// HALBORN

Lodestar - Lodestar Finance

Smart Contract Security Assessment

Prepared by: **Halborn**

Date of Engagement: **May 15th, 2023 - June 26th, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	05/15/2023	Roberto Reigada
0.2	Document Updates	06/23/2023	Roberto Reigada
0.3	Document Updates	06/23/2023	Roberto Reigada
0.4	Document Draft Review	06/23/2023	Gokberk Gulgun
0.5	Document Draft Review	06/23/2023	Gabi Urrutia
1.0	Remediation Plan	07/14/2023	Roberto Reigada
1.1	Remediation Plan Review	07/14/2023	Gokberk Gulgun
1.2	Remediation Plan Review	07/14/2023	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Gokberk Gulgun	Halborn	Gokberk.Gulgun@halborn.com
Roberto Reigada	Halborn	Roberto.Reigada@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Lodestar engaged Halborn to conduct a security assessment on their smart contracts beginning on May 15th, 2023 and ending on June 26th, 2023. The security assessment was scoped to the smart contracts provided in the following GitHub repositories:

- Lodestar-Finance/lodestar-protocol.
- Lodestar-Finance/plvglp-oracle.
- LodestarFinance/lodestar-token-fix.
- Lodestar-Finance/plvGLP-looper.

1.2 ASSESSMENT SUMMARY

The team at Halborn was provided six weeks for the engagement and assigned a full-time security engineer to verify the security of the smart contracts. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this assessment is to:

- Ensure that smart contract functions operate as intended.
- Identify potential security issues with the smart contracts.

In summary, Halborn identified some security risks that were mostly addressed by the Lodestar team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard

to the scope of this assessment. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the code and can quickly identify items that do not follow the security best practices. The following phases and associated tools were used during the assessment:

- Research into architecture and purpose.
- Smart contract manual code review and walkthrough.
- Graphing out functionality and contract logic/connectivity/functions. (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes.
- Manual testing by custom scripts.
- Scanning of solidity files for vulnerabilities, security hot-spots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment. (Foundry)

2. RISK METHODOLOGY

Every vulnerability and issue observed by Halborn is ranked based on **two sets** of **Metrics** and a **Severity Coefficient**. This system is inspired by the industry standard Common Vulnerability Scoring System.

The two **Metric sets** are: **Exploitability** and **Impact**. **Exploitability** captures the ease and technical means by which vulnerabilities can be exploited and **Impact** describes the consequences of a successful exploit.

The **Severity Coefficients** is designed to further refine the accuracy of the ranking with two factors: **Reversibility** and **Scope**. These capture the impact of the vulnerability on the environment as well as the number of users and smart contracts affected.

The final score is a value between 0-10 rounded up to 1 decimal place and 10 corresponding to the highest security risk. This provides an objective and accurate rating of the severity of security vulnerabilities in smart contracts.

The system is designed to assist in identifying and prioritizing vulnerabilities based on their level of risk to address the most critical issues in a timely manner.

2.1 EXPLOITABILITY

Attack Origin (AO):

Captures whether the attack requires compromising a specific account.

Attack Cost (AC):

Captures the cost of exploiting the vulnerability incurred by the attacker relative to sending a single transaction on the relevant blockchain. Includes but is not limited to financial and computational cost.

Attack Complexity (AX):

Describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability. Includes but is not limited to macro situation, available third-party liquidity and regulatory challenges.

Metrics:

Exploitability Metric (m_E)	Metric Value	Numerical Value
Attack Origin (AO)	Arbitrary (AO:A) Specific (AO:S)	1 0.2
Attack Cost (AC)	Low (AC:L) Medium (AC:M) High (AC:H)	1 0.67 0.33
Attack Complexity (AX)	Low (AX:L) Medium (AX:M) High (AX:H)	1 0.67 0.33

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

2.2 IMPACT

Confidentiality (C):

Measures the impact to the confidentiality of the information resources managed by the contract due to a successfully exploited vulnerability. Confidentiality refers to limiting access to authorized users only.

Integrity (I):

Measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of data stored and/or processed on-chain. Integrity impact directly affecting Deposit or Yield records is excluded.

Availability (A):

Measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. This metric refers to smart contract features and functionality, not state. Availability impact directly affecting Deposit or Yield is excluded.

Deposit (D):

Measures the impact to the deposits made to the contract by either users or owners.

Yield (Y):

Measures the impact to the yield generated by the contract for either users or owners.

Metrics:

Impact Metric (m_I)	Metric Value	Numerical Value
Confidentiality (C)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Integrity (I)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Availability (A)	None (A:N)	0
	Low (A:L)	0.25
	Medium (A:M)	0.5
	High (A:H)	0.75
	Critical	1
Deposit (D)	None (D:N)	0
	Low (D:L)	0.25
	Medium (D:M)	0.5
	High (D:H)	0.75
	Critical (D:C)	1
Yield (Y)	None (Y:N)	0
	Low (Y:L)	0.25
	Medium: (Y:M)	0.5
	High: (Y:H)	0.75
	Critical (Y:H)	1

Impact I is calculated using the following formula:

$$I = max(m_I) + \frac{\sum m_I - max(m_I)}{4}$$

2.3 SEVERITY COEFFICIENT

Reversibility (R):

Describes the share of the exploited vulnerability effects that can be reversed. For upgradeable contracts, assume the contract private key is available.

Scope (S):

Captures whether a vulnerability in one vulnerable contract impacts resources in other contracts.

Coefficient (C)	Coefficient Value	Numerical Value
Reversibility (r)	None (R:N) Partial (R:P) Full (R:F)	1 0.5 0.25
Scope (s)	Changed (S:C) Unchanged (S:U)	1.25 1

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

$$S = min(10, EIC * 10)$$

The score is rounded up to 1 decimal places.

Severity	Score Value Range
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9
Low	2 - 4.4
Informational	0 - 1.9

2.4 SCOPE

1. IN-SCOPE TREE & COMMIT :

The security assessment was scoped to the following smart contracts:

GitHub repository: Lodestar-Finance/lodestar-protocol
Assessed Commit ID: d19be010dea56ba706449e68d394591e97b30916
Fixed Commit ID #1: 241ad9405b6b0d950eca14faeb3a7bf2fd18c824, PR #55
Fixed Commit ID #2: 5bae12489d1bca1ae42265853f43d5749a4b0dc8, PR #54
Fixed Commit ID #3: e0d50353800fa6cff1444fb7144ca47518195427
Fixed Commit ID #4: f62e743da12da115333ddedb25721cb9bde7f7d2
Smart contracts in scope:
- BaseJumpRateModelV2.sol
- CarefulMath.sol
- CDaiDelegate.sol
- CErc20.sol
- CErc20Delegate.sol
- CErc20Delegator.sol
- CErc20Immutable.sol
- CEther.sol
- Comptroller.sol
- ComptrollerStorage.sol
- CToken.sol
- CTokenInterfaces.sol
- ERC20.sol
- ErrorReporter.sol
- Exponential.sol
- ExponentialNoError.sol
- Fauceteer.sol
- InterestRateModel.sol
- JumpRateModel.sol
- JumpRateModelV2.sol
- Maximillion.sol
- PriceOracle.sol
- Reservoir.sol
- SimplePriceOracle.sol

- Timelock.sol
- Unitroller.sol
- Whitelist.sol
- WhitePaperInterestRateModel.sol
- Comp.sol
- GovernorAlpha.sol
- GovernorBravoDelegate.sol
- GovernorBravoDelegator.sol
- CompoundLens.sol
- Denominations.sol
- PriceOracleProxyETH.sol
- SushiOracle.sol
- CEtherDelegate.sol
- CEtherDelegator.sol
- CEtherUpgradeable.sol

GitHub repository: Lodestar-Finance/plvglp-oracle
Assessed Commit ID: 569f6945bf254587ff645c17584fcf79d8fcc8cc
Smart contracts in scope:
- plvGLPOracle.sol

GitHub repository: LodestarFinance/lodestar-token-fix
Assessed Commit ID: 8807ec57f88b96421e97b0011833897ac48b4bb6
Smart contracts in scope:
- TokenFix.sol

GitHub repository: Lodestar-Finance/plvGLP-looper
Assessed Commit ID: 0731a52b1b02aa69521f21d30fc11cb37c0a0cc8

2. REMEDIATION COMMIT IDs:

Fixed Commit ID #1: 306d538a0b295fb96b37ccd774c27fb658b0538f
Fixed Commit ID #2: 164063788549a7b9db59284c36eab7905d76dcc6
Fixed Commit ID #3: ac8c4db810f221a30aed3c2125283deef10173c8
Fixed Commit ID #4: 7c706d50739de365cff6b46739f5e71bf37db4ac
Fixed Com56cd1f57d4663d163b07012852b148c0311075a7
Smart contracts in scope:
- Ploopy.sol renamed to Loopy.sol

3. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
2	2	4	7	6

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) ANY USER CAN BORROW ON BEHALF OF ANOTHER USER THAT APPROVED THE PLOOPY CONTRACT	Critical (10)	SOLVED - 06/14/2023
(HAL-02) CETHERDELEGATOR DELEGATECALLS SOME CETHERUPGRADEABLE FUNCTIONS INCORRECTLY, ALWAYS REVERTING	Critical (10)	SOLVED - 07/14/2023
(HAL-03) EMPTY MARKETS ARE VULNERABLE TO INFLATION ATTACKS	High (8.4)	SOLVED - 06/18/2023
(HAL-04) LOOPING WITH PLVGLP BORROWS USDC AND REPAYS WITH PLVGLP, ALWAYS REVERTING	High (7.5)	SOLVED - 07/14/2023
(HAL-05) PLOOPY LEVERAGE CALCULATION IS WRONGLY IMPLEMENTED	Medium (5.6)	SOLVED - 07/05/2023
(HAL-06) SEQUENCER STATUS IS NOT CHECKED FOR LPLVGLP PRICE	Medium (5.0)	SOLVED - 07/05/2023
(HAL-07) SWAPTHROUGHUNISWAP CALL WILL ALWAYS REVERT AS NATIVEUSDC WILL ALWAYS BE ZERO	Medium (5.0)	SOLVED - 07/14/2023
(HAL-08) MISSING INITIAL SWAP CALL TO CONVERT USDCBRIDGED INTO USDCNATIVE BEFORE MINTING	Medium (5.0)	SOLVED - 07/14/2023
(HAL-09) USEWALLETBALANCE PARAMETER IS NOT USED CORRECTLY	Low (3.9)	SOLVED - 07/05/2023
(HAL-10) BLOCKSPERYEAR ARE NOT CORRECTLY ADJUSTED IN THE RATE MODELS	Low (3.1)	RISK ACCEPTED
(HAL-11) MINTANDSTAKEGLP() CALL DOES NOT CHECK FOR SLIPPAGE	Low (2.6)	SOLVED - 07/05/2023
(HAL-12) LATESTROUNDDATA CALL MAY RETURN STALE RESULTS	Low (2.5)	RISK ACCEPTED
(HAL-13) PLOOPY CONTRACT ASSUMES THAT THE BALANCER FLASHLOAN FEE WILL ALWAYS BE ZERO	Low (2.5)	SOLVED - 07/14/2023
(HAL-14) LOOPING CAN NOT BE DISABLED AFTER BEING ENABLED	Low (2.5)	SOLVED - 07/05/2023

(HAL-15) WRONG REQUIRE CHECK MESSAGE IN SETAGGREGATORS FUNCTION	Low (2.5)	SOLVED - 07/05/2023
(HAL-16) BLOCK.TIMESTAMP IS USED TO SET THE SWAP DEADLINE	Informational (0.0)	ACKNOWLEDGED
(HAL-17) FUNCTION GETLODEPRICE(ADDRESS POOLADDRESS) CAN BE REMOVED	Informational (0.0)	SOLVED - 07/05/2023
(HAL-18) SUSHIORACLE PRICE CAN BE MANIPULATED	Informational (0.0)	SOLVED - 06/18/2023
(HAL-19) PLOOPY CONTRACT IS MISSING A FUNCTION TO ADD NEW MARKETS	Informational (0.0)	SOLVED - 07/05/2023
(HAL-20) LACK OF A DOUBLE-STEP TRANSFEROWNERSHIP PATTERN	Informational (0.0)	SOLVED - 07/05/2023
(HAL-21) ALLOWANCE CHECK CAN BE REMOVED FROM PLOOPY.LOOP FUNCTION	Informational (0.0)	SOLVED - 07/05/2023

FINDINGS & TECH DETAILS

4.1 (HAL-01) ANY USER CAN BORROW ON BEHALF OF ANOTHER USER THAT APPROVED THE PLOOPY CONTRACT - CRITICAL(10)

Commit IDs affected:
- 0731a52b1b02aa69521f21d30fc11cb37c0a0cc8

Description:

In the Ploopy contract, the receiveFlashLoan() function is used to handle all the logic when a flash loan is received from the Balancer contract:

Listing 1: Ploopy.sol (Line 170)

```
164 function receiveFlashLoan(
165   IERC20[] memory tokens,
166   uint256[] memory amounts,
167   uint256[] memory feeAmounts,
168   bytes memory userData
169 ) external override nonReentrant {
170   if (msg.sender != address(BALANCER_VAULT)) revert UNAUTHORIZED('
 ↳ balancer vault is not the sender');
171
172   // additional checks?
173
174   UserData memory data = abi.decode(userData, (UserData));
175   if (data.borrowedAmount != amounts[0] || data.borrowedToken !=
 ↳ tokens[0]) revert FAILED('borrowed amounts and/or borrowed tokens
 ↳ do not match initially set values');
176
177   // sanity check: flashloan has no fees
178   if (feeAmounts[0] > 0) revert FAILED('balancer fee > 0');
179
180   // account for some plvGLP specific logic
181   if (data.tokenToLoop == PLVGLP) {
182     // mint GLP. approval needed.
183     uint256 glpAmount = REWARD_ROUTER_V2.mintAndStakeGlp(
184       address(data.borrowedToken),
185       data.borrowedAmount,
186       0,
```

```
187         0
188       );
189       if (glpAmount == 0) revert FAILED('glp=0');
190
191       // TODO whitelist this contract for plvGLP mint
192       // mint plvGLP. approval needed.
193       uint256 _oldPlvglpBal = PLVGLP.balanceOf(address(this));
194       GLP_DEPOSITOR.deposit(glpAmount);
195
196       // check new balances and confirm we properly minted
197       uint256 _newPlvglpBal = PLVGLP.balanceOf(address(this));
198       emit plvGLPBalance(_newPlvglpBal);
199       require(_newPlvglpBal > _oldPlvglpBal, "glp deposit failed,
 ↳ new balance < old balance");
200     }
201
202     uint256 _finalBal;
203
204     // mint our respective token by depositing it into Lodestar's
 ↳ respective lToken contract (approval needed)
205     unchecked {
206       lTokenMapping[data.tokenToLoop].mint(data.tokenToLoop.
 ↳ balanceOf(address(this)));
207       lTokenMapping[data.tokenToLoop].transfer(data.user,
 ↳ lTokenMapping[data.tokenToLoop].balanceOf(address(this)));
208       _finalBal = lTokenMapping[data.tokenToLoop].balanceOf(address(
 ↳ this));
209
210       emit lTokenBalance(_finalBal);
211       require(_finalBal == 0, "lToken balance not 0 at the end of
 ↳ loop");
212     }
213
214     // call borrowBehalf to borrow tokens on behalf of user
215     lTokenMapping[data.tokenToLoop].borrowBehalf(data.borrowedAmount
 ↳ , data.user);
216     // repay loan, where msg.sender = vault
217     data.tokenToLoop.safeTransferFrom(data.user, msg.sender, data.
 ↳ borrowedAmount);
218 }
```

The `receiveFlashLoan()` function has some access control mechanism in place: This function can only be called by the `Balancer` contract.

Although, any user is free to call the balancer.flashLoan() function passing it the address of the Looper:

Listing 2: flashLoan() call (Lines 2,10)

```
1 UserData memory userData = UserData({
2     user: victim_address,
3     tokenAmount: 10000_000000,
4     borrowedToken: contract_USDC,
5     borrowedAmount: 10000_000000,
6     tokenToLoop: contract_USDC
7 });
8
9 // Exploit call
10 BALANCER_VAULT.flashLoan(IFlashLoanRecipient(contract_Ploopy, [
↳ USDC], [10000_000000], abi.encode(userData));
```

Balancer contract would call back the Ploopy.receiveFlashLoan() function and trigger a borrow on behalf of another user: lTokenMapping[data.tokenToLoop].borrowBehalf(data.borrowedAmount, data.user);

The consequences:
1. Lowering the account liquidity of the users, being able to get them very close to a liquidation.
2. If the attack is performed with the plvGLP token, this will trigger a REWARD_ROUTER_V2.mintAndStakeGlp(address(data.borrowedToken), data.borrowedAmount, 0, 0) call. As the borrowedAmount is a user controlled parameter and the slippage is set to 0, the attacker can sandwich this call to get profit constantly from different users in the Lodestar protocol.

Proof of Concept:

```
User1 calls -> contract_USDC.approve(contract_Ploopy, type(uint256).max)

User1 calls -> contract_Comptroller.enableLooping(true)
  contract_USDC.balanceOf(user1) -> 1000000000
  contract_USDC.balanceOf(user2) -> 0
  contract_USDC.balanceOf(contract_Ploopy) -> 0

  ----------------------------------------
  Account liquidity for -> 0xE6b3367318C5e11a6eED3Cd0D850eC06A02E9b90
      error -> 0
      liquidity -> 509185705432086500
      shortfall -> 0

  ----------------------------------------
  ----------------------------------------
  Account liquidity for -> 0x88C0e901bd1fd1a77BdA342f0d2210fDC71Cef6B
      error -> 0
      liquidity -> 0
      shortfall -> 0

  ----------------------------------------

User2(attacker) calls -> BALANCER_VAULT.flashLoan(IFlashLoanRecipient(contract_Ploopy, tokens, 56600e6, abi.encode(userData))

contract_USDC.balanceOf(user1) -> 1000000000
  contract_USDC.balanceOf(user2) -> 0
  contract_USDC.balanceOf(address(contract_Ploopy)) -> 0

  ----------------------------------------
  Account liquidity for -> 0xE6b3367318C5e11a6eED3Cd0D850eC06A02E9b90
      error -> 0
      liquidity -> 5990420063906900
      shortfall -> 0

  ----------------------------------------
  ----------------------------------------
  Account liquidity for -> 0x88C0e901bd1fd1a77BdA342f0d2210fDC71Cef6B
      error -> 0
      liquidity -> 0
      shortfall -> 0

  ----------------------------------------
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:H/A:N/D:C/Y:N/R:N/S:U (10)

Recommendation:

It is recommended to add a `require` statement in the `Ploopy.receiveFlashLoan()` function that checks that `tx.origin` is equal to the `userData.user` field passed as parameter to this function.

Remediation Plan:

SOLVED: The Lodestar team solved the issue by implementing the recommended solution.

Commit ID : 306d538a0b295fb96b37ccd774c27fb658b0538f.

4.2 (HAL-02) CETHERDELEGATOR DELEGATECALLS SOME CETHERUPGRADEABLE FUNCTIONS INCORRECTLY, ALWAYS REVERTING - CRITICAL(10)

Commit IDs affected:
- e0d50353800fa6cff1444fb7144ca47518195427

Description:

The CEtherDelegator is used as a proxy contract for the CEtherDelegate contract. The CEtherDelegator contract delegate calls to the implementation (CEtherDelegate). Although, some of the functions implemented in the CEtherDelegator contract delegate calls functions selectors that do not exist in the CEtherDelegate contract. For example:

Listing 3: CEtherDelegator.sol (Line 89)

```
88 function mint() external payable override {
89     delegateToImplementation(abi.encodeWithSignature("mint(uint256
↳ )", msg.value));
90 }
```

Listing 4: CEtherUpgradeable.sol (Line 38)

```
38 function mint() external payable {
39     mintInternal(msg.value);
40 }
```

Listing 5: CEtherDelegator.sol (Line 157)

```
156 function repayBorrow() external payable override {
157     delegateToImplementation(abi.encodeWithSignature("repayBorrow(
↳ uint256)", msg.value));
```

```
158 }
```

Listing 6: CEtherUpgradeable.sol (Line 78)

```
78 function repayBorrow() external payable {
79     repayBorrowInternal(msg.value);
80 }
```

Listing 7: CEtherDelegator.sol (Line 157)

```
156 function repayBorrowBehalf(address borrower) external payable
 ↳ override {
157     delegateToImplementation(abi.encodeWithSignature("
 ↳ repayBorrowBehalf(address,uint256)", borrower, msg.value));
158 }
```

Listing 8: CEtherUpgradeable.sol (Line 78)

```
78 function repayBorrowBehalf(address borrower) external payable {
79     repayBorrowBehalfInternal(borrower, msg.value);
80 }
```

Listing 9: CEtherDelegator.sol (Lines 175-177)

```
174 function liquidateBorrow(address borrower, CTokenInterface
 ↳ cTokenCollateral) external payable override {
175     delegateToImplementation(
176         abi.encodeWithSignature("liquidateBorrow(address,address)"
 ↳ , borrower, cTokenCollateral)
177     );
178 }
```

Listing 10: CEtherUpgradeable.sol (Line 122)

```
122 function liquidateBorrow(address borrower, CToken cTokenCollateral
 ↳ ) external payable returns (uint) {
123     liquidateBorrowInternal(borrower, msg.value, cTokenCollateral)
 ↳ ;
124     return NO_ERROR;
125 }
```

These function calls will always revert, leaving the CEtherDelegator unusable.

Functions affected:

- CEtherDelegator.mint()
- CEtherDelegator.repayBorrow()
- CEtherDelegator.repayBorrowBehalf()
- CEtherDelegator.liquidateBorrow()

Proof of Concept:

```
├─ [5866] CEtherDelegator::mint{value: 10000000000000000000}()
│   ├─ [235] CEtherDelegate::mint(10000000000000000000) [delegatecall]
│   │   └─ ← "EvmError: Revert"
│   └─ ← "EvmError: Revert"
└─ ← "EvmError: Revert"
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:C/A:N/D:N/Y:N/R:N/S:U (10)

Recommendation:

It is recommended to correct in the CEtherDelegator contract the different abi.encodeWithSignature() calls to use the correct function signatures.

