| L | sub | Internal 🔒 | | |
| L | sub | Internal 🔒 | | |
| L | mul | Internal 🔒 | | |
| L | mul | Internal 🔒 | | |
| L | div | Internal 🔒 | | |
| L | div | Internal 🔒 | | |
| L | mod | Internal 🔒 | | |
| L | mod | Internal 🔒 | | |
| |||||

```

```

| **JumpRateModelV2** | Implementation | InterestRateModel |||

```



```

| L | <Constructor> | Public ! | ● |NO! |
| L | utilizationRate | Public ! | |NO! |
| L | getBorrowRate | Public ! | |NO! |
| L | getSupplyRate | Public ! | |NO! |

```

```

| **CErc20Delegate** | Implementation | CErc20, CDelegateInterface |||

```

```

| L | <Constructor> | Public ! | ● |NO! |
| L | _becomeImplementation | Public ! | ● |NO! |
| L | _resignImplementation | Public ! | ● |NO! |

```

```

|||||

```

```

| **CErc20Delegator** | Implementation | CTokenInterface, CErc20Interface,
CDelegatorInterface |||

```

```

| L | <Constructor> | Public ! | ● |NO! |
| L | _setImplementation | Public ! | ● |NO! |
| L | mint | External ! | ● |NO! |
| L | redeem | External ! | ● |NO! |
| L | redeemUnderlying | External ! | ● |NO! |
| L | borrow | External ! | ● |NO! |
| L | repayBorrow | External ! | ● |NO! |
| L | repayBorrowBehalf | External ! | ● |NO! |
| L | liquidateBorrow | External ! | ● |NO! |
| L | transfer | External ! | ● |NO! |
| L | transferFrom | External ! | ● |NO! |
| L | approve | External ! | ● |NO! |
| L | allowance | External ! | |NO! |
| L | balanceOf | External ! | |NO! |
| L | balanceOfUnderlying | External ! | ● |NO! |
| L | getAccountSnapshot | External ! | |NO! |
| L | borrowRatePerBlock | External ! | |NO! |
| L | supplyRatePerBlock | External ! | |NO! |
| L | totalBorrowsCurrent | External ! | ● |NO! |

```



```

| L | borrowBalanceCurrent | External ! | ● | NO ! |
| L | borrowBalanceStored | Public ! | | NO ! |
| L | exchangeRateCurrent | Public ! | ● | NO ! |
| L | exchangeRateStored | Public ! | | NO ! |
| L | getCash | External ! | | NO ! |
| L | accrueInterest | Public ! | ● | NO ! |
| L | seize | External ! | ● | NO ! |
| L | sweepToken | External ! | ● | NO ! |
| L | _setPendingAdmin | External ! | ● | NO ! |
| L | _setComptroller | Public ! | ● | NO ! |
| L | _setReserveFactor | External ! | ● | NO ! |
| L | _acceptAdmin | External ! | ● | NO ! |
| L | _addReserves | External ! | ● | NO ! |
| L | _reduceReserves | External ! | ● | NO ! |
| L | _setInterestRateModel | Public ! | ● | NO ! |
| L | delegateTo | Internal 🏠 | ● | |
| L | delegateToImplementation | Public ! | ● | NO ! |
| L | delegateToViewImplementation | Public ! | | NO ! |
| L | <Fallback> | External ! | 🏠 | NO ! |

|||||

| **CErc20Immutable** | Implementation | CErc20 |||
| L | <Constructor> | Public ! | ● | NO ! |

|||||

| **CEtherDelegate** | Implementation | CToken, CDelegateInterface |||
| L | <Constructor> | Public ! | ● | NO ! |
| L | _becomeImplementation | Public ! | ● | NO ! |
| L | _resignImplementation | Public ! | ● | NO ! |
| L | initialize | Public ! | ● | NO ! |
| L | mint | External ! | 🏠 | NO ! |
| L | redeem | External ! | ● | NO ! |
| L | redeemUnderlying | External ! | ● | NO ! |

```



```

| L | borrow | External ! | 🔴 | NO ! | |
| L | repayBorrow | External ! | 🟡 | NO ! |
| L | repayBorrowBehalf | External ! | 🟡 | NO ! |
| L | liquidateBorrow | External ! | 🟡 | NO ! |
| L | <Fallback> | External ! | 🟡 | NO ! |
| L | getCashPrior | Internal 🔒 | | |
| L | doTransferIn | Internal 🔒 | 🔴 | | |
| L | doTransferOut | Internal 🔒 | 🔴 | | |
| L | requireNoError | Internal 🔒 | | |

```

```

|||||

```

```

| **CEtherInterface** | Implementation | CErc20Storage |||

```

```

| L | mint | External ! | 🟡 | NO ! |
| L | redeem | External ! | 🔴 | NO ! |
| L | redeemUnderlying | External ! | 🔴 | NO ! |
| L | borrow | External ! | 🔴 | NO ! |
| L | repayBorrow | External ! | 🟡 | NO ! |
| L | repayBorrowBehalf | External ! | 🟡 | NO ! |
| L | liquidateBorrow | External ! | 🟡 | NO ! |
| L | _addReserves | External ! | 🔴 | NO ! |

```

```

|||||

```

```

| **CEtherDelegator** | Implementation | CTokenInterface, CEtherInterface,
CDelegatorInterface |||

```