Remediation Plan:

SOLVED: The Lodestar team solved the issue by implementing the recommended solution.

Commit ID : f62e743da12da115333ddedb25721cb9bde7f7d2.

4.3 (HAL-03) EMPTY MARKETS ARE VULNERABLE TO INFLATION ATTACKS - HIGH (8.4)

Commit IDs affected:
- d19be010dea56ba706449e68d394591e97b30916

Description:

This vulnerability has existed in the Compound v2 code since its launch, presenting itself when markets are launched with a collateral value in place but no depositors or following markets becoming empty due to user withdrawal post-launch.
This issue exploits a rounding error that is present in the redeemFresh() function:

Listing 11: CToken.sol (Line 520)

```
490 /**
491  * @notice User redeems cTokens in exchange for the underlying
 ↳ asset
492  * @dev Assumes interest has already been accrued up to the
 ↳ current block
493  * @param redeemer The address of the account which is redeeming
 ↳ the tokens
494  * @param redeemTokensIn The number of cTokens to redeem into
 ↳ underlying (only one of redeemTokensIn or redeemAmountIn may be
 ↳ non-zero)
495  * @param redeemAmountIn The number of underlying tokens to
 ↳ receive from redeeming cTokens (only one of redeemTokensIn or
 ↳ redeemAmountIn may be non-zero)
496  */
497 function redeemFresh(address payable redeemer, uint redeemTokensIn
 ↳ , uint redeemAmountIn) internal {
498     require(redeemTokensIn == 0 || redeemAmountIn == 0, "one of
 ↳ redeemTokensIn or redeemAmountIn must be zero");
499
500     /* exchangeRate = invoke Exchange Rate Stored() */
```

```
501     Exp memory exchangeRate = Exp({mantissa:
 ↳ exchangeRateStoredInternal()});
502
503     uint redeemTokens;
504     uint redeemAmount;
505     /* If redeemTokensIn > 0: */
506     if (redeemTokensIn > 0) {
507         /*
508          * We calculate the exchange rate and the amount of
 ↳ underlying to be redeemed:
509          *  redeemTokens = redeemTokensIn
510          *  redeemAmount = redeemTokensIn x exchangeRateCurrent
511          */
512         redeemTokens = redeemTokensIn;
513         redeemAmount = mul_ScalarTruncate(exchangeRate,
 ↳ redeemTokensIn);
514     } else {
515         /*
516          * We get the current exchange rate and calculate the
 ↳ amount to be redeemed:
517          *  redeemTokens = redeemAmountIn / exchangeRate
518          *  redeemAmount = redeemAmountIn
519          */
520         redeemTokens = div_(redeemAmountIn, exchangeRate);
521         redeemAmount = redeemAmountIn;
522     }
523
524     /* Fail if redeem not allowed */
525     uint allowed = comptroller.redeemAllowed(address(this),
 ↳ redeemer, redeemTokens);
526     if (allowed != 0) {
527         revert RedeemComptrollerRejection(allowed);
528     }
529
530     /* Verify market's block number equals current block number */
531     if (accrualBlockNumber != getBlockNumber()) {
532         revert RedeemFreshnessCheck();
533     }
534
535     /* Fail gracefully if protocol has insufficient cash */
536     if (getCashPrior() < redeemAmount) {
537         revert RedeemTransferOutNotPossible();
538     }
539
```

```
540     ////////////////////////
541     // EFFECTS & INTERACTIONS
542     // (No safe failures beyond this point)
543
544     /*
545      * We write the previously calculated values into storage.
546      *  Note: Avoid token reentrancy attacks by writing reduced
 ↳ supply before external transfer.
547      */
548     totalSupply = totalSupply - redeemTokens;
549     accountTokens[redeemer] = accountTokens[redeemer] -
 ↳ redeemTokens;
550
551     /*
552      * We invoke doTransferOut for the redeemer and the
 ↳ redeemAmount.
553      *  Note: The cToken must handle variations between ERC-20 and
 ↳  ETH underlying.
554      *  On success, the cToken has redeemAmount less of cash.
555      *  doTransferOut reverts if anything goes wrong, since we can
 ↳ 't be sure if side effects occurred.
556      */
557     doTransferOut(redeemer, redeemAmount);
558
559     /* We emit a Transfer event, and a Redeem event */
560     emit Transfer(redeemer, address(this), redeemTokens);
561     emit Redeem(redeemer, redeemAmount, redeemTokens);
562
563     /* We call the defense hook */
564     comptroller.redeemVerify(address(this), redeemer, redeemAmount
 ↳ , redeemTokens);
565 }
```

This is achieved by donating a large amount of the underlying asset to the market contract, manipulating the exchangeRate.

Proof of Concept:

```
    Initial attacker Ether balance: 0
    Initial attacker UNI balance: 10000000000000000000000000

1. Deposit a small amount of UNI to the empty lUNI pool to obtain shares
     ATTACKER CALLS -> contract_Uni.approve(address(contract_lUni), type(uint256).max)
     ATTACKER CALLS -> contract_lUni.mint(4 * 1e8)
     ATTACKER CALLS -> contract_lUni.redeem(contract_lUni.totalSupply() - 2)
     Attacker's share in lUNI: 2
     UNI amount in lUNI: 400000000

2. Donate a large amount of UNI to the lUNI pool to increase the exchangeRate(the number of UNI represented by each share)
    exchangeRate before manipulation: 200000000000000000000000000
     ATTACKER CALLS -> contract_Uni.transfer(address(contract_lUni), 9999999999999999600000000)
    exchangeRate after manipulation: 50000000000000000000000000000000000000000000

3. Lend tokens from the lUNI pool
     ATTACKER CALLS -> contract_Comptroller.enterMarkets([contract_lETH])
     ATTACKER CALLS -> contract_lETH.borrow(521648677747296581033)

4. Redeem UNI from the lUNI pool
    Calculate the amount of shares represented by the redeem amount: 1
    Another way of calculating, redeemAmount * 1e18 / exchangeRate: 1
    Due to the inflation attack, the attacker redeems all previously donated UNI with a calculated share of: 1
     ATTACKER CALLS -> contract_lUni.redeemUnderlying(999999999999999599999999)
    After redeeming the attacker's share in lUNI: 1
    The UNI amount in lUNI: 400000001
    Attacker Ether balance: 521648677747296581033
    Attacker UNI balance: 999999999999999599999999
```

References:

- Hundred Finance Hack Post Mortem
- Hundred Finance exploit example

BVSS:

AO:A/AC:L/AX:M/C:N/I:N/A:N/D:C/Y:C/R:N/S:U (8.4)

Recommendation:

It is recommended to ensure that markets are never empty by minting small **cToken** (or equivalent) balances at the time of market creation, preventing the rounding error being used maliciously. A possible approach is following UniswapV2 implementation that permanently locks the first MINIMUM_LIQUIDITY tokens.

Remediation Plan:

SOLVED: The Lodestar team is aware of this CompoundV2 issue and states that and have a team controlled wallet with some of each asset deposited in each market to always have a MINIMUM_LIQUIDITY in every market.

Moreover, the Lodestar team updated the CToken implementation, so it ensures that there is always a minimum liquidity of 1e6 tokens in the market.

Commit ID : 5bae12489d1bca1ae42265853f43d5749a4b0dc8.

4.4 (HAL-04) LOOPING WITH PLVGLP BORROWS USDC AND REPAYS WITH PLVGLP, ALWAYS REVERTING - HIGH (7.5)

Commit IDs affected:
- 0731a52b1b02aa69521f21d30fc11cb37c0a0cc8

Description:

The contract Ploopy allows looping with the following assets:
- USDC: Borrows USDC, repays with USDC.
- USDT: Borrows USDT, repays with USDT.
- DAI: Borrows DAI, repays with DAI.
- WBTC: Borrows WBTC, repays with WBTC.
- FRAX: Borrows FRAX, repays with FRAX.
- ARB: Borrows ARB, repays with ARB.
- plvGLP: Borrows USDC, repays with plvGLP.

As we can see below, when users loop with the plvGLP token, a different logic is performed:

Listing 12: Ploopy.sol (Line 118)

```
112 function loop(IERC20 _token, uint256 _amount, uint16 _leverage,
 ↳ uint16 _useWalletBalance) external {
113   require(allowedTokens[_token], "token not allowed to loop");
114   require(tx.origin == msg.sender, "not an EOA");
115   require(_amount > 0, "amount must be greater than 0");
116   require(_leverage >= DIVISOR && _leverage <= MAX_LEVERAGE, "
 ↳ invalid leverage, range must be between DIVISOR and MAX_LEVERAGE
 ↳ values");
117
118   // if the user wants us to mint using their existing wallet
 ↳ balance (indiciated with 1), then do so.
119   // otherwise, read their existing balance and flash loan to
 ↳ increase their position
120   if (_useWalletBalance == 1) {
121     // transfer tokens to this contract so we can mint in 1 go.
```

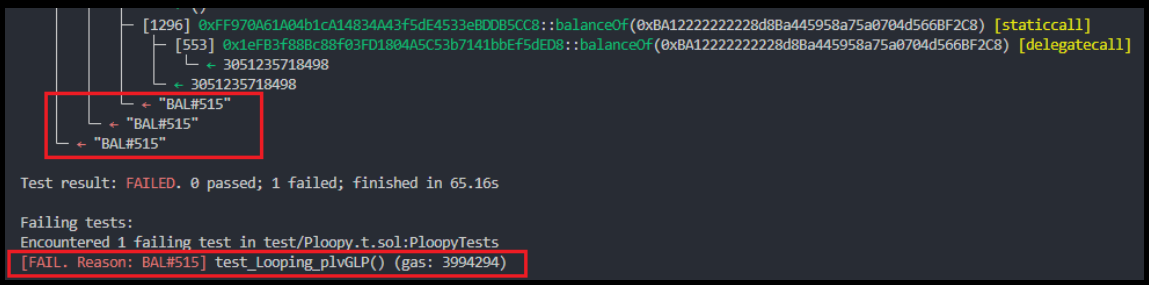
```
122     _token.safeTransferFrom(msg.sender, address(this), _amount);
123     emit Transfer(msg.sender, address(this), _amount);
124   }
125
126   uint256 loanAmount;
127   IERC20 _tokenToBorrow;
128
129   if (_token == PLVGLP) {
130     uint256 _tokenPriceInEth;
131     uint256 _usdcPriceInEth;
132     uint256 _computedAmount;
133
134     // plvGLP borrows USDC to loop
135     _tokenToBorrow = USDC;
136     _tokenPriceInEth = PRICE_ORACLE.getUnderlyingPrice(address(
 ↳ lTokenMapping[_token]));
137     _usdcPriceInEth = (PRICE_ORACLE.getUnderlyingPrice(address(
 ↳ lUSDC)) / 1e12);
138     _computedAmount = (_amount * (_tokenPriceInEth /
 ↳ _usdcPriceInEth));
139
140     loanAmount = getNotionalLoanAmountIn1e18(
141       _computedAmount,
142       _leverage
143     );
144   } else {
145     // the rest of the contracts just borrow whatever token is
 ↳ supplied
146     _tokenToBorrow = _token;
147     loanAmount = getNotionalLoanAmountIn1e18(
148       _amount, // we can just send over the exact amount, as we
 ↳ are either looping stables or eth
149       _leverage
150     );
151   }
152
153   if (_tokenToBorrow.balanceOf(address(BALANCER_VAULT)) <
 ↳ loanAmount) revert FAILED('balancer vault token balance < loan');
154   emit Loan(loanAmount);
155   emit BalanceOf(_tokenToBorrow.balanceOf(address(BALANCER_VAULT))
 ↳ , loanAmount);
156
157   // check approval to spend USDC (for paying back flashloan).
158   // possibly can omit to save gas as tx will fail with exceed
```

```
 ↳ allowance anyway.
159   if (_tokenToBorrow.allowance(msg.sender, address(this)) <
 ↳ loanAmount) revert INVALID_APPROVAL();
160   emit Allowance(_tokenToBorrow.allowance(msg.sender, address(this
 ↳ )), loanAmount);
161
162   IERC20[] memory tokens = new IERC20[](1);
163   tokens[0] = _tokenToBorrow;
164
165   uint256[] memory loanAmounts = new uint256[](1);
166   loanAmounts[0] = loanAmount;
167
168   UserData memory userData = UserData({
169     user: msg.sender,
170     tokenAmount: _amount,
171     borrowedToken: _tokenToBorrow,
172     borrowedAmount: loanAmount,
173     tokenToLoop: _token
174   });
175   emit UserDataEvent(msg.sender, _amount, address(_tokenToBorrow),
 ↳  loanAmount, address(_token));
176
177   BALANCER_VAULT.flashLoan(IFlashLoanRecipient(this), tokens,
 ↳ loanAmounts, abi.encode(userData));
178 }
179
180 In this case, the `_tokenToBorrow` will be USDC. Although, during
 ↳ the repayment of the flashloan, the contract tries to repay with
 ↳ plvGLP token instead of USDC:
181 ``` {language=solidity firstnumber="164" caption="Ploopy.sol"
 ↳ hlines=214-217}
182 function receiveFlashLoan(
183   IERC20[] memory tokens,
184   uint256[] memory amounts,
185   uint256[] memory feeAmounts,
186   bytes memory userData
187 ) external override nonReentrant {
188   if (msg.sender != address(BALANCER_VAULT)) revert UNAUTHORIZED('
 ↳ balancer vault is not the sender');
189
190   // additional checks?
191
192   UserData memory data = abi.decode(userData, (UserData));
193   if (data.borrowedAmount != amounts[0] || data.borrowedToken !=
```

```
 ↳ tokens[0]) revert FAILED('borrowed amounts and/or borrowed tokens
 ↳ do not match initially set values');
194
195   // sanity check: flashloan has no fees
196   if (feeAmounts[0] > 0) revert FAILED('balancer fee > 0');
197
198   // account for some plvGLP specific logic
199   if (data.tokenToLoop == PLVGLP) {
200     // mint GLP. approval needed.
201     uint256 glpAmount = REWARD_ROUTER_V2.mintAndStakeGlp(
202       address(data.borrowedToken),
203       data.borrowedAmount,
204       0,
205       0
206     );
207     if (glpAmount == 0) revert FAILED('glp=0');
208
209     // TODO whitelist this contract for plvGLP mint
210     // mint plvGLP. approval needed.
211     uint256 _oldPlvglpBal = PLVGLP.balanceOf(address(this));
212     GLP_DEPOSITOR.deposit(glpAmount);
213
214     // check new balances and confirm we properly minted
215     uint256 _newPlvglpBal = PLVGLP.balanceOf(address(this));
216     emit plvGLPBalance(_newPlvglpBal);
217     require(_newPlvglpBal > _oldPlvglpBal, "glp deposit failed,
 ↳ new balance < old balance");
218   }
219
220   uint256 _finalBal;
221
222   // mint our respective token by depositing it into Lodestar's
 ↳ respective lToken contract (approval needed)
223   unchecked {
224     lTokenMapping[data.tokenToLoop].mint(data.tokenToLoop.
 ↳ balanceOf(address(this)));
225     lTokenMapping[data.tokenToLoop].transfer(data.user,
 ↳ lTokenMapping[data.tokenToLoop].balanceOf(address(this)));
226     _finalBal = lTokenMapping[data.tokenToLoop].balanceOf(address(
 ↳ this));
227
228     emit lTokenBalance(_finalBal);
229     require(_finalBal == 0, "lToken balance not 0 at the end of
 ↳ loop");
```

```
230    }
231
232    // call borrowBehalf to borrow tokens on behalf of user
233    lTokenMapping[data.tokenToLoop].borrowBehalf(data.borrowedAmount
 ↳ , data.user);
234    // repay loan, where msg.sender = vault
235    data.tokenToLoop.safeTransferFrom(data.user, msg.sender, data.
 ↳ borrowedAmount);
236 }
```

As the repayment of the flash loan is done with the wrong token, the transaction reverts with the following error: `BAL#515 - INVALID_POST_LOAN_BALANCE`.



BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:H/D:N/Y:N/R:N/S:U (7.5)

Recommendation:

It is recommended to update the `receiveFlashLoan()` function so, in the case that the `tokenToLoop` is plvGLP, the repayment is done with USDC.

Remediation Plan:

SOLVED: The Lodestar team solved the issue by implementing the recommended solution.

Commit ID : 306d538a0b295fb96b37ccd774c27fb658b0538f.

4.5 (HAL-05) PLOOPY LEVERAGE CALCULATION IS WRONGLY IMPLEMENTED - MEDIUM (5.6)

Commit IDs affected:
- 0731a52b1b02aa69521f21d30fc11cb37c0a0cc8

Description:

In the Ploopy contract, the function loop() allows users to loop to a desired leverage, within a pre-set range of 1x to 3x leverage:

Listing 13: Ploopy.sol (Lines 140,147)

```
112 function loop(IERC20 _token, uint256 _amount, uint16 _leverage,
 ↳ uint16 _useWalletBalance) external {
113   require(allowedTokens[_token], "token not allowed to loop");
114   require(tx.origin == msg.sender, "not an EOA");
115   require(_amount > 0, "amount must be greater than 0");
116   require(_leverage >= DIVISOR && _leverage <= MAX_LEVERAGE, "
 ↳ invalid leverage, range must be between DIVISOR and MAX_LEVERAGE
 ↳ values");
117
118   // if the user wants us to mint using their existing wallet
 ↳ balance (indiciated with 1), then do so.
119   // otherwise, read their existing balance and flash loan to
 ↳ increase their position
120   if (_useWalletBalance == 1) {
121     // transfer tokens to this contract so we can mint in 1 go.
122     _token.safeTransferFrom(msg.sender, address(this), _amount);
123     emit Transfer(msg.sender, address(this), _amount);
124   }
125
126   uint256 loanAmount;
127   IERC20 _tokenToBorrow;
128
129   if (_token == PLVGLP) {
130     uint256 _tokenPriceInEth;
131     uint256 _usdcPriceInEth;
132     uint256 _computedAmount;
```

```
133
134     // plvGLP borrows USDC to loop
135     _tokenToBorrow = USDC;
136     _tokenPriceInEth = PRICE_ORACLE.getUnderlyingPrice(address(
 ↳ lTokenMapping[_token]));
137     _usdcPriceInEth = (PRICE_ORACLE.getUnderlyingPrice(address(
 ↳ lUSDC)) / 1e12);
138     _computedAmount = (_amount * (_tokenPriceInEth /
 ↳ _usdcPriceInEth));
139
140     loanAmount = getNotionalLoanAmountIn1e18(
141       _computedAmount,
142       _leverage
143     );
144   } else {
145     // the rest of the contracts just borrow whatever token is
 ↳ supplied
146     _tokenToBorrow = _token;
147     loanAmount = getNotionalLoanAmountIn1e18(
148       _amount, // we can just send over the exact amount, as we
 ↳ are either looping stables or eth
149       _leverage
150     );
151   }
152
153   if (_tokenToBorrow.balanceOf(address(BALANCER_VAULT)) <
 ↳ loanAmount) revert FAILED('balancer vault token balance < loan');
154   emit Loan(loanAmount);
155   emit BalanceOf(_tokenToBorrow.balanceOf(address(BALANCER_VAULT))
 ↳ , loanAmount);
156
157   // check approval to spend USDC (for paying back flashloan).
158   // possibly can omit to save gas as tx will fail with exceed
 ↳ allowance anyway.
159   if (_tokenToBorrow.allowance(msg.sender, address(this)) <
 ↳ loanAmount) revert INVALID_APPROVAL();
160   emit Allowance(_tokenToBorrow.allowance(msg.sender, address(this
 ↳ )), loanAmount);
161
162   IERC20[] memory tokens = new IERC20[](1);
163   tokens[0] = _tokenToBorrow;
164
165   uint256[] memory loanAmounts = new uint256[](1);
166   loanAmounts[0] = loanAmount;
```

```
167
168   UserData memory userData = UserData({
169     user: msg.sender,
170     tokenAmount: _amount,
171     borrowedToken: _tokenToBorrow,
172     borrowedAmount: loanAmount,
173     tokenToLoop: _token
174   });
175   emit UserDataEvent(msg.sender, _amount, address(_tokenToBorrow),
 ↳  loanAmount, address(_token));
176
177   BALANCER_VAULT.flashLoan(IFlashLoanRecipient(this), tokens,
 ↳ loanAmounts, abi.encode(userData));
178 }
```

In order to calculate the amount of tokens loaned, the internal view function `getNotionalLoanAmountIn1e18()` is called:

Listing 14: Ploopy.sol (Line 225)

```
220 function getNotionalLoanAmountIn1e18(
221   uint256 _notionalTokenAmountIn1e18,
222   uint16 _leverage
223 ) private pure returns (uint256) {
224   unchecked {
225     return ((_leverage - DIVISOR) * _notionalTokenAmountIn1e18) /
 ↳ DIVISOR;
226   }
227 }
```

After the `loanAmount` is calculated, a flash loan is taken from the Balancer Vault, the flash loaned tokens are used to mint more `cTokens` and then those tokens are borrowed from the Lodestar market to repay the flash loan. Although, the `leverage` calculation or the term `leverage` is wrongly used here as it simply defines the amount of assets that will be flash loaned from Balancer without considering the current position of the user in Lodestar. For example, let's imagine these 2 different situations:

Situation 1
User1 initial liquidity:
- liquidity: 4_907186648501354500
- shortfall: 0
Uner1 initial snapshot:
- cWantBalance: 50000000000000
- borrowed: 0
- exchangeRate: 200000000000000

User1 calls: contract_Ploopy.loop(USDC, 1000e6, 30000, 0)
- 2000_000000 USDC is flash loaned from Balancer Vault.
- mint() is called, minting more cTokens.
- 100000_00000000 cTokens are transferred to user1.
- 2000_000000 USDC are borrowed from the lUSDC market.
- This 2000_000000 USDC is used to repay the Balancer's flash loan.

User1 final liquidity:
- liquidity: 4_734319793085926400
- shortfall: 0
Uner1 final snapshot:
- cWantBalance: 60000000000000
- borrowed: 2000000000
- exchangeRate: 200000000000000

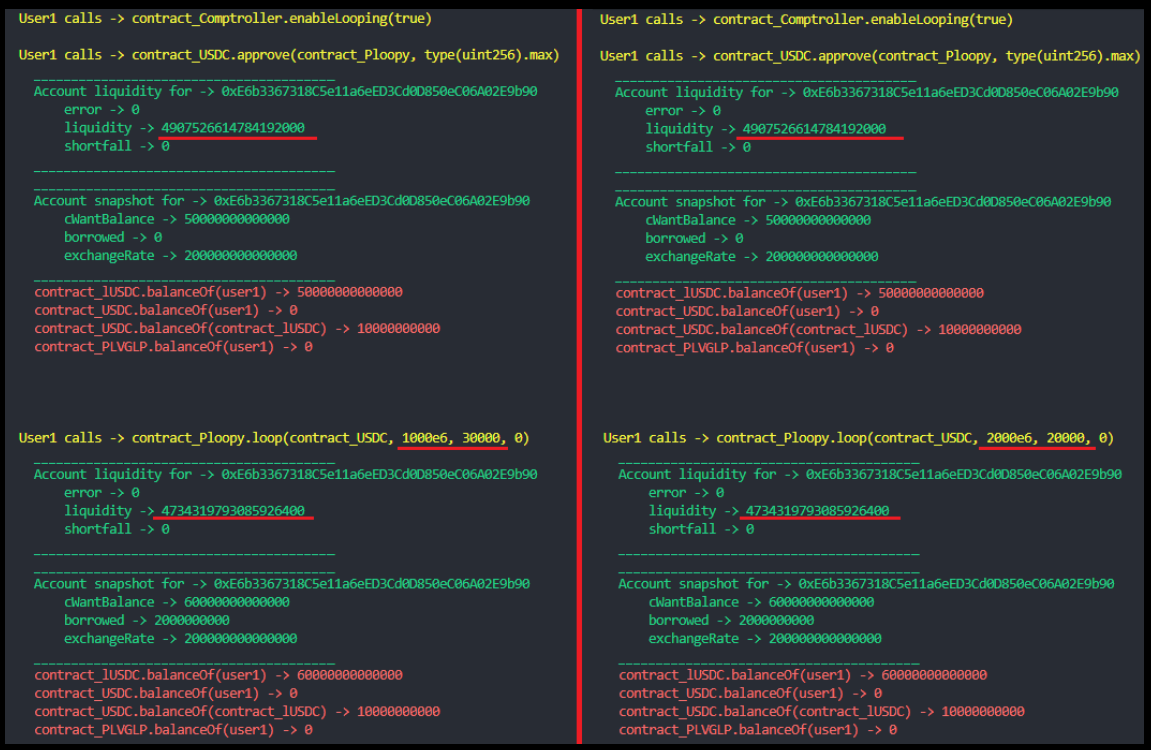
Situation 2
User1 initial liquidity:
- liquidity: 4_907186648501354500
- shortfall: 0
Uner1 initial snapshot:
- cWantBalance: 50000000000000
- borrowed: 0
- exchangeRate: 200000000000000

User1 calls: contract_Ploopy.loop(USDC, 2000e6, 20000, 0)
- 2000_000000 USDC is flash loaned from Balancer Vault.
- mint() is called, minting more cTokens.
- 100000_00000000 cTokens are transferred to user1.
- 2000_000000 USDC are borrowed from the lUSDC market.

- This 2000_000000 USDC is used to repay the Balancer's flash loan.

User1 final liquidity:
- liquidity: 4_734319793085926400
- shortfall: 0

Uner1 final snapshot:
- cWantBalance: 60000000000000
- borrowed: 2000000000
- exchangeRate: 200000000000000



As we can see, both situations end up flash loaning 2000 USDC from Balancer and getting the user1 Lodestar's position to the same liquidity. Hence, the leverage check of being between 1x and 3x can be easily bypassed as the user can simply call the loop() function with a higher _amount.

BVSS:

AO:A/AC:L/AX:L/C:N/I:L/A:N/D:M/Y:N/R:N/S:U (5.6)

Recommendation:

Consider refactoring the loop() function and the Ploopy contract in general, so the looping considers the current position of the user in Lodestar to calculate the leverage.

Remediation Plan:

SOLVED: The Lodestar team solved the issue by implementing the recommended solution.

Commit ID : 164063788549a7b9db59284c36eab7905d76dcc6.

4.6 (HAL-06) SEQUENCER STATUS IS NOT CHECKED FOR LPLVGLP PRICE - MEDIUM (5.0)

Commit IDs affected:
- d19be010dea56ba706449e68d394591e97b30916

Description:

In the PriceOracleProxyETH contract the function getUnderlyingPrice() gets the underlying price of a listed cToken asset:

Listing 15: PriceOracleProxyETH.sol (Line 108)

```
92 /**
93  * @notice Get the underlying price of a listed cToken asset
94  * @param cToken The cToken to get the underlying price of
95  * @return The underlying asset price mantissa (scaled by 1e18)
 ↳ Price is given in ether (1e18 = 1 ether)
96  */
97 function getUnderlyingPrice(CToken cToken) public view returns (
 ↳ uint256) {
98     address cTokenAddress = address(cToken);
99     AggregatorInfo memory aggregatorInfo = aggregators[
 ↳ cTokenAddress];
100     if (cTokenAddress == letherAddress) {
101         uint256 price = 1e18;
102         return price;
103     } else if (cTokenAddress == lplvGLPAddress) {
104         uint256 price = getPlvGLPPrice();
105         price = div_(price, Exp({mantissa: getPriceFromChainlink(
 ↳ ethUsdAggregator)}));
106         return price;
107     } else if (address(aggregatorInfo.source) != address(0)) {
108         bool sequencerStatus = getSequencerStatus(sequencerAddress
 ↳ );
109         uint256 price = getPriceFromChainlink(aggregatorInfo.
 ↳ source);
110         if (sequencerStatus == false) {
```

```
111             // If flag is raised we shouldn't perform any critical
 ↳  operations
112             revert("Chainlink feeds are not being updated");
113         } else if (aggregatorInfo.base == AggregatorBase.USD) {
114             // Convert the price to ETH based if it's USD based.
115             price = div_(price, Exp({mantissa:
 ↳ getPriceFromChainlink(ethUsdAggregator)}));
116             uint256 underlyingDecimals = EIP20Interface(CErc20(
 ↳ cTokenAddress).underlying()).decimals();
117             return price * 10 ** (18 - underlyingDecimals);
118         } else if (aggregatorInfo.base == AggregatorBase.ETH) {
119             uint256 underlyingDecimals = EIP20Interface(CErc20(
 ↳ cTokenAddress).underlying()).decimals();
120             return price * 10 ** (18 - underlyingDecimals);
121         }
122     }
123     revert("Invalid Oracle Request");
124 }
```

In the line 108, a call is done to the sequencer feed in order to check its status and revert in case that the sequencer is down. Although, this should also be checked when `cTokenAddress == lplvGLPAddress` as a call to the `ethUsdAggregator` is also done. This could lead to get an outdated price from the `ethUsdAggregator`.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:M/Y:N/R:N/S:U (5.0)

Recommendation:

It is recommended to update the `getUnderlyingPrice()` function as shown below:

Listing 16: PriceOracleProxyETH.sol (Lines 100,105,106,107,108)

```
92 /**
93  * @notice Get the underlying price of a listed cToken asset
94  * @param cToken The cToken to get the underlying price of
```

```
95  * @return The underlying asset price mantissa (scaled by 1e18)
↳ Price is given in ether (1e18 = 1 ether)
96  */
97 function getUnderlyingPrice(CToken cToken) public view returns (
↳ uint256) {
98     address cTokenAddress = address(cToken);
99     AggregatorInfo memory aggregatorInfo = aggregators[
↳ cTokenAddress];
100    bool sequencerStatus;
101    if (cTokenAddress == letherAddress) {
102        uint256 price = 1e18;
103        return price;
104    } else if (cTokenAddress == lplvGLPAddress) {
105        sequencerStatus = getSequencerStatus(sequencerAddress);
106        if (sequencerStatus == false){
107            revert("Chainlink feeds are not being updated");
108        }
109        uint256 price = getPlvGLPPrice();
110        price = div_(price, Exp({mantissa: getPriceFromChainlink(
↳ ethUsdAggregator)}));
111        return price;
112    } else if (address(aggregatorInfo.source) != address(0)) {
113        sequencerStatus = getSequencerStatus(sequencerAddress);
114        uint256 price = getPriceFromChainlink(aggregatorInfo.
↳ source);
115        if (sequencerStatus == false) {
116            // If flag is raised we shouldn't perform any critical
↳  operations
117            revert("Chainlink feeds are not being updated");
118        } else if (aggregatorInfo.base == AggregatorBase.USD) {
119            // Convert the price to ETH based if it's USD based.
120            price = div_(price, Exp({mantissa:
↳ getPriceFromChainlink(ethUsdAggregator)}));
121            uint256 underlyingDecimals = EIP20Interface(CErc20(
↳ cTokenAddress).underlying()).decimals();
122            return price * 10 ** (18 - underlyingDecimals);
123        } else if (aggregatorInfo.base == AggregatorBase.ETH) {
124            uint256 underlyingDecimals = EIP20Interface(CErc20(
↳ cTokenAddress).underlying()).decimals();
125            return price * 10 ** (18 - underlyingDecimals);
126        }
127    }
128    revert("Invalid Oracle Request");
129 }
```

Remediation Plan:

SOLVED: The Lodestar team solved the issue by implementing the recommended solution.

Commit ID : 241ad9405b6b0d950eca14faeb3a7bf2fd18c824.

4.7 (HAL-07) SWAPTHROUGHUNISWAP CALL WILL ALWAYS REVERT AS NATIVEUSDC WILL ALWAYS BE ZERO - MEDIUM (5.0)

Commit IDs affected:
- ac8c4db810f221a30aed3c2125283deef10173c8

Description:

In the Loopy contract, in the receiveFlashloan() function, in the specific logic related to the USDC market, the following call is executed:

Listing 17: Loopy.sol

```
326 lUSDC.borrowBehalf(repayAmountFactoringInFeeAmount, data.user);
```

This call will borrow and send USDC to the data.user. Although, the contract assumes that this USDC amount will be received by the contract itself:

Listing 18: Loopy.sol

```
327 //we need to swap our native USDC for bridged USDC to repay the
 ↳ loan
328 uint256 nativeUSDCBalance = USDC_NATIVE.balanceOf(address(this));
```

For this reason, the variable nativeUSDCBalance will always be zero, reverting during the Swap.swapThroughUniswap() call, as the amountIn will be zero.

BVSS:

AO:A/AC:L/AX:L/C:N/I:M/A:N/D:N/Y:N/R:N/S:U (5.0)

Recommendation:

It is recommended to transfer the USDC_NATIVE from the data.user to the Loopy contract before the swapThroughUniswap() call.

Remediation Plan:

SOLVED: The Lodestar team solved the issue by implementing the recommended solution.

Commit ID : 7c706d50739de365cff6b46739f5e71bf37db4ac.

4.8 (HAL-08) MISSING INITIAL SWAP CALL TO CONVERT USDCBRIDGED INTO USDCNATIVE BEFORE MINTING - MEDIUM (5.0)

Commit IDs affected:
- ac8c4db810f221a30aed3c2125283deef10173c8

Description:

In the Loopy contract some special logic was implemented as the new USDC market deployed will use USDC_NATIVE while the balancer's vault uses USDC_BRIDGED.

Basically, when looping with USDC_NATIVE, USDC_BRIDGED is flash loaned. Although, as the underlying of the USDC market is USDC_NATIVE, a swap from USDC_BRIDGED to USDC_NATIVE is needed in order to call `mint()` in the USDC market. This initial swap is missing in the code.

For this reason, any `loop()` call with `_useWalletBalance == 0` will revert.

BVSS:

AO:A/AC:L/AX:L/C:N/I:M/A:N/D:N/Y:N/R:N/S:U (5.0)

Recommendation:

It is recommended to implement the missing initial swap from USDC_BRIDGED to USDC_NATIVE in the `Loopy.receiveFlashLoan()` function.

Remediation Plan:

SOLVED: The Lodestar team solved the issue by implementing the recommended solution.

Commit ID : 7c706d50739de365cff6b46739f5e71bf37db4ac.

4.9 (HAL-09) USEWALLETBALANCE PARAMETER IS NOT USED CORRECTLY - LOW (3.9)

Commit IDs affected:
- 0731a52b1b02aa69521f21d30fc11cb37c0a0cc8

Description:

In the Ploopy contract, the function loop() contains the parameter _useWalletBalance. This parameter is used in case that the user wants the contract to mint using their existing wallet balance. Otherwise, the contract should read his existing balance and flash loan to increase his position.

This logic is not correctly implemented, as the contract will always take a flash loan.

For example, with the following loop() parameters:
- _token = USDC
- _amount = 1000e6
- _leverage = 20000
- _useWalletBalance = 1

1000e6 USDC will be transferred to the smart contract, then a flash loan of another 1000e6 will be taken, and 2000e6 USDC in total will be minted for lUSDC.

In the case that, _useWalletBalance is set to 1, a flash loan should not be taken, and the contract should simply call mint() to transfer the cTokens to the user and then call the borrowBehalf() function.

BVSS:

AO:A/AC:L/AX:L/C:N/I:L/A:N/D:L/Y:N/R:N/S:C (3.9)

Recommendation:

It is recommended to correct the logic of the loop() function, so it does not take a flash loan when _useWalletBalance is set to 1.

Remediation Plan:

SOLVED: The Lodestar team solved the issue by implementing the recommended solution.

Commit ID : 164063788549a7b9db59284c36eab7905d76dcc6.

4.10 (HAL-10) BLOCKSPERYEAR ARE NOT CORRECTLY ADJUSTED IN THE RATE MODELS - LOW (3.1)

Commit IDs affected:
- d19be010dea56ba706449e68d394591e97b30916

Description:

Lodestar finance provides a system of lending and borrowing with a dynamic interest rate model that responds to the utilization rate of each asset. The rate model is fundamental in deciding how these interest rates adjust based on utilization. Initially, Compound V2 started with the WhitePaperInterestRateModel, and later on, they introduced the JumpRateModel. The differences between the two are:

- WhitePaperInterestRateModel: This is the initial model proposed in the Compound whitepaper. It follows a linear interest rate model where the borrow rate increases linearly with the utilization rate of the asset. However, it has a utilization "kink" after which the interest rate increases more sharply. This is to prevent over-utilization of assets.
- JumpRateModel: The JumpRateModel introduces an exponential function after the utilization "kink" instead of the linear one in the WhitePaperInterestRateModel. In this model, when the utilization rate reaches a certain point (the "kink"), the borrow rate starts increasing at a much faster, exponential rate. This helps to drastically disincentivize borrowing when liquidity becomes scarce, and encourages lenders to supply more assets.

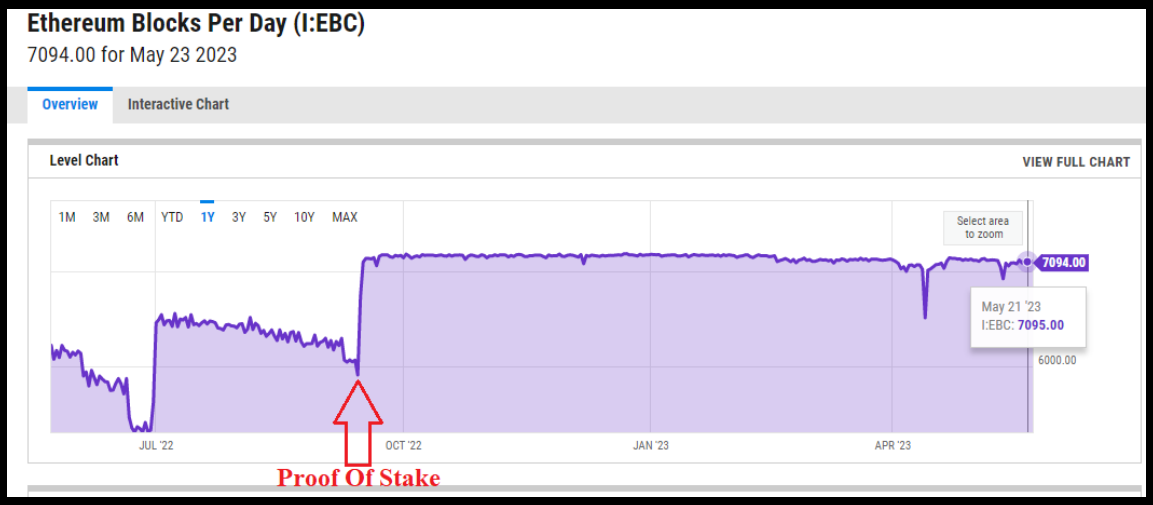
The blocksPerYear constant is a value used in Lodestar and Compound's smart contracts to annualized interest rates.

Interest rates in DeFi platforms are typically expressed as annual percentages (Annual Percentage Rate or APR). In Ethereum, before Proof of

Stake, a new block was added approximately every 13-15 seconds, which leads to a total of around 2,000,000 to 2,500,000 blocks per year. However, in order to simplify calculations in their contracts, Lodestar and Compound chose to use a fixed value of 2102400 for the blocksPerYear constant.

This constant is used to convert the per-block interest rate into an annual rate. For example, if the interest rate is 0.05% per block, the annual interest rate would be 0.05% * blocksPerYear.

While using a constant value simplifies calculations, it does not perfectly reflect the actual number of blocks produced in a year on the Ethereum network, which can fluctuate based on network conditions. In this case, since Proof of Stake was introduced, a block is mined every 12 seconds, and approximately 7100 blocks are mined every day:



If we multiply this value by 365 days, we get 2,591,500 blocksPerYear.

Although, the current blocksPerYear set in the Lodestar's WhitePaperInterestRateModel contract is:
{language=solidity firstnumber="19" caption="WhitePaperInterestRateModel.sol" uint public constant blocksPerYear = 2102400;

And the Lodestar's JumpRateModel contract is:
{language=solidity firstnumber="18" caption="JumpRateModel.sol" uint public constant blocksPerYear = 2628000;

So basically, the WhitePaperInterestRateModel contract assumes that an average of 5760 blocks are mined per day and the JumpRateModel assumes that an average of 7200 blocks are mined per day.

This value is not correctly adjusted, especially in the WhitePaperInterestRateModel contract. This affects the borrow rate and supply rate per block of both Rate Models.

BVSS:

AO:A/AC:L/AX:L/C:N/I:L/A:N/D:N/Y:L/R:N/S:U (3.1)

Recommendation:

It is recommended to adjust the blocksPerYear constant in both Rate Models to either 2628000, assuming that every day an average of 7200 blocks are mined in the Ethereum mainnet or to 2591500, assuming that every day an average of 7100 blocks are mined in the Ethereum mainnet.

Remediation Plan:

RISK ACCEPTED: The Lodestar team accepted the risk of this finding.