```

| L | <Constructor> | Public ! | 🔴 | NO ! |
| L | _setImplementation | Public ! | 🔴 | NO ! |
| L | mint | External ! | 🔴 | NO ! |
| L | mint | External ! | 🟡 | NO ! |
| L | redeem | External ! | 🔴 | NO ! |
| L | redeemUnderlying | External ! | 🔴 | NO ! |
| L | borrow | External ! | 🔴 | NO ! |
| L | repayBorrow | External ! | 🟡 | NO ! |
| L | repayBorrowBehalf | External ! | 🟡 | NO ! |
| L | liquidateBorrow | External ! | 🟡 | NO ! |

```



L	transfer	External	!	●	NO!	
L	transferFrom	External	!	●	NO!	
L	approve	External	!	●	NO!	
L	allowance	External	!		NO!	
L	balanceOf	External	!		NO!	
L	balanceOfUnderlying	External	!	●	NO!	
L	getAccountSnapshot	External	!		NO!	
L	borrowRatePerBlock	External	!		NO!	
L	supplyRatePerBlock	External	!		NO!	
L	totalBorrowsCurrent	External	!	●	NO!	
L	borrowBalanceCurrent	External	!	●	NO!	
L	borrowBalanceStored	Public	!		NO!	
L	exchangeRateCurrent	Public	!	●	NO!	
L	exchangeRateStored	Public	!		NO!	
L	getCash	External	!		NO!	
L	accrueInterest	Public	!	●	NO!	
L	seize	External	!	●	NO!	
L	sweepToken	External	!	●	NO!	
L	_setPendingAdmin	External	!	●	NO!	
L	_setComptroller	Public	!	●	NO!	
L	_setReserveFactor	External	!	●	NO!	
L	_acceptAdmin	External	!	●	NO!	
L	_addReserves	External	!	●	NO!	
L	_reduceReserves	External	!	●	NO!	
L	_setInterestRateModel	Public	!	●	NO!	
L	delegateTo	Internal	🔒	●		
L	delegateToImplementation	Public	!	●	NO!	
L	delegateToViewImplementation	Public	!		NO!	
L	<Fallback>	External	!	👤	NO!	



## DISCLAIMERS

InterFi Network provides the easy-to-understand audit of solidity source codes (commonly known as smart contracts).

The smart contract for this particular audit was analyzed for common contract vulnerabilities, and centralization exploits. This audit report makes no statements or warranties on the security of the code. This audit report does not provide any warranty or guarantee regarding the absolute bug-free nature of the smart contract analyzed, nor do they provide any indication of the client's business, business model or legal compliance. This audit report does not extend to the compiler layer, any other areas beyond the programming language, or other programming aspects that could present security risks. Cryptographic tokens are emergent technologies, they carry high levels of technical risks and uncertainty. You agree that your access and/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. This audit report could include false positives, false negatives, and other unpredictable results.

## CONFIDENTIALITY

This report is subject to the terms and conditions (including without limitations, description of services, confidentiality, disclaimer and limitation of liability) outlined in the scope of the audit provided to the client. This report should not be transmitted, disclosed, referred to, or relied upon by any individual for any purpose without InterFi Network's prior written consent.

## NO FINANCIAL ADVICE

This audit report does not indicate the endorsement of any particular project or team, nor guarantees its security. No third party should rely on the reports in any way, including to make any decisions to buy or sell a product, service or any other asset. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. This audit report should not be used in any way



to make decisions around investment or involvement. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort.

FOR AVOIDANCE OF DOUBT, SERVICES, INCLUDING ANY ASSOCIATED AUDIT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

## **TECHNICAL DISCLAIMER**

ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, INTERFI NETWORK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO SERVICES, AUDIT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, INTERFI NETWORK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM THE COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

WITHOUT LIMITING THE FOREGOING, INTERFI NETWORK MAKES NO WARRANTY OF ANY KIND THAT ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET THE CLIENT'S OR ANY OTHER INDIVIDUAL'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE.

## **TIMELINESS OF CONTENT**

The content contained in this audit report is subject to change without any prior notice. InterFi Network does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following the publication.





## LINKS TO OTHER WEBSITES

This audit report provides, through hypertext or other computer links, access to websites and social accounts operated by individuals other than InterFi Network. Such hyperlinks are provided for your reference and convenience only and are the exclusive responsibility of such websites' and social accounts' owners. You agree that InterFi Network is not responsible for the content or operation of such websites and social accounts and that InterFi Network shall have no liability to you or any other person or entity for the use of third-party websites and social accounts. You are solely responsible for determining the extent to which you may use any content at any other websites and social accounts to which you link from the report.

INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI INTERFI  
CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL AUDIT REPORT CONFIDENTIAL



## ABOUT INTERFI NETWORK

InterFi Network provides intelligent blockchain solutions. We provide solidity development, testing, and auditing services. We have developed 150+ solidity codes, audited 1000+ smart contracts, and analyzed 500,000+ code lines. We have worked on major public blockchains e.g., Ethereum, Binance, Cronos, Doge, Polygon, Avalanche, Metis, Fantom, Bitcoin Cash, Velas, Oasis, etc.

InterFi Network is built by engineers, developers, UI experts, and blockchain enthusiasts. Our team currently consists of 4 core members, and 6+ casual contributors.

Website: <https://interfi.network>

Email: [hello@interfi.network](mailto:hello@interfi.network)

GitHub: <https://github.com/interfinetwork>


Telegram (Engineering): <https://t.me/interfiaudits>

Telegram (Onboarding): <https://t.me/interfisupport>



 interfinetwork

 hello@interfi.network

 <https://interfi.network>

SMART CONTRACT AUDITS | SOLIDITY DEVELOPMENT AND TESTING  
RELENTLESSLY SECURING PUBLIC AND PRIVATE BLOCKCHAINS