4.11 (HAL-11) MINTANDSTAKEGLP() CALL DOES NOT CHECK FOR SLIPPAGE - LOW (2.6)

Commit IDs affected:
- 0731a52b1b02aa69521f21d30fc11cb37c0a0cc8

Description:

The contract Ploopy contains the function loop(). Through these functions, a user can loop any balancer supported, flash-loanable asset. This includes USDC, USDT, DAI, WBTC, FRAX, ARB and plvGLP tokens. Users can supply an existing token from their wallet to a specific leveraged point, or they can leverage an existing balance to a specific point:

Listing 19: Ploopy.sol (Lines 183-188)

```
95 function loop(IERC20 _token, uint256 _amount, uint16 _leverage,
↳ uint16 _useWalletBalance) external {
96   require(allowedTokens[_token], "token not allowed to loop");
97   require(tx.origin == msg.sender, "not an EOA");
98   require(_amount > 0, "amount must be greater than 0");
99   require(_leverage >= DIVISOR && _leverage <= MAX_LEVERAGE, "
↳ invalid leverage, range must be between DIVISOR and MAX_LEVERAGE
↳ values");
100
101   // if the user wants us to mint using their existing wallet
↳ balance (indiciated with 1), then do so.
102   // otherwise, read their existing balance and flash loan to
↳ increase their position
103   if (_useWalletBalance == 1) {
104     // transfer tokens to this contract so we can mint in 1 go.
105     _token.safeTransferFrom(msg.sender, address(this), _amount);
106     emit Transfer(msg.sender, address(this), _amount);
107   }
108
109   uint256 loanAmount;
110   IERC20 _tokenToBorrow;
111
```

```
112   if (_token == PLVGLP) {
113     uint256 _tokenPriceInEth;
114     uint256 _usdcPriceInEth;
115     uint256 _computedAmount;
116 
117     // plvGLP borrows USDC to loop
118     _tokenToBorrow = USDC;
119     _tokenPriceInEth = PRICE_ORACLE.getUnderlyingPrice(address(
 ↳ lTokenMapping[_token]));
120     _usdcPriceInEth = (PRICE_ORACLE.getUnderlyingPrice(address(
 ↳ lUSDC)) / 1e12);
121     _computedAmount = (_amount * (_tokenPriceInEth /
 ↳ _usdcPriceInEth));
122 
123     loanAmount = getNotionalLoanAmountIn1e18(
124       _computedAmount,
125       _leverage
126     );
127   } else {
128     // the rest of the contracts just borrow whatever token is
 ↳ supplied
129     _tokenToBorrow = _token;
130     loanAmount = getNotionalLoanAmountIn1e18(
131       _amount, // we can just send over the exact amount, as we
 ↳ are either looping stables or eth
132       _leverage
133     );
134   }
135 
136   if (_tokenToBorrow.balanceOf(address(BALANCER_VAULT)) <
 ↳ loanAmount) revert FAILED('balancer vault token balance < loan');
137   emit Loan(loanAmount);
138   emit BalanceOf(_tokenToBorrow.balanceOf(address(BALANCER_VAULT))
 ↳ , loanAmount);
139 
140   // check approval to spend USDC (for paying back flashloan).
141   // possibly can omit to save gas as tx will fail with exceed
 ↳ allowance anyway.
142   if (_tokenToBorrow.allowance(msg.sender, address(this)) <
 ↳ loanAmount) revert INVALID_APPROVAL();
143   emit Allowance(_tokenToBorrow.allowance(msg.sender, address(this
 ↳ )), loanAmount);
144 
145   IERC20[] memory tokens = new IERC20[](1);
```

```
146   tokens[0] = _tokenToBorrow;
147
148   uint256[] memory loanAmounts = new uint256[](1);
149   loanAmounts[0] = loanAmount;
150
151   UserData memory userData = UserData({
152     user: msg.sender,
153     tokenAmount: _amount,
154     borrowedToken: _tokenToBorrow,
155     borrowedAmount: loanAmount,
156     tokenToLoop: _token
157   });
158   emit UserDataEvent(msg.sender, _amount, address(_tokenToBorrow),
 ↳  loanAmount, address(_token));
159
160   BALANCER_VAULT.flashLoan(IFlashLoanRecipient(this), tokens,
 ↳ loanAmounts, abi.encode(userData));
161 }
162
163
164 function receiveFlashLoan(
165   IERC20[] memory tokens,
166   uint256[] memory amounts,
167   uint256[] memory feeAmounts,
168   bytes memory userData
169 ) external override nonReentrant {
170   if (msg.sender != address(BALANCER_VAULT)) revert UNAUTHORIZED('
 ↳ balancer vault is not the sender');
171
172   // additional checks?
173
174   UserData memory data = abi.decode(userData, (UserData));
175   if (data.borrowedAmount != amounts[0] || data.borrowedToken !=
 ↳ tokens[0]) revert FAILED('borrowed amounts and/or borrowed tokens
 ↳ do not match initially set values');
176
177   // sanity check: flashloan has no fees
178   if (feeAmounts[0] > 0) revert FAILED('balancer fee > 0');
179
180   // account for some plvGLP specific logic
181   if (data.tokenToLoop == PLVGLP) {
182     // mint GLP. approval needed.
183     uint256 glpAmount = REWARD_ROUTER_V2.mintAndStakeGlp(
184       address(data.borrowedToken),
```

```
185         data.borrowedAmount,
186         0,
187         0
188       );
189       if (glpAmount == 0) revert FAILED('glp=0');
190
191       // TODO whitelist this contract for plvGLP mint
192       // mint plvGLP. approval needed.
193       uint256 _oldPlvglpBal = PLVGLP.balanceOf(address(this));
194       GLP_DEPOSITOR.deposit(glpAmount);
195
196       // check new balances and confirm we properly minted
197       uint256 _newPlvglpBal = PLVGLP.balanceOf(address(this));
198       emit plvGLPBalance(_newPlvglpBal);
199       require(_newPlvglpBal > _oldPlvglpBal, "glp deposit failed,
 ↳ new balance < old balance");
200     }
201
202     uint256 _finalBal;
203
204     // mint our respective token by depositing it into Lodestar's
 ↳ respective lToken contract (approval needed)
205     unchecked {
206       lTokenMapping[data.tokenToLoop].mint(data.tokenToLoop.
 ↳ balanceOf(address(this)));
207       lTokenMapping[data.tokenToLoop].transfer(data.user,
 ↳ lTokenMapping[data.tokenToLoop].balanceOf(address(this)));
208       _finalBal = lTokenMapping[data.tokenToLoop].balanceOf(address(
 ↳ this));
209
210       emit lTokenBalance(_finalBal);
211       require(_finalBal == 0, "lToken balance not 0 at the end of
 ↳ loop");
212     }
213
214     // call borrowBehalf to borrow tokens on behalf of user
215     lTokenMapping[data.tokenToLoop].borrowBehalf(data.borrowedAmount
 ↳ , data.user);
216     // repay loan, where msg.sender = vault
217     data.tokenToLoop.safeTransferFrom(data.user, msg.sender, data.
 ↳ borrowedAmount);
218 }
```

The current loop() implementation performs a flashloan and then calls the

REWARD_ROUTER_V2.mintAndStakeGlp() function:

Listing 20: RewardRouterV2.sol (Line 133)

```
129 function mintAndStakeGlp(address _token, uint256 _amount, uint256
 ↳ _minUsdg, uint256 _minGlp) external nonReentrant returns (uint256)
 ↳  {
130     require(_amount > 0, "RewardRouter: invalid _amount");
131
132     address account = msg.sender;
133     uint256 glpAmount = IGlpManager(glpManager).
 ↳ addLiquidityForAccount(account, account, _token, _amount, _minUsdg
 ↳ , _minGlp);
134     IRewardTracker(feeGlpTracker).stakeForAccount(account, account
 ↳ , glp, glpAmount);
135     IRewardTracker(stakedGlpTracker).stakeForAccount(account,
 ↳ account, feeGlpTracker, glpAmount);
136
137     emit StakeGlp(account, glpAmount);
138
139     return glpAmount;
140 }
```

The mintAndStakeGlp() function contains the following parameters:
- _token: the token to buy GLP with
- _amount: the amount of token to use for the purchase
- _minUsdg: the minimum acceptable USD value of the GLP purchased
- _minGlp: the minimum acceptable GLP amount

These 2 minAmount parameters are set to zero. The _minGlp parameter should be set accordingly in order to prevent any possible slippage. This issue could be paired with ANY USER CAN BORROW ON BEHALF OF ANOTHER USER THAT APPROVED THE PLOOPY CONTRACT issue in order to exploit the protocol.

BVSS:

AO:A/AC:L/AX:M/C:N/I:N/A:N/D:L/Y:L/R:N/S:C (2.6)

Recommendation:

It is recommended to calculate off-chain the _minGlp and use it in the mintAndStakeGlp() function call to mitigate any possible slippage. Below is an example implementation in typescript that calculates the expected GLPMintAmount for a token:
- getExpectedGLPMintAmountForToken

Remediation Plan:

SOLVED: The Lodestar team solved the issue by allowing a maximum of 5% slippage during the mintAndStakeGlp() call.

Commit ID : 164063788549a7b9db59284c36eab7905d76dcc6.

4.12 (HAL-12) LATESTROUNDDATA CALL MAY RETURN STALE RESULTS - LOW (2.5)

Commit IDs affected:
- d19be010dea56ba706449e68d394591e97b30916

Description:

In the PriceOracleProxyETH contract, the function getPriceFromChainlink() is used to retrieve prices from different Chainlink aggregators:

Listing 21: PriceOracleProxyETH.sol (Lines 134-135)

```
128 /**
129  * @notice Get price from ChainLink
130  * @param aggregator The ChainLink aggregator to get the price of
131  * @return The price
132  */
133 function getPriceFromChainlink(AggregatorV3Interface aggregator)
 ↳ internal view returns (uint256) {
134     (uint80 roundId, int256 price, uint startedAt, uint updatedAt,
 ↳  uint80 answeredInRound) = aggregator
135         .latestRoundData();
136     require(roundId == answeredInRound && startedAt == updatedAt,
 ↳ "Price not fresh");
137     require(price > 0, "invalid price");
138
139     // Extend the decimals to 1e18.
140     return uint256(price) * 10 ** (18 - uint256(aggregator.
 ↳ decimals()));
141 }
```

The return values of the latestRoundData() call are:
- roundId: The round ID. Oracles provide periodic data updates to the aggregators. Data feeds are updated in rounds. Rounds are identified by their roundId, which increases with each new round. This increase may not be monotonic. Knowing the roundId of a previous round allows contracts to consume historical data.
- answer: The data that this specific feed provides, in this case, the

price of an asset.
- startedAt: Timestamp of when the round started.
- updatedAt: Timestamp of when the round was updated.
- answeredInRound: The round ID of the round in which the answer was computed.

In the current implementation:
1. getPriceFromChainlink() checks that roundId == answeredInRound but roundId < answeredInRound would also be valid.
2. getPriceFromChainlink() checks that startedAt == updatedAt which does not really check for stale results. The condition should be require(block.timestamp <= updatedAt + GRACE_PERIOD_TIME).

This could lead to stale prices.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:L/Y:N/R:N/S:U (2.5)

Recommendation:

It is recommended to update the getPriceFromChainlink() function as shown below:

Listing 22: PriceOracleProxyETH.sol (Lines 136-138)

```
128 /**
129  * @notice Get price from ChainLink
130  * @param aggregator The ChainLink aggregator to get the price of
131  * @return The price
132  */
133 function getPriceFromChainlink(AggregatorV3Interface aggregator)
 ↳ internal view returns (uint256) {
134     (uint80 roundId, int256 price, uint startedAt, uint updatedAt,
 ↳  uint80 answeredInRound) = aggregator
135         .latestRoundData();
136     require(answeredInRound >= roundID, "Stale price");
137     require(price > 0, "invalid price");
138     require(block.timestamp <= updatedAt + GRACE_PERIOD_TIME, "
 ↳ Stale price");
```

```
139
140     // Extend the decimals to 1e18.
141     return uint256(price) * 10 ** (18 - uint256(aggregator.
 ↳ decimals()));
142 }
```

Remediation Plan:

RISK ACCEPTED: The Lodestar team accepted the risk of this finding.

4.13 (HAL-13) PLOOPY CONTRACT ASSUMES THAT THE BALANCER FLASHLOAN FEE WILL ALWAYS BE ZERO - LOW (2.5)

Commit IDs affected:
- 0731a52b1b02aa69521f21d30fc11cb37c0a0cc8

Description:

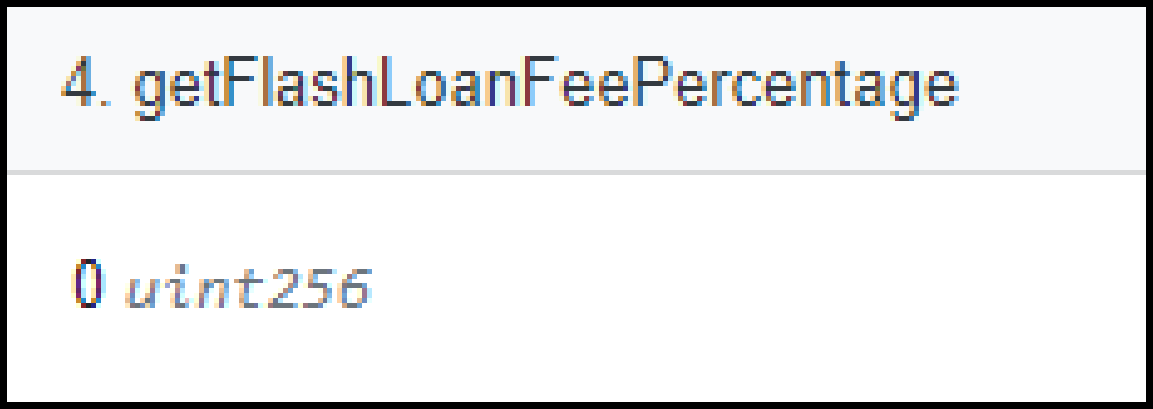
In the Ploopy contract, the function loop() takes a flash loan of the _token from the Balancer Vault, uses the flash loaned tokens to mint more cTokens and then borrows more _token to repay the flash loan. Although, the repayment is done assuming that the flash loan fee will be always zero. This is currently true, although if the Balancer ever updates this value, the Ploopy contract will stop working and any loop() call will always revert:

FlashLoans.sol

Arbiscan (One) > Vault > contracts > vault > FlashLoans.sol

```
49          IERC20 previousToken = IERC20(0);
50
51          for (uint256 i = 0; i < tokens.length; ++i) {
52              IERC20 token = tokens[i];
53              uint256 amount = amounts[i];
54
55              _require(token > previousToken, token == IERC20(0) ? Errors.ZERO_TOKEN : Errors.UNSORTED_TOKENS);
56              previousToken = token;
57
58              preLoanBalances[i] = token.balanceOf(address(this));
59              feeAmounts[i] = _calculateFlashLoanFeeAmount(amount);
60
61              _require(preLoanBalances[i] >= amount, Errors.INSUFFICIENT_FLASH_LOAN_BALANCE);
62              token.safeTransfer(address(recipient), amount);
63          }
64
65          recipient.receiveFlashLoan(tokens, amounts, feeAmounts, userData);
66
67          for (uint256 i = 0; i < tokens.length; ++i) {
68              IERC20 token = tokens[i];
69              uint256 preLoanBalance = preLoanBalances[i];
70
71              // Checking for loan repayment first (without accounting for fees) makes for simpler debugging, and results
72              // in more accurate revert reasons if the flash loan protocol fee percentage is zero.
73              uint256 postLoanBalance = token.balanceOf(address(this));
74              _require(postLoanBalance >= preLoanBalance, Errors.INVALID_POST_LOAN_BALANCE);
75
76              // No need for checked arithmetic since we know the loan was fully repaid.
77              uint256 receivedFeeAmount = postLoanBalance - preLoanBalance;
78              _require(receivedFeeAmount >= feeAmounts[i], Errors.INSUFFICIENT_FLASH_LOAN_FEE_AMOUNT);
79
80              _payFeeAmount(token, receivedFeeAmount);
81              emit FlashLoan(recipient, token, amounts[i], receivedFeeAmount);
82          }
83      }
84  }
```

Balancer ProtocolFeesCollector contract



BVSS:

AO:A/AC:L/AX:L/C:N/I:L/A:N/D:N/Y:N/R:N/S:U (2.5)

Recommendation:

Consider calling the getFlashLoanFeePercentage() function every time a new flash loan is executed to calculate the total amount that should be borrowed to correctly repay the flash loan.

Remediation Plan:

SOLVED: The Lodestar team solved the issue by implementing the recommended solution.

Commit ID : 7c706d50739de365cff6b46739f5e71bf37db4ac.

4.14 (HAL-14) LOOPING CAN NOT BE DISABLED AFTER BEING ENABLED - LOW (2.5)

Commit IDs affected:
- d19be010dea56ba706449e68d394591e97b30916

Description:

The Comptroller contract contains the function enableLooping() which is used by the users to allow/disallow the looping contract to borrow or redeem on their behalf:

Listing 23: Comptroller.sol (Lines 519,520)

```
511 /**
512  * @notice Toggle for user to allow for looping contract to borrow
 ↳  and redeem on their behalf.
513  * @param state The requested state for the sender to be set to.
514  * @dev Cannot be called by smart contracts.
515  */
516 function enableLooping(bool state) external override returns (bool
 ↳ ) {
517     require(tx.origin == msg.sender, "!EoA");
518     loopEnabled[msg.sender] = state;
519     if (!loopEnabled[msg.sender]) {
520         revert("FAILED");
521     } else {
522         return true;
523     }
524 }
```

Although, as can be seen in the code above, once a user calls enableLooping (true), the user will not be able to set it back to false as loopEnabled [msg.sender] would be set to false and in the next line, the if (! loopEnabled[msg.sender]) would be entered and the call would revert.

BVSS:

AO:A/AC:L/AX:L/C:N/I:L/A:N/D:N/Y:N/R:N/S:U (2.5)

Recommendation:

It is recommended to update the enableLooping() function as shown below:

Listing 24: Comptroller.sol (Line 518)

```
511 /**
512  * @notice Toggle for user to allow for looping contract to borrow
 ↳  and redeem on their behalf.
513  * @param state The requested state for the sender to be set to.
514  * @dev Cannot be called by smart contracts.
515  */
516 function enableLooping(bool state) external override returns (bool
 ↳ ) {
517     require(tx.origin == msg.sender, "!EoA");
518     loopEnabled[msg.sender] = state;
519     return true;
520 }
```

Remediation Plan:

SOLVED: The Lodestar team solved the issue by implementing the recommended solution.

Commit ID : 241ad9405b6b0d950eca14faeb3a7bf2fd18c824.

4.15 (HAL-15) WRONG REQUIRE CHECK MESSAGE IN SETAGGREGATORS FUNCTION - LOW (2.5)

Commit IDs affected:
- 569f6945bf254587ff645c17584fcf79d8fcc8cc

Description:

The PriceOracleProxyETH contract implements the function _setAggregators():

Listing 25: PriceOracleProxyETH.sol (Line 249)

```
238 /**
239  * @notice Set ChainLink aggregators for multiple cTokens
240  * @param cTokenAddresses The list of cTokens
241  * @param sources The list of ChainLink aggregator sources
242  * @param bases The list of ChainLink aggregator bases
243  */
244 function _setAggregators(
245     address[] calldata cTokenAddresses,
246     address[] calldata sources,
247     AggregatorBase[] calldata bases
248 ) external {
249     require(msg.sender == admin || msg.sender == guardian, "only
 ↳ the admin or guardian may set the aggregators");
250     require(cTokenAddresses.length == sources.length &&
 ↳ cTokenAddresses.length == bases.length, "mismatched data");
251     for (uint256 i = 0; i < cTokenAddresses.length; i++) {
252         if (sources[i] != address(0)) {
253             require(msg.sender == admin, "Only the admin or
 ↳ guardian can clear the aggregators");
254         }
255         aggregators[cTokenAddresses[i]] = AggregatorInfo({
256             source: AggregatorV3Interface(sources[i]),
257             base: bases[i]
258         });
259         emit AggregatorUpdated(cTokenAddresses[i], sources[i],
 ↳ bases[i]);
```

```
260     }
261 }
```

In one of the require statements is mentioned “Only the admin or guardian may set the aggregators”. Although, with the current implementation the admin can set the aggregator while clearing the aggregators can be done by either the admin or the guardian.

BVSS:

AO:A/AC:L/AX:L/C:N/I:L/A:N/D:N/Y:N/R:N/S:U (2.5)

Recommendation:

Consider updating the _setAggregators() function require statement as shown below:

Listing 26: PriceOracleProxyETH.sol (Line 249)

```
238 /**
239  * @notice Set ChainLink aggregators for multiple cTokens
240  * @param cTokenAddresses The list of cTokens
241  * @param sources The list of ChainLink aggregator sources
242  * @param bases The list of ChainLink aggregator bases
243  */
244 function _setAggregators(
245     address[] calldata cTokenAddresses,
246     address[] calldata sources,
247     AggregatorBase[] calldata bases
248 ) external {
249     require(msg.sender == admin || msg.sender == guardian, "only
 ↳ the admin may set the aggregators");
250     require(cTokenAddresses.length == sources.length &&
 ↳ cTokenAddresses.length == bases.length, "mismatched data");
251     for (uint256 i = 0; i < cTokenAddresses.length; i++) {
252         if (sources[i] != address(0)) {
253             require(msg.sender == admin, "Only the admin or
 ↳ guardian can clear the aggregators");
254         }
255         aggregators[cTokenAddresses[i]] = AggregatorInfo({
```

```
256             source: AggregatorV3Interface(sources[i]),
257             base: bases[i]
258         });
259         emit AggregatorUpdated(cTokenAddresses[i], sources[i],
 ↳ bases[i]);
260     }
261 }
```

Remediation Plan:

SOLVED: The Lodestar team solved the issue by removing the guardian role and using instead the Ownable2Step library.

Commit ID : 241ad9405b6b0d950eca14faeb3a7bf2fd18c824.

4.16 (HAL-16) BLOCK.TIMESTAMP IS USED TO SET THE SWAP DEADLINE - INFORMATIONAL (0.0)

Commit IDs affected:
- ac8c4db810f221a30aed3c2125283deef10173c8

Description:

In the Swap contract, block.timestamp is being used as the deadline parameter in the different swap calls:

Listing 27: Swap.sol (Lines 19,39)

```
 8 function swapThroughUniswap(
 9     address token0Address,
10     address token1Address,
11     uint256 amountIn,
12     uint256 minAmountOut
13 ) public returns (uint256) {
14     uint24 poolFee = 3000;
15
16     ISwapRouter.ExactInputParams memory params = ISwapRouter.
↳ ExactInputParams({
17         path: abi.encodePacked(token0Address, poolFee,
↳ token1Address),
18         recipient: address(this),
19         deadline: block.timestamp,
20         amountIn: amountIn,
21         amountOutMinimum: minAmountOut
22     });
23
24     uint256 amountOut = UNI_ROUTER.exactInput(params);
25     return amountOut;
26 }
27
28 //NOTE:Only involves swapping tokens for tokens, any operations
↳ involving ETH will be wrap/unwrap calls to WETH contract
29 function swapThroughSushiswap(
30     address token0Address,
```

```
31      address token1Address,
32      uint256 amountIn,
33      uint256 minAmountOut
34  ) public {
35      address[] memory path = new address[](2);
36      path[0] = token0Address;
37      path[1] = token1Address;
38      address to = address(this);
39      uint256 deadline = block.timestamp;
40      SUSHI_ROUTER.
↳ swapExactTokensForTokensSupportingFeeOnTransferTokens(amountIn,
↳ minAmountOut, path, to, deadline);
41  }
```

Because of this, a malicious miner/sequencer can hold the transaction and execute it whenever wanted in order to acquire some profit from it.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)

Recommendation:

Consider adding a deadline parameter to the loop() function instead of the hard-coded block.timestamp.

Remediation Plan:

ACKNOWLEDGED: The Lodestar team acknowledged this finding.

4.17 (HAL-17) FUNCTION GETLODEPRICE(ADDRESS POOLADDRESS) CAN BE REMOVED - INFORMATIONAL (0.0)

Commit IDs affected:
- d19be010dea56ba706449e68d394591e97b30916

Description:

The PriceOracleProxyETH contract implements the following functions:

Listing 28: PriceOracleProxyETH.sol

```
153 /**
154  * @notice Get price of LODE token
155  * @return the price of LODE in wei
156  */
157 function getLodePrice() public view returns (uint256) {
158     uint256 price = SushiOracleInterface(lodeOracle).price();
159     return price;
160 }
```

Listing 29: PriceOracleProxyETH.sol

```
178 /**
179  * @notice Get price of LODE token
180  * @param poolAddress the address of the LODE token contract
181  * @return the price of LODE in wei
182  */
183 function getLodePrice(address poolAddress) public view returns (
 ↳ uint256) {
184     uint256 price = SushiOracleInterface(poolAddress).price();
185     return price;
186 }
```

As both functions implement the same logic, consider removing the getLodePrice(address poolAddress) function from the PriceOracleProxyETH contract to reduce deployment gas costs.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)

Recommendation:

It is recommended to remove the `getLodePrice(address poolAddress)` function from the `PriceOracleProxyETH` contract.

Remediation Plan:

SOLVED: The `Lodestar team` solved the issue by removing the `getLodePrice(address poolAddress)` function from the `PriceOracleProxyETH` contract.

Commit ID : `241ad9405b6b0d950eca14faeb3a7bf2fd18c824`.

4.18 (HAL-18) SUSHIORACLE PRICE CAN BE MANIPULATED - INFORMATIONAL (0.0)

Commit IDs affected:
- d19be010dea56ba706449e68d394591e97b30916

Description:

The SushiOracle contract makes use of the balances of a pool to calculate the price of a token:

Listing 30: SushiOracle.sol (Lines 38,39)

```
30 function getTokenBalance(address tokenAddress) public view returns
↳  (uint256) {
31     uint256 balance = EIP20Interface(tokenAddress).balanceOf(
↳ poolContract);
32     return balance;
33 }
34
35 function price() public view returns (uint256) {
36     uint256 balanceA = getTokenBalance(tokenA);
37     uint256 balanceB = getTokenBalance(tokenB);
38     uint256 price = (balanceA * 1e18) / balanceB;
39     return price;
40 }
```

This contract is basically a spot price oracle. One primary concern in the usage of spot price oracles is the potential for price manipulation. In this case, the price could be easily manipulated by taking a flash loan and depositing into the pool.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)

Recommendation:

It is recommended to avoid using spot price oracles for any price calculation.

Remediation Plan:

SOLVED: The Lodestar team states that this contract is only used to display the price of LODE on the front-end.

4.19 (HAL-19) PLOOPY CONTRACT IS MISSING A FUNCTION TO ADD NEW MARKETS - INFORMATIONAL (0.0)

Commit IDs affected:
- 0731a52b1b02aa69521f21d30fc11cb37c0a0cc8

Description:

The Ploopy contract requires having approved all the different Lodestar markets in order to be able to call the mint() function: lTokenMapping[data.tokenToLoop].mint(data.tokenToLoop.balanceOf(address(this)));

This approval is done directly in the constructor:

Listing 31: Ploopy.sol (Lines 65-71)

```
19 constructor() {
20   // initialize decimals for each token
21   decimals[USDC] = 6;
22   decimals[USDT] = 6;
23   decimals[WBTC] = 8;
24   decimals[DAI] = 18;
25   decimals[FRAX] = 18;
26   decimals[ARB] = 18;
27   decimals[PLVGLP] = 18;
28
29   // set the allowed tokens in the constructor
30   // we can add/remove these with owner functions later
31   allowedTokens[USDC] = true;
32   allowedTokens[USDT] = true;
33   allowedTokens[WBTC] = true;
34   allowedTokens[DAI] = true;
35   allowedTokens[FRAX] = true;
36   allowedTokens[ARB] = true;
37   allowedTokens[PLVGLP] = true;
38
39   // map tokens to lTokens
```

```
40    lTokenMapping[USDC] = lUSDC;
41    lTokenMapping[USDT] = lUSDT;
42    lTokenMapping[WBTC] = lWBTC;
43    lTokenMapping[DAI] = lDAI;
44    lTokenMapping[FRAX] = lFRAX;
45    lTokenMapping[ARB] = lARB;
46    lTokenMapping[PLVGLP] = lPLVGLP;
47
48    // approve glp contracts to spend USDC for minting GLP
49    USDC.approve(address(REWARD_ROUTER_V2), type(uint256).max);
50    USDC.approve(address(GLP), type(uint256).max);
51    USDC.approve(address(GLP_MANAGER), type(uint256).max);
52    // approve GlpDepositor to spend GLP for minting plvGLP
53    sGLP.approve(address(GLP_DEPOSITOR), type(uint256).max);
54    GLP.approve(address(GLP_DEPOSITOR), type(uint256).max);
55    sGLP.approve(address(REWARD_ROUTER_V2), type(uint256).max);
56    GLP.approve(address(REWARD_ROUTER_V2), type(uint256).max);
57    // approve balancer vault
58    USDC.approve(address(VAULT), type(uint256).max);
59    USDT.approve(address(VAULT), type(uint256).max);
60    WBTC.approve(address(VAULT), type(uint256).max);
61    DAI.approve(address(VAULT), type(uint256).max);
62    FRAX.approve(address(VAULT), type(uint256).max);
63    ARB.approve(address(VAULT), type(uint256).max);
64    // approve lTokens to be minted using underlying
65    PLVGLP.approve(address(lPLVGLP), type(uint256).max);
66    USDC.approve(address(lUSDC), type(uint256).max);
67    USDT.approve(address(lUSDT), type(uint256).max);
68    WBTC.approve(address(lWBTC), type(uint256).max);
69    DAI.approve(address(lDAI), type(uint256).max);
70    FRAX.approve(address(lFRAX), type(uint256).max);
71    ARB.approve(address(lARB), type(uint256).max);
72  }
```

The `Ploopy` contract is missing a function that:

1. Approves the `VAULT`.
2. Approves the new Lodestar `MARKET`.
3. Sets the decimals of the new token to `loop()`.
4. Sets the `lTokenMapping` mapping.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)

Recommendation:

It is recommended to add a new `onlyOwner` function that allows to add a new Lodestar market into the `Ploopy` contract. The function `addToken()` can be used for this purpose.

Remediation Plan:

SOLVED: The `Lodestar team` solved the issue by implementing the recommended solution.

Commit ID : 164063788549a7b9db59284c36eab7905d76dcc6.

4.20 (HAL-20) LACK OF A DOUBLE-STEP TRANSFEROWNERSHIP PATTERN - INFORMATIONAL (0.0)

Commit IDs affected:
- d19be010dea56ba706449e68d394591e97b30916

Description:

The PriceOracleProxyETH contract acts as a proxy for other Oracle contracts. Currently, this contract contains 2 different access control roles:
- admin: Can set the guardian, transfer the admin privilege to another account, add a LodeOracle, add a PlvGLPOracle and add or remove different aggregators.
- guardian: Can add new aggregators.

The function _setAdmin() specifically is used to transfer this role to a new account:

Listing 32: PriceOracleProxyETH.sol (Line 212)

```
210 /**
211  * @notice Set admin for price oracle proxy
212  * @param _admin The new admin
213  */
214 function _setAdmin(address _admin) external {
215     require(msg.sender == admin, "only the admin may set new admin
 ↳ ");
216     admin = _admin;
217     emit SetAdmin(admin);
218 }
```

If the nominated EOA account is not a valid account, it is entirely possible that the admin may accidentally transfer this role to an uncontrolled account, losing the access to all functions that can only be managed by the admin.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)

Recommendation:

Consider removing the admin role and instead using the Ownable2Step library. This library implements a two-step process where the owner nominates an account and the nominated account needs to call an acceptOwnership() function for the transfer of the ownership to fully succeed. This ensures the nominated EOA account is a valid and active account:

Listing 33: Ownable2Step.sol (Lines 52-56)

```
 1 // SPDX-License-Identifier: MIT
 2 // OpenZeppelin Contracts (last updated v4.8.0) (access/
↳ Ownable2Step.sol)
 3
 4 pragma solidity ^0.8.0;
 5
 6 import "./Ownable.sol";
 7
 8 /**
 9  * @dev Contract module which provides access control mechanism,
↳ where
10  * there is an account (an owner) that can be granted exclusive
↳ access to
11  * specific functions.
12  *
13  * By default, the owner account will be the one that deploys the
↳ contract. This
14  * can later be changed with {transferOwnership} and {
↳ acceptOwnership}.
15  *
16  * This module is used through inheritance. It will make available
↳  all functions
17  * from parent (Ownable).
18  */
19 abstract contract Ownable2Step is Ownable {
20     address private _pendingOwner;
21
22     event OwnershipTransferStarted(address indexed previousOwner,
↳ address indexed newOwner);
```

```
23
24      /**
25       * @dev Returns the address of the pending owner.
26       */
27      function pendingOwner() public view virtual returns (address)
↳ {
28          return _pendingOwner;
29      }
30
31      /**
32       * @dev Starts the ownership transfer of the contract to a new
↳  account. Replaces the pending transfer if there is one.
33       * Can only be called by the current owner.
34       */
35      function transferOwnership(address newOwner) public virtual
↳ override onlyOwner {
36          _pendingOwner = newOwner;
37          emit OwnershipTransferStarted(owner(), newOwner);
38      }
39
40      /**
41       * @dev Transfers ownership of the contract to a new account
↳ (`newOwner`) and deletes any pending owner.
42       * Internal function without access restriction.
43       */
44      function _transferOwnership(address newOwner) internal virtual
↳  override {
45          delete _pendingOwner;
46          super._transferOwnership(newOwner);
47      }
48
49      /**
50       * @dev The new owner accepts the ownership transfer.
51       */
52      function acceptOwnership() external {
53          address sender = _msgSender();
54          require(pendingOwner() == sender, "Ownable2Step: caller is
↳  not the new owner");
55          _transferOwnership(sender);
56      }
57  }
```

Remediation Plan:

SOLVED: The Lodestar team solved the issue by implementing the recommended solution.

Commit ID : 241ad9405b6b0d950eca14faeb3a7bf2fd18c824.

4.21 (HAL-21) ALLOWANCE CHECK CAN BE REMOVED FROM PLOOPY.LOOP FUNCTION - INFORMATIONAL (0.0)

Commit IDs affected:
- 0731a52b1b02aa69521f21d30fc11cb37c0a0cc8

Description:

The function loop() contains the following allowance check:

Listing 34: Ploopy.sol (Line 142)

```
112 function loop(IERC20 _token, uint256 _amount, uint16 _leverage,
 ↳ uint16 _useWalletBalance) external {
113   require(allowedTokens[_token], "token not allowed to loop");
114   require(tx.origin == msg.sender, "not an EOA");
115   require(_amount > 0, "amount must be greater than 0");
116   require(_leverage >= DIVISOR && _leverage <= MAX_LEVERAGE, "
 ↳ invalid leverage, range must be between DIVISOR and MAX_LEVERAGE
 ↳ values");
117
118   // if the user wants us to mint using their existing wallet
 ↳ balance (indiciated with 1), then do so.
119   // otherwise, read their existing balance and flash loan to
 ↳ increase their position
120   if (_useWalletBalance == 1) {
121     // transfer tokens to this contract so we can mint in 1 go.
122     _token.safeTransferFrom(msg.sender, address(this), _amount);
123     emit Transfer(msg.sender, address(this), _amount);
124   }
125
126   uint256 loanAmount;
127   IERC20 _tokenToBorrow;
128
129   if (_token == PLVGLP) {
130     uint256 _tokenPriceInEth;
131     uint256 _usdcPriceInEth;
132     uint256 _computedAmount;
133
```

```
134     // plvGLP borrows USDC to loop
135     _tokenToBorrow = USDC;
136     _tokenPriceInEth = PRICE_ORACLE.getUnderlyingPrice(address(
 ↳ lTokenMapping[_token]));
137     _usdcPriceInEth = (PRICE_ORACLE.getUnderlyingPrice(address(
 ↳ lUSDC)) / 1e12);
138     _computedAmount = (_amount * (_tokenPriceInEth /
 ↳ _usdcPriceInEth));
139
140     loanAmount = getNotionalLoanAmountIn1e18(
141       _computedAmount,
142       _leverage
143     );
144   } else {
145     // the rest of the contracts just borrow whatever token is
 ↳ supplied
146     _tokenToBorrow = _token;
147     loanAmount = getNotionalLoanAmountIn1e18(
148       _amount, // we can just send over the exact amount, as we
 ↳ are either looping stables or eth
149       _leverage
150     );
151   }
152
153   if (_tokenToBorrow.balanceOf(address(BALANCER_VAULT)) <
 ↳ loanAmount) revert FAILED('balancer vault token balance < loan');
154   emit Loan(loanAmount);
155   emit BalanceOf(_tokenToBorrow.balanceOf(address(BALANCER_VAULT))
 ↳ , loanAmount);
156
157   // check approval to spend USDC (for paying back flashloan).
158   // possibly can omit to save gas as tx will fail with exceed
 ↳ allowance anyway.
159   if (_tokenToBorrow.allowance(msg.sender, address(this)) <
 ↳ loanAmount) revert INVALID_APPROVAL();
160   emit Allowance(_tokenToBorrow.allowance(msg.sender, address(this
 ↳ )), loanAmount);
161
162   IERC20[] memory tokens = new IERC20[](1);
163   tokens[0] = _tokenToBorrow;
164
165   uint256[] memory loanAmounts = new uint256[](1);
166   loanAmounts[0] = loanAmount;
167
```

```
168   UserData memory userData = UserData({
169     user: msg.sender,
170     tokenAmount: _amount,
171     borrowedToken: _tokenToBorrow,
172     borrowedAmount: loanAmount,
173     tokenToLoop: _token
174   });
175   emit UserDataEvent(msg.sender, _amount, address(_tokenToBorrow),
 ↳  loanAmount, address(_token));
176
177   BALANCER_VAULT.flashLoan(IFlashLoanRecipient(this), tokens,
 ↳ loanAmounts, abi.encode(userData));
178 }
```

The require check can be removed as it is redundant and increases the gas costs. The repay of the flash loan will revert in case that the allowance is not enough.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)

Recommendation:

Consider removing the allowance check in the Ploopy.loop() function.

Remediation Plan:

SOLVED: The Lodestar team solved the issue by removing the allowance check in the Ploopy.loop() function.

Commit ID : 164063788549a7b9db59284c36eab7905d76dcc6.

RECOMMENDATIONS OVERVIEW

1. Add a `require` statement in the `Ploopy.receiveFlashLoan()` function that checks that `tx.origin` is equal to the `userData.user` field passed as parameter to this function.
2. Correct in the `CEtherDelegator` contract, the different `abi.encodeWithSignature()` calls to use the correct function signatures.
3. Ensure that markets are never empty by minting small cToken (or equivalent) balances at the time of market creation.
4. Update the `Ploopy.receiveFlashLoan()` function so, in the case that the `tokenToLoop` is plvGLP, the repayment is done with USDC.
5. Refactor the `loop()` function and the `Ploopy` contract in general, so the looping considers the current position of the user in Lodestar to calculate the leverage.
6. Update the `PriceOracleProxyETH.getPriceFromChainlink()` function as suggested in order to avoid stale prices.
7. Update the `PriceOracleProxyETH.getUnderlyingPrice()` function, so it also checks that the sequencer is not down when `cTokenAddress == lplvGLPAddress`.
8. Transfer the USDC_NATIVE from the `data.user` to the `Loopy` contract before the `swapThroughUniswap()` call.
9. Implement the missing initial swap from USDC_BRIDGED to USDC_NATIVE in the `Loopy.receiveFlashLoan()` function.
10. Call the `ProtocolFeesCollector.getFlashLoanFeePercentage()` function every time a new flash loan is executed in the `Ploopy` contract to calculate the total amount that should be borrowed to correctly repay the flash loan.
11. Correct the logic of the `loop()` function, so it does not take a flash loan when `_useWalletBalance` is set to 1.
12. Add a new `onlyOwner` function to the `Ploopy` contract that allows to add a new Lodestar market into the `Ploopy` contract. The function `addToken()` can be used for this purpose.
13. Adjust the `blocksPerYear` constant in both Rate Models to either `2628000`, assuming that every day an average of 7200 blocks are mined in the Ethereum mainnet or to `2591500`, assuming that every day an average of 7100 blocks are mined in the Ethereum mainnet.
14. Consider removing the `admin` role and instead using the `Ownable2Step` library in the `PriceOracleProxyETH` contract.

15. Calculate off chain the `_minGlp` amount and use it in the `mintAndStakeGlp()` function call in the `Ploopy` contract to mitigate any possible slippage.
16. Update the `enableLooping()` function, so it can be disabled.
17. Consider updating the mentioned `require` statement in the `PriceOracleProxyETH._setAggregators()` function.
18. Consider adding a `deadline` parameter to the `loop()` function instead of the hard-coded `block.timestamp`.
19. Remove the `getLodePrice(address poolAddress)` function from the `PriceOracleProxyETH` contract.
20. Avoid using spot price oracles for any price calculation.
21. Consider removing the allowance check in the `Ploopy.loop()` function.

PLVGLPORACLE RISKS

The PlvGLPOracle is a proprietary oracle created by Lodestar that determines the price of PlvGLP tokens. The calculation for GLP price is conducted through the use of the getAum() function from the GLPManager contract:

Listing 35: plvGLPOracle.sol (Line 61)

```
59 function getGLPPrice() public view returns (uint256) {
60     //retrieve the minimized AUM from GLP Manager Contract
61     uint256 glpAUM = GLPManagerInterface(GLPManager).getAum(false)
↳ ;
62     //retrieve the total supply of GLP
63     uint256 glpSupply = ERC20Interface(GLP).totalSupply();
64     //GLP Price = AUM / Total Supply
65     uint256 price = (glpAUM * DECIMAL_DIFFERENCE) / glpSupply;
66     return price;
67 }
```

This calculation is a central element within the Lodestar codebase. We note that since the GLPManager contract is not under the scope of this assessment, the security of the PlvGLPOracle contract cannot be entirely guaranteed. However, upon thorough examination, we find the PlvGLPOracle to demonstrate strong resilience against manipulation due to the following reasons:

1. The PlvGLP price is calculated by computing the TWAP price of plvGLP based on the current price of GLP and the moving average of the plvGLP/GLP exchange rate.
2. The current, cumulative and average indices are updated in the PlvGLPOracle contract when required conditions are met. If the price fails to update, the posted price will fall back to the last previously accepted average index. Access is restricted to only whitelisted addresses. The index is only updated when the requested update is within +/- 1% of the previously accepted index.
3. The getGLPPrice() function makes use of the GLPManager.getAum(false) function. This function internally calls _vault.getMinPrice(token) which internally queries a custom Price Feed: GLPManager.getAum(false)

Listing 36: GLPManager.sol (Line 150)

```
136 function getAum(bool maximise) public view returns (uint256) {
137     uint256 length = vault.allWhitelistedTokensLength();
138     uint256 aum = aumAddition;
139     uint256 shortProfits = 0;
140     IVault _vault = vault;
141
142     for (uint256 i = 0; i < length; i++) {
143         address token = vault.allWhitelistedTokens(i);
144         bool isWhitelisted = vault.whitelistedTokens(token);
145
146         if (!isWhitelisted) {
147             continue;
148         }
149
150         uint256 price = maximise ? _vault.getMaxPrice(token) :
 ↳ _vault.getMinPrice(token);
151         uint256 poolAmount = _vault.poolAmounts(token);
152         uint256 decimals = _vault.tokenDecimals(token);
153
154         if (_vault.stableTokens(token)) {
155             aum = aum.add(poolAmount.mul(price).div(10 ** decimals
 ↳ ));
156         } else {
157             // add global short profit / loss
158             uint256 size = _vault.globalShortSizes(token);
159
160             if (size > 0) {
161                 (uint256 delta, bool hasProfit) =
 ↳ getGlobalShortDelta(token, price, size);
162                 if (!hasProfit) {
163                     // add losses from shorts
164                     aum = aum.add(delta);
165                 } else {
166                     shortProfits = shortProfits.add(delta);
167                 }
168             }
169
170             aum = aum.add(_vault.guaranteedUsd(token));
171
172             uint256 reservedAmount = _vault.reservedAmounts(token)
 ↳ ;
173             aum = aum.add(poolAmount.sub(reservedAmount).mul(price
 ↳ ).div(10 ** decimals));
```

```
174             }
175         }
176
177         aum = shortProfits > aum ? 0 : aum.sub(shortProfits);
178         return aumDeduction > aum ? 0 : aum.sub(aumDeduction);
179     }
```

_vault.getMinPrice(token)

Listing 37: Vault.sol

```
1510 function getMinPrice(address _token) public override view returns
 ↳ (uint256) {
1511     return IVaultPriceFeed(priceFeed).getPrice(_token, false,
 ↳ includeAmmPrice, useSwapPricing);
1512 }
```

priceFeed.getPrice(_token, false, includeAmmPrice, useSwapPricing)

Listing 38: VaultPriceFeed.sol (Lines 149,165,180,183)

```
148 function getPrice(address _token, bool _maximise, bool
 ↳ _includeAmmPrice, bool /* _useSwapPricing */) public override view
 ↳  returns (uint256) {
149     uint256 price = useV2Pricing ? getPriceV2(_token, _maximise,
 ↳ _includeAmmPrice) : getPriceV1(_token, _maximise, _includeAmmPrice
 ↳ );
150
151     uint256 adjustmentBps = adjustmentBasisPoints[_token];
152     if (adjustmentBps > 0) {
153         bool isAdditive = isAdjustmentAdditive[_token];
154         if (isAdditive) {
155             price = price.mul(BASIS_POINTS_DIVISOR.add(
 ↳ adjustmentBps)).div(BASIS_POINTS_DIVISOR);
156         } else {
157             price = price.mul(BASIS_POINTS_DIVISOR.sub(
 ↳ adjustmentBps)).div(BASIS_POINTS_DIVISOR);
158         }
159     }
160
161     return price;
162 }
```

```
163
164 function getPriceV1(address _token, bool _maximise, bool
 ↳ _includeAmmPrice) public view returns (uint256) {
165     uint256 price = getPrimaryPrice(_token, _maximise);
166
167     if (_includeAmmPrice && isAmmEnabled) {
168         uint256 ammPrice = getAmmPrice(_token);
169         if (ammPrice > 0) {
170             if (_maximise && ammPrice > price) {
171                 price = ammPrice;
172             }
173             if (!_maximise && ammPrice < price) {
174                 price = ammPrice;
175             }
176         }
177     }
178
179     if (isSecondaryPriceEnabled) {
180         price = getSecondaryPrice(_token, price, _maximise);
181     }
182
183     if (strictStableTokens[_token]) {
184         uint256 delta = price > ONE_USD ? price.sub(ONE_USD) :
 ↳ ONE_USD.sub(price);
185         if (delta <= maxStrictPriceDeviation) {
186             return ONE_USD;
187         }
188
189         // if _maximise and price is e.g. 1.02, return 1.02
190         if (_maximise && price > ONE_USD) {
191             return price;
192         }
193
194         // if !_maximise and price is e.g. 0.98, return 0.98
195         if (!_maximise && price < ONE_USD) {
196             return price;
197         }
198
199         return ONE_USD;
200     }
201
202     uint256 _spreadBasisPoints = spreadBasisPoints[_token];
203
204     if (_maximise) {
```

```
205         return price.mul(BASIS_POINTS_DIVISOR.add(
 ↳ _spreadBasisPoints)).div(BASIS_POINTS_DIVISOR);
206     }
207
208     return price.mul(BASIS_POINTS_DIVISOR.sub(_spreadBasisPoints))
 ↳ .div(BASIS_POINTS_DIVISOR);
209 }
```

4. The primary Price Feed used in the VaultPriceFeed contract will retrieve the lowest price of the asset of the last 3 Chainlink rounds:

Listing 39: VaultPriceFeed.sol (Line 304)

```
287 function getPrimaryPrice(address _token, bool _maximise) public
 ↳ override view returns (uint256) {
288     address priceFeedAddress = priceFeeds[_token];
289     require(priceFeedAddress != address(0), "VaultPriceFeed:
 ↳ invalid price feed");
290
291     if (chainlinkFlags != address(0)) {
292         bool isRaised = IChainlinkFlags(chainlinkFlags).getFlag(
 ↳ FLAG_ARBITRUM_SEQ_OFFLINE);
293         if (isRaised) {
294                 // If flag is raised we shouldn't perform any
 ↳ critical operations
295             revert("Chainlink feeds are not being updated");
296         }
297     }
298
299     IPriceFeed priceFeed = IPriceFeed(priceFeedAddress);
300
301     uint256 price = 0;
302     uint80 roundId = priceFeed.latestRound();
303
304     for (uint80 i = 0; i < priceSampleSpace; i++) {
305         if (roundId <= i) { break; }
306         uint256 p;
307
308         if (i == 0) {
309             int256 _p = priceFeed.latestAnswer();
310             require(_p > 0, "VaultPriceFeed: invalid price");
```

```
311             p = uint256(_p);
312         } else {
313             (, int256 _p, , ,) = priceFeed.getRoundData(roundId -
 ↳ i);
314             require(_p > 0, "VaultPriceFeed: invalid price");
315             p = uint256(_p);
316         }
317
318         if (price == 0) {
319             price = p;
320             continue;
321         }
322
323         if (_maximise && p > price) {
324             price = p;
325             continue;
326         }
327
328         if (!_maximise && p < price) {
329             price = p;
330         }
331     }
332
333     require(price > 0, "VaultPriceFeed: could not fetch price");
334     // normalise price precision
335     uint256 _priceDecimals = priceDecimals[_token];
336     return price.mul(PRICE_PRECISION).div(10 ** _priceDecimals);
337 }
```

5. On top of that, a secondary Price Feed is used:

Listing 40: VaultPriceFeed.sol (Line 341)

```
339 function getSecondaryPrice(address _token, uint256 _referencePrice
 ↳ , bool _maximise) public view returns (uint256) {
340     if (secondaryPriceFeed == address(0)) { return _referencePrice
 ↳ ; }
341     return ISecondaryPriceFeed(secondaryPriceFeed).getPrice(_token
 ↳ , _referencePrice, _maximise);
342 }
```

FastPriceFeed.getPrice()

Listing 41: FastPriceFeed.sol (Lines 344-352)

```
306 // under regular operation, the fastPrice (prices[token]) is
 ↳ returned and there is no spread returned from this function,
307 // though VaultPriceFeed might apply its own spread
308 //
309 // if the fastPrice has not been updated within priceDuration then
 ↳  it is ignored and only _refPrice with a spread is used (spread:
 ↳ spreadBasisPointsIfInactive)
310 // in case the fastPrice has not been updated for
 ↳ maxPriceUpdateDelay then the _refPrice with a larger spread is
 ↳ used (spread: spreadBasisPointsIfChainError)
311 //
312 // there will be a spread from the _refPrice to the fastPrice in
 ↳ the following cases:
313 // - in case isSpreadEnabled is set to true
314 // - in case the maxDeviationBasisPoints between _refPrice and
 ↳ fastPrice is exceeded
315 // - in case watchers flag an issue
316 // - in case the cumulativeFastDelta exceeds the
 ↳ cumulativeRefDelta by the maxCumulativeDeltaDiff
317 function getPrice(address _token, uint256 _refPrice, bool
 ↳ _maximise) external override view returns (uint256) {
318     if (block.timestamp > lastUpdatedAt.add(maxPriceUpdateDelay))
 ↳ {
319         if (_maximise) {
320             return _refPrice.mul(BASIS_POINTS_DIVISOR.add(
 ↳ spreadBasisPointsIfChainError)).div(BASIS_POINTS_DIVISOR);
321         }
322
323         return _refPrice.mul(BASIS_POINTS_DIVISOR.sub(
 ↳ spreadBasisPointsIfChainError)).div(BASIS_POINTS_DIVISOR);
324     }
325
326     if (block.timestamp > lastUpdatedAt.add(priceDuration)) {
327         if (_maximise) {
328             return _refPrice.mul(BASIS_POINTS_DIVISOR.add(
 ↳ spreadBasisPointsIfInactive)).div(BASIS_POINTS_DIVISOR);
329         }
330
331         return _refPrice.mul(BASIS_POINTS_DIVISOR.sub(
 ↳ spreadBasisPointsIfInactive)).div(BASIS_POINTS_DIVISOR);
332     }
```

```
333
334     uint256 fastPrice = prices[_token];
335     if (fastPrice == 0) { return _refPrice; }
336
337     uint256 diffBasisPoints = _refPrice > fastPrice ? _refPrice.
 ↳ sub(fastPrice) : fastPrice.sub(_refPrice);
338     diffBasisPoints = diffBasisPoints.mul(BASIS_POINTS_DIVISOR).
 ↳ div(_refPrice);
339
340     // create a spread between the _refPrice and the fastPrice if
 ↳ the maxDeviationBasisPoints is exceeded
341     // or if watchers have flagged an issue with the fast price
342     bool hasSpread = !favorFastPrice(_token) || diffBasisPoints >
 ↳ maxDeviationBasisPoints;
343
344     if (hasSpread) {
345         // return the higher of the two prices
346         if (_maximise) {
347             return _refPrice > fastPrice ? _refPrice : fastPrice;
348         }
349
350         // return the lower of the two prices
351         return _refPrice < fastPrice ? _refPrice : fastPrice;
352     }
353
354     return fastPrice;
355 }
```

6. Another measure against price manipulation is that as _maximise is set to false in the getAum(false) call, the FastPriceFeed.getPrice() function will return the lower of the two prices in case of deviation/manipulation.

Risks:

1. Update of the includeAmmPrice state variable from false to true in the Vault contract.
2. Update of the useSwapPricing state variable from false to true in the Vault contract.
3. Update of the useV2Pricing state variable from false to true in the

VaultFeed contract.

4. Update of the isSpreadEnabled state variable from false to true in the FastPriceFeed contract.

AUTOMATED TESTING

7.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the smart contracts in scope. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified the smart contracts in the repository and was able to compile them correctly into their ABIS and binary format, Slither was run against the contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Slither results:

BaseJumpRateModelV2.sol

```
INFO:Detectors:
BaseJumpRateModelV2.getSupplyRate(uint256,uint256,uint256,uint256) (contracts/BaseJumpRateModelV2.sol#122-132) performs a multiplication on the result of a division:
        - rateToPool = (borrowRate * oneMinusReserveFactor) / BASE (contracts/BaseJumpRateModelV2.sol#130)
        - (utilizationRate(cash,borrows,reserves) * rateToPool) / BASE (contracts/BaseJumpRateModelV2.sol#131)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Pragma version^0.8.10 (contracts/BaseJumpRateModelV2.sol#2) allows old versions
Pragma version^0.8.10 (contracts/InterestRateModel.sol#2) allows old versions
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
BaseJumpRateModelV2 (contracts/BaseJumpRateModelV2.sol#11-154) does not implement functions:
        - InterestRateModel.getBorrowRate(uint256,uint256,uint256) (contracts/InterestRateModel.sol#19)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
BaseJumpRateModelV2.owner (contracts/BaseJumpRateModelV2.sol#19) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Detectors:
In a function BaseJumpRateModelV2.getBorrowRateInternal(uint256,uint256,uint256) (contracts/BaseJumpRateModelV2.sol#102-112) variable BaseJumpRateModelV2.BASE (contracts/BaseJumpRateModelV2.sol#14) is read multiple times
In a function BaseJumpRateModelV2.getBorrowRateInternal(uint256,uint256,uint256) (contracts/BaseJumpRateModelV2.sol#102-112) variable BaseJumpRateModelV2.kink (contracts/BaseJumpRateModelV2.sol#45) is read multiple times
In a function BaseJumpRateModelV2.getSupplyRate(uint256,uint256,uint256,uint256) (contracts/BaseJumpRateModelV2.sol#122-132) variable BaseJumpRateModelV2.BASE (contracts/BaseJumpRateModelV2.sol#14) is read multiple times
In a function BaseJumpRateModelV2.updateJumpRateModelInternal(uint256,uint256,uint256,uint256) (contracts/BaseJumpRateModelV2.sol#141-153) variable BaseJumpRateModelV2.blocksPerYear (contracts/BaseJumpRateModelV2.sol#25) is read multiple
 times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

CarefulMath.sol

```
INFO:Detectors:
CarefulMath.addThenSubUInt(uint256,uint256,uint256) (contracts/CarefulMath.sol#76-84) is never used and should be removed
CarefulMath.addUInt(uint256,uint256) (contracts/CarefulMath.sol#63-71) is never used and should be removed
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) is never used and should be removed
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) is never used and should be removed
CarefulMath.subUInt(uint256,uint256) (contracts/CarefulMath.sol#52-58) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.10 (contracts/CarefulMath.sol#2) allows old versions
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

CDaiDelegate.sol

```
INFO:Detectors:
CErc20.doTransferIn(address,uint256) (contracts/CErc20.sol#190-219) uses arbitrary from in transferFrom: token.transferFrom(from,address(this),amount) (contracts/CErc20.sol#195)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#arbitrary-from-in-transferfrom
INFO:Detectors:
Function CErc20Delegate.constructor() (contracts/CErc20Delegate.sol#15) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/strange_setter.md
INFO:Detectors:
CErc20Delegate._becomeImplementation(bytes) (contracts/CErc20Delegate.sol#21-31) uses a Boolean constant improperly:
        -false (contracts/CErc20Delegate.sol#26)
CErc20Delegate._resignImplementation() (contracts/CErc20Delegate.sol#36-43) uses a Boolean constant improperly:
        -false (contracts/CErc20Delegate.sol#38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#misuse-of-a-boolean-constant
INFO:Detectors:
GemLike (contracts/CDaiDelegate.sol#195-199) has incorrect ERC20 function interface:GemLike.approve(address,uint256) (contracts/CDaiDelegate.sol#196)
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transfer(address,uint256) (contracts/EIP20NonStandardInterface.sol#35)
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transferFrom(address,address,uint256) (contracts/EIP20NonStandardInterface.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
INFO:Detectors:
CToken.accrueInterest() (contracts/CToken.sol#325-381) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/CToken.sol#331)
CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/CToken.sol#293)
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CToken.sol#27-60) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/CToken.sol#36)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == NO_ERROR,LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/CToken.sol#811)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) uses a dangerous strict equality:
        - require(bool,string)(CTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
CDaiDelegate.mul(uint256,uint256) (contracts/CDaiDelegate.sol#180-182) uses a dangerous strict equality:
        - require(bool,string)(y == 0 || (z = x * y) / y == x,mul-overflow) (contracts/CDaiDelegate.sol#181)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
```

```
INFO:Detectors:
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745):
        External calls:
        - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#737)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
                - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
        CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - totalSupply = totalSupply - redeemTokens (contracts/CToken.sol#548)
        CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87) can be used in cross function reentrancies:
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
CToken._addReservesFresh(uint256).actualAddAmount (contracts/CToken.sol#1055) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
CDaiDelegate._becomeImplementation(address,address) (contracts/CDaiDelegate.sol#43-68) ignores return value by pot.drip() (contracts/CDaiDelegate.sol#64)
CDaiDelegate._resignImplementation() (contracts/CDaiDelegate.sol#73-93) ignores return value by pot.drip() (contracts/CDaiDelegate.sol#82)
CDaiDelegate.accrueInterest() (contracts/CDaiDelegate.sol#102-108) ignores return value by PotLike(potAddress).drip() (contracts/CDaiDelegate.sol#104)
CErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CErc20.sol#26-41) ignores return value by EIP20Interface(underlying).totalSupply() (contracts/CErc20.sol#40)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
CErc20 doTransferIn parameter from is not related to msg.sender token.transferFrom(from,address(this),amount) (contracts/CErc20.sol#195)
Reference: https://ventral.digital/posts/2022/8/18/sznsdaos-bountyboard-unauthorized-transferfrom-vulnerability
INFO:Detectors:
CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) shadows:
        - CTokenInterface.NewReserveGuardian(address,address) (contracts/CTokenInterfaces.sol#195) (event)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
CErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8).underlying_ (contracts/CErc20.sol#27) lacks a zero-check on :
        - underlying = underlying_ (contracts/CErc20.sol#39)
CToken._setPendingAdmin(address).newPendingAdmin (contracts/CToken.sol#902) lacks a zero-check on :
        - pendingAdmin = newPendingAdmin (contracts/CToken.sol#912)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in CDaiDelegate._becomeImplementation(address,address) (contracts/CDaiDelegate.sol#43-68):
        External calls:
        - daiJoin = daiJoin.daiJoin() (contracts/CDaiDelegate.sol#47)
        - vat = daiJoin.vat() (contracts/CDaiDelegate.sol#48)
        State variables written after the call(s):
        - daiJoinAddress = daiJoinAddress_ (contracts/CDaiDelegate.sol#52)
        - potAddress = potAddress_ (contracts/CDaiDelegate.sol#53)
        - vatAddress = address(vat) (contracts/CDaiDelegate.sol#54)
Reentrancy in CDaiDelegate.accrueInterest() (contracts/CDaiDelegate.sol#102-108):
        External calls:
        - PotLike(potAddress).drip() (contracts/CDaiDelegate.sol#104)
        State variables written after the call(s):
        - super.accrueInterest() (contracts/CDaiDelegate.sol#107)
                - accrualBlockNumber = currentBlockNumber (contracts/CToken.sol#372)
        - super.accrueInterest() (contracts/CDaiDelegate.sol#107)
                - borrowIndex = borrowIndexNew (contracts/CToken.sol#373)
        - super.accrueInterest() (contracts/CDaiDelegate.sol#107)
                - totalBorrows = totalBorrowsNew (contracts/CToken.sol#374)
        - super.accrueInterest() (contracts/CDaiDelegate.sol#107)
                - totalReserves = totalReservesNew (contracts/CToken.sol#375)
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#629)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#630)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#631)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        State variables written after the call(s):
        - accountTokens[minter] = accountTokens[minter] + mintTokens (contracts/CToken.sol#442)
        - totalSupply = totalSupply + mintTokens (contracts/CToken.sol#441)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - accountTokens[redeemer] = accountTokens[redeemer] - redeemTokens (contracts/CToken.sol#549)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#713)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#714)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
Reentrancy in CToken.seizeInternal(address,address,address,uint256) (contracts/CToken.sol#855-892):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        State variables written after the call(s):
        - accountTokens[borrower] = accountTokens[borrower] - seizeTokens (contracts/CToken.sol#885)
        - accountTokens[liquidator] = accountTokens[liquidator] + liquidatorSeizeTokens (contracts/CToken.sol#886)
        - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        - totalSupply = totalSupply - protocolSeizeTokens (contracts/CToken.sol#884)
Reentrancy in CToken.transferTokens(address,address,address,uint256) (contracts/CToken.sol#71-115):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/CToken.sol#73)
        State variables written after the call(s):
        - accountTokens[src] = srcTokensNew (contracts/CToken.sol#100)
        - accountTokens[dst] = dstTokensNew (contracts/CToken.sol#101)
        - transferAllowances[src][spender] = allowanceNew (contracts/CToken.sol#105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in CDaiDelegate.accrueInterest() (contracts/CDaiDelegate.sol#102-108):
        External calls:
        - PotLike(potAddress).drip() (contracts/CDaiDelegate.sol#104)
        Event emitted after the call(s):
        - AccrueInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/CToken.sol#378)
                - super.accrueInterest() (contracts/CDaiDelegate.sol#107)
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        Event emitted after the call(s):
        - Borrow(borrower,borrowAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#642)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
                - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        Event emitted after the call(s):
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
```

CErc20.sol

```
INFO:Detectors:
Function CErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CErc20.sol#26-41) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initialize.md
INFO:Detectors:
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transfer(address,uint256) (contracts/EIP20NonStandardInterface.sol#35)
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transferFrom(address,address,uint256) (contracts/EIP20NonStandardInterface.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
INFO:Detectors:
CToken.accrueInterest() (contracts/CToken.sol#325-381) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/CToken.sol#331)
CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/CToken.sol#293)
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CToken.sol#27-60) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/CToken.sol#36)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == NO_ERROR,LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/CToken.sol#811)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) uses a dangerous strict equality:
        - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745):
        External calls:
        - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#737)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
              - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
              - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
              - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
              - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
              - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
        CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
              - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - totalSupply = totalSupply - redeemTokens (contracts/CToken.sol#548)
        CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87) can be used in cross function reentrancies:
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
CToken._addReservesFresh(uint256).actualAddAmount (contracts/CToken.sol#1055) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
CErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CErc20.sol#26-41) ignores return value by EIP20Interface(underlying).totalSupply() (contracts/CErc20.sol#40)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) shadows:
        - CTokenInterfaceNewReserveGuardian(address,address) (contracts/CTokenInterfaces.sol#195) (event)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
CToken._setPendingAdmin(address).newPendingAdmin (contracts/CToken.sol#902) lacks a zero-check on :
              - pendingAdmin = newPendingAdmin (contracts/CToken.sol#912)
CErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8).underlying_ (contracts/CErc20.sol#27) lacks a zero-check on :
              - underlying = underlying_ (contracts/CErc20.sol#39)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#629)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#630)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#631)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        State variables written after the call(s):
        - accountTokens[minter] = accountTokens[minter] + mintTokens (contracts/CToken.sol#442)
        - totalSupply = totalSupply + mintTokens (contracts/CToken.sol#441)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - accountTokens[redeemer] = accountTokens[redeemer] - redeemTokens (contracts/CToken.sol#549)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#713)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#714)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
Reentrancy in CToken.seizeInternal(address,address,address,uint256) (contracts/CToken.sol#855-892):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        State variables written after the call(s):
        - accountTokens[borrower] = accountTokens[borrower] - seizeTokens (contracts/CToken.sol#885)
        - accountTokens[liquidator] = accountTokens[liquidator] + liquidatorSeizeTokens (contracts/CToken.sol#886)
        - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        - totalSupply = totalSupply - protocolSeizeTokens (contracts/CToken.sol#884)
Reentrancy in CToken.transferTokens(address,address,address,uint256) (contracts/CToken.sol#71-115):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/CToken.sol#73)
        State variables written after the call(s):
        - accountTokens[src] = srcTokensNew (contracts/CToken.sol#100)
        - accountTokens[dst] = dstTokensNew (contracts/CToken.sol#101)
        - transferAllowances[src][spender] = allowanceNew (contracts/CToken.sol#105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        Event emitted after the call(s):
        - Borrow(borrower,borrowAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#642)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
              - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
              - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
              - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
              - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        Event emitted after the call(s):
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
              - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
              - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
              - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
              - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
              - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(cTokenCollateral),seizeTokens) (contracts/CToken.sol#824)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        Event emitted after the call(s):
        - Mint(minter,actualMintAmount,mintTokens) (contracts/CToken.sol#445)
        - Transfer(address(this),minter,mintTokens) (contracts/CToken.sol#446)
```

```
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        Event emitted after the call(s):
        - Redeem(redeemer,redeemAmount,redeemTokens) (contracts/CToken.sol#561)
        - Transfer(redeemer,address(this),redeemTokens) (contracts/CToken.sol#560)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Function CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) has a dubious typecast: address<=address
Function CToken._acceptAdmin() (contracts/CToken.sol#925-945) has a dubious typecast: address<=address
Function CErc20.doTransferIn(address,uint256) (contracts/CErc20.sol#190-219) has a dubious typecast: address<=EIP20Interface
Function ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) has a dubious typecast: uint256<=uint256
Function ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) has a dubious typecast: uint256<=uint256
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
CErc20.doTransferIn(address,uint256) (contracts/CErc20.sol#190-219) uses assembly
        - INLINE ASM (contracts/CErc20.sol#198-213)
CErc20.doTransferOut(address,uint256) (contracts/CErc20.sol#230-252) uses assembly
        - INLINE ASM (contracts/CErc20.sol#235-250)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) is never used and should be removed
ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#92-94) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#88-90) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#152-154) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#156-158) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#140-142) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#144-146) is never used and should be removed
ExponentialNoError.div_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#160-162) is never used and should be removed
ExponentialNoError.fraction(uint256,uint256) (contracts/ExponentialNoError.sol#168-170) is never used and should be removed
ExponentialNoError.greaterThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#67-69) is never used and should be removed
ExponentialNoError.isZeroExp(ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#74-76) is never used and should be removed
ExponentialNoError.lessThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#53-55) is never used and should be removed
ExponentialNoError.lessThanOrEqualExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#60-62) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#124-126) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#128-130) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#112-114) is never used and should be removed
ExponentialNoError.mul_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#132-134) is never used and should be removed
ExponentialNoError.safe224(uint256,string) (contracts/ExponentialNoError.sol#78-81) is never used and should be removed
ExponentialNoError.safe32(uint256,string) (contracts/ExponentialNoError.sol#83-86) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#104-106) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#100-102) is never used and should be removed
ExponentialNoError.sub_(uint256,uint256) (contracts/ExponentialNoError.sol#108-110) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.10 (contracts/CErc20.sol#2) allows old versions
Pragma version^0.8.10 (contracts/CToken.sol#2) allows old versions
Pragma version^0.8.10 (contracts/CTokenInterfaces.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ComptrollerInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20Interface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20NonStandardInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ErrorReporter.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ExponentialNoError.sol#2) allows old versions
Pragma version^0.8.10 (contracts/InterestRateModel.sol#2) allows old versions
Pragma version0.8.10 (contracts/Whitelist.sol#3) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function CErc20._addReserves(uint256) (contracts/CErc20.sol#165-167) is not in mixedCase
Function CErc20._delegateCompLikeTo(address) (contracts/CErc20.sol#259-262) is not in mixedCase
Function CToken._setPendingAdmin(address) (contracts/CToken.sol#902-918) is not in mixedCase
Function CToken._acceptAdmin() (contracts/CToken.sol#925-945) is not in mixedCase
Function CToken._setComptroller(ComptrollerInterface) (contracts/CToken.sol#952-969) is not in mixedCase
Function CToken._setReserveGuardian(address) (contracts/CToken.sol#976-979) is not in mixedCase
Function CToken._setReserveFactor(uint256) (contracts/CToken.sol#998-1002) is not in mixedCase
Function CToken._reduceReserves(uint256) (contracts/CToken.sol#1093-1097) is not in mixedCase
Function CToken._setInterestRateModel(InterestRateModel) (contracts/CToken.sol#1154-1158) is not in mixedCase
Variable CTokenStorage._notEntered (contracts/CTokenInterfaces.sol#13) is not in mixedCase
Constant CTokenStorage.borrowRateMaxMantissa (contracts/CTokenInterfaces.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant CTokenStorage.reserveFactorMaxMantissa (contracts/CTokenInterfaces.sol#34) is not in UPPER_CASE_WITH_UNDERSCORES
Function CTokenInterface._setPendingAdmin(address) (contracts/CTokenInterfaces.sol#260) is not in mixedCase
Function CTokenInterface._acceptAdmin() (contracts/CTokenInterfaces.sol#262) is not in mixedCase
Function CTokenInterface._setComptroller(ComptrollerInterface) (contracts/CTokenInterfaces.sol#264) is not in mixedCase
Function CTokenInterface._setReserveGuardian(address) (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Parameter CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Function CTokenInterface._setReserveFactor(uint256) (contracts/CTokenInterfaces.sol#268) is not in mixedCase
Function CTokenInterface._reduceReserves(uint256) (contracts/CTokenInterfaces.sol#270) is not in mixedCase
Function CTokenInterface._setInterestRateModel(InterestRateModel) (contracts/CTokenInterfaces.sol#272) is not in mixedCase
Function CErc20Interface._addReserves(uint256) (contracts/CTokenInterfaces.sol#311) is not in mixedCase
Function CDelegatorInterface._setImplementation(address,bool,bytes) (contracts/CTokenInterfaces.sol#333-337) is not in mixedCase
Function CDelegatorInterface._becomeImplementation(bytes) (contracts/CTokenInterfaces.sol#346) is not in mixedCase
Function CDelegateInterface._resignImplementation() (contracts/CTokenInterfaces.sol#351) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncate(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#37-40) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncateAddUint(ExponentialNoError.Exp,uint256,uint256) (contracts/ExponentialNoError.sol#45-48) is not in mixedCase
Constant ExponentialNoError.expScale (contracts/ExponentialNoError.sol#12) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.doubleScale (contracts/ExponentialNoError.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.mantissaOne (contracts/ExponentialNoError.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter Whitelist.updateWhitelist(address,bool)._address (contracts/Whitelist.sol#11) is not in mixedCase
Parameter Whitelist.updateWhitelist(address,bool)._isActive (contracts/Whitelist.sol#12) is not in mixedCase
Parameter Whitelist.getWhitelisted(address)._address (contracts/Whitelist.sol#17) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable CToken.seize(address,address,uint256).seizeTokens (contracts/CToken.sol#839) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CTokenInterface.seize(address,address,uint256).seizeTokens (contracts/CTokenInterfaces.sol#256) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface).seizeTokens (contracts/CToken.sol#806) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.seizeInternal(address,address,address,uint256).seizeTokens (contracts/CToken.sol#855) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is never used in CErc20 (contracts/CErc20.sol#15-263)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
Function CErc20.doTransferIn(address,uint256) (contracts/CErc20.sol#190-219) contains magic number: 32
Function CErc20.doTransferOut(address,uint256) (contracts/CErc20.sol#230-252) contains magic number: 32
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
CDelegationStorage.implementation (contracts/CTokenInterfaces.sol#318) should be constant
```

CErc20Delegate.sol

```
INFO:Detectors:
Function CErc20Delegate.constructor() (contracts/CErc20Delegate.sol#15) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/strange_setter.md
INFO:Detectors:
CErc20Delegate._becomeImplementation(bytes) (contracts/CErc20Delegate.sol#21-31) uses a Boolean constant improperly:
        -false (contracts/CErc20Delegate.sol#26)
CErc20Delegate._resignImplementation() (contracts/CErc20Delegate.sol#36-43) uses a Boolean constant improperly:
        -false (contracts/CErc20Delegate.sol#38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#misuse-of-a-boolean-constant
INFO:Detectors:
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transfer(address,uint256) (contracts/EIP20NonStandardInterface.sol#35)
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transferFrom(address,address,uint256) (contracts/EIP20NonStandardInterface.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
INFO:Detectors:
CToken.accrueInterest() (contracts/CToken.sol#325-381) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/CToken.sol#331)
CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/CToken.sol#293)
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CToken.sol#27-60) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/CToken.sol#36)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == NO_ERROR,LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/CToken.sol#811)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) uses a dangerous strict equality:
        - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745):
        External calls:
        - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#737)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
                - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
```

```
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
        CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - totalSupply = totalSupply - redeemTokens (contracts/CToken.sol#548)
        CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87) can be used in cross function reentrancies:
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
CToken._addReservesFresh(uint256).actualAddAmount (contracts/CToken.sol#1055) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
CErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CErc20.sol#26-41) ignores return value by EIP20Interface(underlying).totalSupply() (contracts/CErc20.sol#40)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) shadows:
        - CTokenInterfaceNewReserveGuardian(address,address) (contracts/CTokenInterfaces.sol#195) (event)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
CErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8).underlying_ (contracts/CErc20.sol#27) lacks a zero-check on :
                - underlying = underlying_ (contracts/CErc20.sol#39)
CToken._setPendingAdmin(address).newPendingAdmin (contracts/CToken.sol#902) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (contracts/CToken.sol#912)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#629)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#630)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#631)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        State variables written after the call(s):
        - accountTokens[minter] = accountTokens[minter] + mintTokens (contracts/CToken.sol#442)
        - totalSupply = totalSupply + mintTokens (contracts/CToken.sol#441)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - accountTokens[redeemer] = accountTokens[redeemer] - redeemTokens (contracts/CToken.sol#549)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#713)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#714)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
Reentrancy in CToken.seizeInternal(address,address,address,uint256) (contracts/CToken.sol#855-892):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        State variables written after the call(s):
        - accountTokens[borrower] = accountTokens[borrower] - seizeTokens (contracts/CToken.sol#885)
        - accountTokens[liquidator] = accountTokens[liquidator] + liquidatorSeizeTokens (contracts/CToken.sol#886)
        - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        - totalSupply = totalSupply - protocolSeizeTokens (contracts/CToken.sol#884)
Reentrancy in CToken.transferTokens(address,address,address,uint256) (contracts/CToken.sol#71-115):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/CToken.sol#73)
        State variables written after the call(s):
        - accountTokens[src] = srcTokensNew (contracts/CToken.sol#100)
        - accountTokens[dst] = dstTokensNew (contracts/CToken.sol#101)
        - transferAllowances[src][spender] = allowanceNew (contracts/CToken.sol#105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        Event emitted after the call(s):
        - Borrow(borrower,borrowAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#642)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
                - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        Event emitted after the call(s):
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(cTokenCollateral),seizeTokens) (contracts/CToken.sol#824)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        Event emitted after the call(s):
        - Mint(minter,actualMintAmount,mintTokens) (contracts/CToken.sol#445)
        - Transfer(address(this),minter,mintTokens) (contracts/CToken.sol#446)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        Event emitted after the call(s):
        - Redeem(redeemer,redeemAmount,redeemTokens) (contracts/CToken.sol#561)
        - Transfer(redeemer,address(this),redeemTokens) (contracts/CToken.sol#560)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Function CErc20.doTransferIn(address,uint256) (contracts/CErc20.sol#190-219) has a dubious typecast: address<=EIP20Interface
Function CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) has a dubious typecast: address<=address
Function CToken._acceptAdmin() (contracts/CToken.sol#925-945) has a dubious typecast: address<=address
Function ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) has a dubious typecast: uint256<=uint256
Function ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) has a dubious typecast: uint256<=uint256
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
CErc20.doTransferIn(address,uint256) (contracts/CErc20.sol#190-219) uses assembly
        - INLINE ASM (contracts/CErc20.sol#198-213)
CErc20.doTransferOut(address,uint256) (contracts/CErc20.sol#230-252) uses assembly
        - INLINE ASM (contracts/CErc20.sol#235-250)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
INFO:Detectors:
ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) is never used and should be removed
ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#92-94) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#88-90) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#152-154) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#156-158) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#140-142) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#144-146) is never used and should be removed
ExponentialNoError.div_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#160-162) is never used and should be removed
ExponentialNoError.fraction(uint256,uint256) (contracts/ExponentialNoError.sol#168-170) is never used and should be removed
ExponentialNoError.greaterThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#67-69) is never used and should be removed
ExponentialNoError.isZeroExp(ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#74-76) is never used and should be removed
ExponentialNoError.lessThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#53-55) is never used and should be removed
ExponentialNoError.lessThanOrEqualExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#60-62) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#124-126) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#128-130) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#112-114) is never used and should be removed
ExponentialNoError.mul_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#132-134) is never used and should be removed
ExponentialNoError.safe224(uint256,string) (contracts/ExponentialNoError.sol#78-81) is never used and should be removed
ExponentialNoError.safe32(uint256,string) (contracts/ExponentialNoError.sol#83-86) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#104-106) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#100-102) is never used and should be removed
ExponentialNoError.sub_(uint256,uint256) (contracts/ExponentialNoError.sol#108-110) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.10 (contracts/CErc20.sol#2) allows old versions
Pragma version^0.8.10 (contracts/CErc20Delegate.sol#2) allows old versions
Pragma version^0.8.10 (contracts/CToken.sol#2) allows old versions
Pragma version^0.8.10 (contracts/CTokenInterfaces.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ComptrollerInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20Interface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20NonStandardInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ErrorReporter.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ExponentialNoError.sol#2) allows old versions
Pragma version^0.8.10 (contracts/InterestRateModel.sol#2) allows old versions
Pragma version0.8.10 (contracts/Whitelist.sol#3) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function CErc20._addReserves(uint256) (contracts/CErc20.sol#165-167) is not in mixedCase
Function CErc20._delegateCompLikeTo(address) (contracts/CErc20.sol#259-262) is not in mixedCase
Function CErc20Delegate._becomeImplementation(bytes) (contracts/CErc20Delegate.sol#21-31) is not in mixedCase
Function CErc20Delegate._resignImplementation() (contracts/CErc20Delegate.sol#36-43) is not in mixedCase
Function CToken._setPendingAdmin(address) (contracts/CToken.sol#902-918) is not in mixedCase
Function CToken._acceptAdmin() (contracts/CToken.sol#925-945) is not in mixedCase
Function CToken._setComptroller(ComptrollerInterface) (contracts/CToken.sol#952-969) is not in mixedCase
Function CToken._setReserveGuardian(address) (contracts/CToken.sol#976-979) is not in mixedCase
Function CToken._setReserveFactor(uint256) (contracts/CToken.sol#998-1002) is not in mixedCase
Function CToken._reduceReserves(uint256) (contracts/CToken.sol#1093-1097) is not in mixedCase
Function CToken._setInterestRateModel(InterestRateModel) (contracts/CToken.sol#1154-1158) is not in mixedCase
Variable CToken5torage._notEntered (contracts/CTokenInterfaces.sol#13) is not in mixedCase
Constant CTokenStorage.borrowRateMaxMantissa (contracts/CTokenInterfaces.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant CTokenStorage.reserveFactorMaxMantissa (contracts/CTokenInterfaces.sol#34) is not in UPPER_CASE_WITH_UNDERSCORES
Function CTokenInterface._setPendingAdmin(address) (contracts/CTokenInterfaces.sol#260) is not in mixedCase
Function CTokenInterface._acceptAdmin() (contracts/CTokenInterfaces.sol#262) is not in mixedCase
Function CTokenInterface._setComptroller(ComptrollerInterface) (contracts/CTokenInterfaces.sol#264) is not in mixedCase
Function CTokenInterface._setReserveGuardian(address) (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Parameter CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Function CTokenInterface._setReserveFactor(uint256) (contracts/CTokenInterfaces.sol#268) is not in mixedCase
Function CTokenInterface._reduceReserves(uint256) (contracts/CTokenInterfaces.sol#270) is not in mixedCase
Function CTokenInterface._setInterestRateModel(InterestRateModel) (contracts/CTokenInterfaces.sol#272) is not in mixedCase
Function CErc20Interface._addReserves(uint256) (contracts/CTokenInterfaces.sol#311) is not in mixedCase
Function CDelegatorInterface._setImplementation(address,bool,bytes) (contracts/CTokenInterfaces.sol#333-337) is not in mixedCase
Function CDelegateInterface._becomeImplementation(bytes) (contracts/CTokenInterfaces.sol#346) is not in mixedCase
Function CDelegateInterface._resignImplementation() (contracts/CTokenInterfaces.sol#351) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncate(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#37-40) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncateAddUInt(ExponentialNoError.Exp,uint256,uint256) (contracts/ExponentialNoError.sol#45-48) is not in mixedCase
Constant ExponentialNoError.expScale (contracts/ExponentialNoError.sol#12) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.doubleScale (contracts/ExponentialNoError.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.mantissaOne (contracts/ExponentialNoError.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter Whitelist.updateWhitelist(address,bool)._address (contracts/Whitelist.sol#11) is not in mixedCase
Parameter Whitelist.updateWhitelist(address,bool)._isActive (contracts/Whitelist.sol#12) is not in mixedCase
Parameter Whitelist.getWhitelisted(address)._address (contracts/Whitelist.sol#17) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "data (contracts/CErc20Delegate.sol#23)" inCErc20Delegate (contracts/CErc20Delegate.sol#11-44)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable CToken.seize(address,address,uint256).seizeTokens (contracts/CToken.sol#839) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.seizeInternal(address,address,address,uint256).seizeTokens (contracts/CToken.sol#855) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CTokenInterface.seize(address,address,uint256).seizeTokens (contracts/CTokenInterfaces.sol#256) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface).seizeTokens (contracts/CToken.sol#806) is too similar to CToken.seizeInternal(address,address,address,uint256).seizeToken (contracts/CToken.sol#855)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is never used in CErc20Delegate (contracts/CErc20Delegate.sol#11-44)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
Function CErc20.doTransferIn(address,uint256) (contracts/CErc20.sol#190-219) contains magic number: 32
Function CErc20.doTransferOut(address,uint256) (contracts/CErc20.sol#230-252) contains magic number: 32
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
CDelegationStorage.implementation (contracts/CTokenInterfaces.sol#318) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

CErc20Delegator.sol

```
INFO:Detectors:
CErc20Delegator.delegateTo(address,bytes) (contracts/CErc20Delegator.sol#500-508) uses delegatecall to a input-controlled function id
        - (success,returnData) = callee.delegatecall(data) (contracts/CErc20Delegator.sol#501)
CErc20Delegator.fallback() (contracts/CErc20Delegator.sol#543-561) uses delegatecall to a input-controlled function id
        - (success) = implementation.delegatecall(msg.data) (contracts/CErc20Delegator.sol#547)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-delegatecall
INFO:Detectors:
Token contract has a CErc20Delegator.fallback() (contracts/CErc20Delegator.sol#543-561) which might be dangerous to implement
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/token_fallback.md
INFO:Detectors:
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transfer(address,uint256) (contracts/EIP20NonStandardInterface.sol#35)
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transferFrom(address,address,uint256) (contracts/EIP20NonStandardInterface.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
INFO:Detectors:
Reentrancy in CErc20Delegator._setImplementation(address,bool,bytes) (contracts/CErc20Delegator.sol#68-85):
        External calls:
        - delegateToImplementation(abi.encodeWithSignature("_resignImplementation()")) (contracts/CErc20Delegator.sol#76)
                - (success,returnData) = callee.delegatecall(data) (contracts/CErc20Delegator.sol#501)
        State variables written after the call(s):
        - implementation = implementation_ (contracts/CErc20Delegator.sol#80)
        CDelegationStorage.implementation (contracts/CTokenInterfaces.sol#318) can be used in cross function reentrancies:
        - CErc20Delegator._setImplementation(address,bool,bytes) (contracts/CErc20Delegator.sol#68-85)
        - CErc20Delegator.delegateToImplementation(bytes) (contracts/CErc20Delegator.sol#516-518)
        - CErc20Delegator.fallback() (contracts/CErc20Delegator.sol#543-561)
        - CDelegationStorage.implementation (contracts/CTokenInterfaces.sol#318)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
Function CErc20Delegator.delegateTo(address,bytes) (contracts/CErc20Delegator.sol#500-508) contains a low level call to a custom address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/call_forward_to_protected.md
INFO:Detectors:
CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) shadows:
        - CTokenInterface.NewReserveGuardian(address,address) (contracts/CTokenInterfaces.sol#195) (event)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
CErc20Delegator.constructor(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8,address,address,bytes).admin_ (contracts/CErc20Delegator.sol#33) lacks a zero-check on :
                - admin = admin_ (contracts/CErc20Delegator.sol#59)
CErc20Delegator._setImplementation(address,bool,bytes).implementation_ (contracts/CErc20Delegator.sol#69) lacks a zero-check on :
                - implementation = implementation_ (contracts/CErc20Delegator.sol#80)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in CErc20Delegator._setImplementation(address,bool,bytes) (contracts/CErc20Delegator.sol#68-85):
        External calls:
        - delegateToImplementation(abi.encodeWithSignature("_resignImplementation()")) (contracts/CErc20Delegator.sol#76)
                - (success,returnData) = callee.delegatecall(data) (contracts/CErc20Delegator.sol#501)
        - delegateToImplementation(abi.encodeWithSignature("_becomeImplementation(bytes)",becomeImplementationData)) (contracts/CErc20Delegator.sol#82)
                - (success,returnData) = callee.delegatecall(data) (contracts/CErc20Delegator.sol#501)
        Event emitted after the call(s):
        - NewImplementation(oldImplementation,implementation) (contracts/CErc20Delegator.sol#84)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Function ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) has a dubious typecast: uint256<=uint256
Function ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) has a dubious typecast: uint256<=uint256
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
```

```
INFO:Detectors:
CErc20Delegator.delegateTo(address,bytes) (contracts/CErc20Delegator.sol#500-508) uses assembly
        - INLINE ASM (contracts/CErc20Delegator.sol#502-506)
CErc20Delegator.delegateToViewImplementation(bytes) (contracts/CErc20Delegator.sol#527-537) uses assembly
        - INLINE ASM (contracts/CErc20Delegator.sol#531-535)
CErc20Delegator.fallback() (contracts/CErc20Delegator.sol#543-561) uses assembly
        - INLINE ASM (contracts/CErc20Delegator.sol#549-560)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) is never used and should be removed
ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.10 (contracts/CErc20Delegator.sol#2) allows old versions
Pragma version^0.8.10 (contracts/CTokenInterfaces.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ComptrollerInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20NonStandardInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ErrorReporter.sol#2) allows old versions
Pragma version^0.8.10 (contracts/InterestRateModel.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in CErc20Delegator.delegateTo(address,bytes) (contracts/CErc20Delegator.sol#500-508):
        - (success,returnData) = callee.delegatecall(data) (contracts/CErc20Delegator.sol#501)
Low level call in CErc20Delegator.delegateToViewImplementation(bytes) (contracts/CErc20Delegator.sol#527-537):
        - (success,returnData) = address(this).staticcall(abi.encodeWithSignature(delegateToImplementation(bytes),data)) (contracts/CErc20Delegator.sol#528-530)
Low level call in CErc20Delegator.fallback() (contracts/CErc20Delegator.sol#543-561):
        - (success) = implementation.delegatecall(msg.data) (contracts/CErc20Delegator.sol#547)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function CErc20Delegator._setImplementation(address,bool,bytes) (contracts/CErc20Delegator.sol#68-85) is not in mixedCase
Function CErc20Delegator._setPendingAdmin(address) (contracts/CErc20Delegator.sol#407-412) is not in mixedCase
Function CErc20Delegator._setComptroller(ComptrollerInterface) (contracts/CErc20Delegator.sol#419-424) is not in mixedCase
Function CErc20Delegator._setReserveGuardian(address) (contracts/CErc20Delegator.sol#431-436) is not in mixedCase
Function CErc20Delegator._setReserveFactor(uint256) (contracts/CErc20Delegator.sol#443-448) is not in mixedCase
Function CErc20Delegator._acceptAdmin() (contracts/CErc20Delegator.sol#455-458) is not in mixedCase
Function CErc20Delegator._addReserves(uint256) (contracts/CErc20Delegator.sol#465-468) is not in mixedCase
Function CErc20Delegator._reduceReserves(uint256) (contracts/CErc20Delegator.sol#475-478) is not in mixedCase
Function CErc20Delegator._setInterestRateModel(InterestRateModel) (contracts/CErc20Delegator.sol#486-491) is not in mixedCase
Variable CTokenStorage._notEntered (contracts/CTokenInterfaces.sol#13) is not in mixedCase
Constant CTokenStorage.borrowRateMaxMantissa (contracts/CTokenInterfaces.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant CTokenStorage.reserveFactorMaxMantissa (contracts/CTokenInterfaces.sol#34) is not in UPPER_CASE_WITH_UNDERSCORES
Function CTokenInterface._setPendingAdmin(address) (contracts/CTokenInterfaces.sol#260) is not in mixedCase
Function CTokenInterface._acceptAdmin() (contracts/CTokenInterfaces.sol#262) is not in mixedCase
Function CTokenInterface._setComptroller(ComptrollerInterface) (contracts/CTokenInterfaces.sol#264) is not in mixedCase
Function CTokenInterface._setReserveGuardian(address) (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Parameter CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Function CTokenInterface._setReserveFactor(uint256) (contracts/CTokenInterfaces.sol#268) is not in mixedCase
Function CTokenInterface._reduceReserves(uint256) (contracts/CTokenInterfaces.sol#270) is not in mixedCase
Function CTokenInterface._setInterestRateModel(InterestRateModel) (contracts/CTokenInterfaces.sol#272) is not in mixedCase
Function CErc20Interface._addReserves(uint256) (contracts/CTokenInterfaces.sol#311) is not in mixedCase
Function CDelegatorInterface._setImplementation(address,bool,bytes) (contracts/CTokenInterfaces.sol#333-337) is not in mixedCase
Function CDelegateInterface._becomeImplementation(bytes) (contracts/CTokenInterfaces.sol#346) is not in mixedCase
Function CDelegateInterface._resignImplementation() (contracts/CTokenInterfaces.sol#351) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
CTokenStorage._notEntered (contracts/CTokenInterfaces.sol#13) is never used in CErc20Delegator (contracts/CErc20Delegator.sol#11-562)
CTokenStorage.borrowRateMaxMantissa (contracts/CTokenInterfaces.sol#31) is never used in CErc20Delegator (contracts/CErc20Delegator.sol#11-562)
CTokenStorage.reserveFactorMaxMantissa (contracts/CTokenInterfaces.sol#34) is never used in CErc20Delegator (contracts/CErc20Delegator.sol#11-562)
CTokenStorage.initialExchangeRateMantissa (contracts/CTokenInterfaces.sol#37) is never used in CErc20Delegator (contracts/CErc20Delegator.sol#11-562)
CTokenStorage.accountTokens (contracts/CTokenInterfaces.sol#90) is never used in CErc20Delegator (contracts/CErc20Delegator.sol#11-562)
CTokenStorage.transferAllowances (contracts/CTokenInterfaces.sol#93) is never used in CErc20Delegator (contracts/CErc20Delegator.sol#11-562)
CTokenStorage.accountBorrows (contracts/CTokenInterfaces.sol#106) is never used in CErc20Delegator (contracts/CErc20Delegator.sol#11-562)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
CDelegationStorage.implementation (contracts/CTokenInterfaces.sol#318) should be constant
CErc20Storage.underlying (contracts/CTokenInterfaces.sol#279) should be constant
CTokenStorage._notEntered (contracts/CTokenInterfaces.sol#13) should be constant
CTokenStorage.accrualBlockNumber (contracts/CTokenInterfaces.sol#67) should be constant
CTokenStorage.borrowIndex (contracts/CTokenInterfaces.sol#72) should be constant
CTokenStorage.comptroller (contracts/CTokenInterfaces.sol#49) should be constant
CTokenStorage.decimals (contracts/CTokenInterfaces.sol#28) should be constant
CTokenStorage.initialExchangeRateMantissa (contracts/CTokenInterfaces.sol#37) should be constant
CTokenStorage.interestRateModel (contracts/CTokenInterfaces.sol#54) should be constant
CTokenStorage.name (contracts/CTokenInterfaces.sol#18) should be constant
CTokenStorage.pendingAdmin (contracts/CTokenInterfaces.sol#44) should be constant
CTokenStorage.reserveFactorMantissa (contracts/CTokenInterfaces.sol#62) should be constant
CTokenStorage.reserveGuardian (contracts/CTokenInterfaces.sol#116) should be constant
CTokenStorage.symbol (contracts/CTokenInterfaces.sol#23) should be constant
CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77) should be constant
CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82) should be constant
CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
CTokenStorage.admin (contracts/CTokenInterfaces.sol#39) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

CErc20Immutable.sol

```
INFO:Detectors:
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transfer(address,uint256) (contracts/EIP20NonStandardInterface.sol#35)
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transferFrom(address,address,uint256) (contracts/EIP20NonStandardInterface.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
INFO:Detectors:
CToken.accrueInterest() (contracts/CToken.sol#325-381) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/CToken.sol#331)
CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/CToken.sol#293)
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CToken.sol#27-60) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/CToken.sol#36)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == NO_ERROR,LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/CToken.sol#811)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) uses a dangerous strict equality:
        - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745):
        External calls:
        - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#737)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
                - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
        CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - totalSupply = totalSupply - redeemTokens (contracts/CToken.sol#548)
        CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87) can be used in cross function reentrancies:
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
CToken._addReservesFresh(uint256).actualAddAmount (contracts/CToken.sol#1055) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
CErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CErc20.sol#26-41) ignores return value by EIP20Interface(underlying).totalSupply() (contracts/CErc20.sol#40)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) shadows:
        - CTokenInterface.NewReserveGuardian(address,address) (contracts/CTokenInterfaces.sol#195) (event)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
```

```
INFO:Detectors:
CErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8).underlying_ (contracts/CErc20.sol#27) lacks a zero-check on :
        - underlying = underlying_ (contracts/CErc20.sol#39)
CToken._setPendingAdmin(address).newPendingAdmin (contracts/CToken.sol#902) lacks a zero-check on :
        - pendingAdmin = newPendingAdmin (contracts/CToken.sol#912)
CErc20Immutable.constructor(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8,address).admin_ (contracts/CErc20Immutable.sol#31) lacks a zero-check on :
        - admin = admin_ (contracts/CErc20Immutable.sol#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#629)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#630)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#631)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        State variables written after the call(s):
        - accountTokens[minter] = accountTokens[minter] + mintTokens (contracts/CToken.sol#442)
        - totalSupply = totalSupply + mintTokens (contracts/CToken.sol#441)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - accountTokens[redeemer] = accountTokens[redeemer] - redeemTokens (contracts/CToken.sol#549)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#713)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#714)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
Reentrancy in CToken.seizeInternal(address,address,address,uint256) (contracts/CToken.sol#855-892):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        State variables written after the call(s):
        - accountTokens[borrower] = accountTokens[borrower] - seizeTokens (contracts/CToken.sol#885)
        - accountTokens[liquidator] = accountTokens[liquidator] + liquidatorSeizeTokens (contracts/CToken.sol#886)
        - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        - totalSupply = totalSupply - protocolSeizeTokens (contracts/CToken.sol#884)
Reentrancy in CToken.transferTokens(address,address,address,uint256) (contracts/CToken.sol#71-115):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/CToken.sol#73)
        State variables written after the call(s):
        - accountTokens[src] = srcTokensNew (contracts/CToken.sol#100)
        - accountTokens[dst] = dstTokensNew (contracts/CToken.sol#101)
        - transferAllowances[src][spender] = allowanceNew (contracts/CToken.sol#105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        Event emitted after the call(s):
        - Borrow(borrower,borrowAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#642)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
                - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        Event emitted after the call(s):
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(cTokenCollateral),seizeTokens) (contracts/CToken.sol#824)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        Event emitted after the call(s):
        - Mint(minter,actualMintAmount,mintTokens) (contracts/CToken.sol#445)
        - Transfer(address(this),minter,mintTokens) (contracts/CToken.sol#446)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        Event emitted after the call(s):
        - Redeem(redeemer,redeemAmount,redeemTokens) (contracts/CToken.sol#561)
        - Transfer(redeemer,address(this),redeemTokens) (contracts/CToken.sol#560)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Function CErc20.doTransferIn(address,uint256) (contracts/CErc20.sol#190-219) has a dubious typecast: address<=EIP20Interface
Function CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) has a dubious typecast: address<=address
Function CToken._acceptAdmin() (contracts/CToken.sol#925-945) has a dubious typecast: address<=address
Function ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) has a dubious typecast: uint256<=uint256
Function ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) has a dubious typecast: uint256<=uint256
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
CErc20.doTransferIn(address,uint256) (contracts/CErc20.sol#190-219) uses assembly
        - INLINE ASM (contracts/CErc20.sol#198-213)
CErc20.doTransferOut(address,uint256) (contracts/CErc20.sol#230-252) uses assembly
        - INLINE ASM (contracts/CErc20.sol#235-250)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) is never used and should be removed
ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#92-94) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#88-90) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#152-154) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#156-158) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#140-142) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#144-146) is never used and should be removed
ExponentialNoError.div_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#160-162) is never used and should be removed
ExponentialNoError.fraction(uint256,uint256) (contracts/ExponentialNoError.sol#168-170) is never used and should be removed
ExponentialNoError.greaterThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#67-69) is never used and should be removed
ExponentialNoError.isZeroExp(ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#74-76) is never used and should be removed
ExponentialNoError.lessThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#53-55) is never used and should be removed
ExponentialNoError.lessThanOrEqualExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#60-62) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#124-126) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#128-130) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#112-114) is never used and should be removed
ExponentialNoError.mul_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#132-134) is never used and should be removed
ExponentialNoError.safe224(uint256,string) (contracts/ExponentialNoError.sol#78-81) is never used and should be removed
ExponentialNoError.safe32(uint256,string) (contracts/ExponentialNoError.sol#83-86) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#104-106) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#100-102) is never used and should be removed
ExponentialNoError.sub_(uint256,uint256) (contracts/ExponentialNoError.sol#108-110) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.10 (contracts/CErc20.sol#2) allows old versions
Pragma version^0.8.10 (contracts/CErc20Immutable.sol#2) allows old versions
Pragma version^0.8.10 (contracts/CToken.sol#2) allows old versions
Pragma version^0.8.10 (contracts/CTokenInterfaces.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ComptrollerInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20Interface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20NonStandardInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ErrorReporter.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ExponentialNoError.sol#2) allows old versions
Pragma version^0.8.10 (contracts/InterestRateModel.sol#2) allows old versions
Pragma version0.8.10 (contracts/Whitelist.sol#3) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
INFO:Detectors:
Function CErc20._addReserves(uint256) (contracts/CErc20.sol#165-167) is not in mixedCase
Function CErc20._delegateCompLikeTo(address) (contracts/CErc20.sol#259-262) is not in mixedCase
Function CToken._setPendingAdmin(address) (contracts/CToken.sol#902-918) is not in mixedCase
Function CToken._acceptAdmin() (contracts/CToken.sol#925-945) is not in mixedCase
Function CToken._setComptroller(ComptrollerInterface) (contracts/CToken.sol#952-969) is not in mixedCase
Function CToken._setReserveGuardian(address) (contracts/CToken.sol#976-979) is not in mixedCase
Function CToken._setReserveFactor(uint256) (contracts/CToken.sol#998-1002) is not in mixedCase
Function CToken._reduceReserves(uint256) (contracts/CToken.sol#1093-1097) is not in mixedCase
Function CToken._setInterestRateModel(InterestRateModel) (contracts/CToken.sol#1154-1158) is not in mixedCase
Variable CTokenStorage._notEntered (contracts/CTokenInterfaces.sol#13) is not in mixedCase
Constant CTokenStorage.borrowRateMaxMantissa (contracts/CTokenInterfaces.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant CTokenStorage.reserveFactorMaxMantissa (contracts/CTokenInterfaces.sol#34) is not in UPPER_CASE_WITH_UNDERSCORES
Function CTokenInterface._setPendingAdmin(address) (contracts/CTokenInterfaces.sol#260) is not in mixedCase
Function CTokenInterface._acceptAdmin() (contracts/CTokenInterfaces.sol#262) is not in mixedCase
Function CTokenInterface._setComptroller(ComptrollerInterface) (contracts/CTokenInterfaces.sol#264) is not in mixedCase
Function CTokenInterface._setReserveGuardian(address) (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Parameter CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Function CTokenInterface._setReserveFactor(uint256) (contracts/CTokenInterfaces.sol#268) is not in mixedCase
Function CTokenInterface._reduceReserves(uint256) (contracts/CTokenInterfaces.sol#270) is not in mixedCase
Function CTokenInterface._setInterestRateModel(InterestRateModel) (contracts/CTokenInterfaces.sol#272) is not in mixedCase
Function CErc20Interface._addReserves(uint256) (contracts/CTokenInterfaces.sol#311) is not in mixedCase
Function CDelegatorInterface._setImplementation(address,bool,bytes) (contracts/CTokenInterfaces.sol#333-337) is not in mixedCase
Function CDelegateInterface._becomeImplementation(bytes) (contracts/CTokenInterfaces.sol#346) is not in mixedCase
Function CDelegateInterface._resignImplementation() (contracts/CTokenInterfaces.sol#351) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncate(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#37-40) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncateAddUInt(ExponentialNoError.Exp,uint256,uint256) (contracts/ExponentialNoError.sol#45-48) is not in mixedCase
Constant ExponentialNoError.expScale (contracts/ExponentialNoError.sol#12) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.doubleScale (contracts/ExponentialNoError.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.mantissaOne (contracts/ExponentialNoError.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter Whitelist.updateWhitelist(address,bool)._address (contracts/Whitelist.sol#11) is not in mixedCase
Parameter Whitelist.updateWhitelist(address,bool)._isActive (contracts/Whitelist.sol#12) is not in mixedCase
Parameter Whitelist.getWhitelisted(address)._address (contracts/Whitelist.sol#17) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable CTokenInterface.seize(address,address,uint256).seizeTokens (contracts/CTokenInterfaces.sol#256) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.seize(address,address,uint256).seizeTokens (contracts/CToken.sol#839) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface).seizeTokens (contracts/CToken.sol#806) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.seizeInternal(address,address,address,uint256).seizeTokens (contracts/CToken.sol#855) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is never used in CErc20Immutable (contracts/CErc20Immutable.sol#11-50)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
Function CErc20.doTransferIn(address,uint256) (contracts/CErc20.sol#190-219) contains magic number: 32
Function CErc20.doTransferOut(address,uint256) (contracts/CErc20.sol#230-252) contains magic number: 32
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
CDelegationStorage.implementation (contracts/CTokenInterfaces.sol#318) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

CEther.sol

```
INFO:Detectors:
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transfer(address,uint256) (contracts/EIP20NonStandardInterface.sol#35)
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transferFrom(address,address,uint256) (contracts/EIP20NonStandardInterface.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
INFO:Detectors:
CToken.accrueInterest() (contracts/CToken.sol#325-381) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/CToken.sol#331)
CEther.doTransferIn(address,uint256) (contracts/CEther.sol#166-171) uses a dangerous strict equality:
        - require(bool,string)(msg.value == amount,value mismatch) (contracts/CEther.sol#169)
CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/CToken.sol#293)
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CToken.sol#27-60) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/CToken.sol#36)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == NO_ERROR,LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/CToken.sol#811)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) uses a dangerous strict equality:
        - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745):
        External calls:
        - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#737)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
                - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
        CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - totalSupply = totalSupply - redeemTokens (contracts/CToken.sol#548)
        CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87) can be used in cross function reentrancies:
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
CToken._addReservesFresh(uint256).actualAddAmount (contracts/CToken.sol#1055) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) shadows:
        - CTokenInterfaceNewReserveGuardian(address,address) (contracts/CTokenInterfaces.sol#195) (event)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
CToken._setPendingAdmin(address).newPendingAdmin (contracts/CToken.sol#902) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (contracts/CToken.sol#912)
CEther.constructor(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8,address).admin_ (contracts/CEther.sol#29) lacks a zero-check on :
                - admin = admin_ (contracts/CEther.sol#37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#629)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#630)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#631)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        State variables written after the call(s):
        - accountTokens[minter] = accountTokens[minter] + mintTokens (contracts/CToken.sol#442)
        - totalSupply = totalSupply + mintTokens (contracts/CToken.sol#441)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - accountTokens[redeemer] = accountTokens[redeemer] - redeemTokens (contracts/CToken.sol#549)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#713)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#714)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
Reentrancy in CToken.seizeInternal(address,address,address,uint256) (contracts/CToken.sol#855-892):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        State variables written after the call(s):
        - accountTokens[borrower] = accountTokens[borrower] - seizeTokens (contracts/CToken.sol#885)
        - accountTokens[liquidator] = accountTokens[liquidator] + liquidatorSeizeTokens (contracts/CToken.sol#886)
        - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        - totalSupply = totalSupply - protocolSeizeTokens (contracts/CToken.sol#884)
Reentrancy in CToken.transferTokens(address,address,address,uint256) (contracts/CToken.sol#71-115):
```

Comptroller.sol

```
INFO:Detectors:
Unitroller.fallback() (contracts/Unitroller.sol#134-150) uses delegatecall to a input-controlled function id
        - (success) = comptrollerImplementation.delegatecall(msg.data) (contracts/Unitroller.sol#136)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-delegatecall
INFO:Detectors:
Comptroller.grantCompInternal(address,uint256) (contracts/Comptroller.sol#1499-1507) ignores return value by comp.transfer(user,amount) (contracts/Comptroller.sol#1503)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
UnitrollerAdminStorage.comptrollerImplementation (contracts/ComptrollerStorage.sol#21) is never initialized. It is used in:
        - Comptroller.adminOrInitializing() (contracts/Comptroller.sol#1261-1263)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
Function CToken.nonReentrant() (contracts/CToken.sol#1222-1227) is an unprotected initializer.
Function Comptroller._initializeMarket(address) (contracts/Comptroller.sol#1071-1094) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initialize.md
INFO:Detectors:
Comptroller.mintVerify(address,address,uint256,uint256) (contracts/Comptroller.sol#295-306) uses a Boolean constant improperly:
        -false (contracts/Comptroller.sol#303)
Comptroller.borrowVerify(address,address,uint256) (contracts/Comptroller.sol#441-451) uses a Boolean constant improperly:
        -false (contracts/Comptroller.sol#448)
Comptroller.repayBorrowVerify(address,address,address,uint256,uint256) (contracts/Comptroller.sol#491-509) uses a Boolean constant improperly:
        -false (contracts/Comptroller.sol#506)
Comptroller.liquidateBorrowVerify(address,address,address,address,uint256,uint256) (contracts/Comptroller.sol#588-608) uses a Boolean constant improperly:
        -false (contracts/Comptroller.sol#605)
Comptroller.seizeVerify(address,address,address,address,uint256) (contracts/Comptroller.sol#655-673) uses a Boolean constant improperly:
        -false (contracts/Comptroller.sol#670)
Comptroller.transferVerify(address,address,address,uint256) (contracts/Comptroller.sol#714-725) uses a Boolean constant improperly:
        -false (contracts/Comptroller.sol#722)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#misuse-of-a-boolean-constant
INFO:Detectors:
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transfer(address,uint256) (contracts/EIP20NonStandardInterface.sol#35)
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transferFrom(address,address,uint256) (contracts/EIP20NonStandardInterface.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
INFO:Detectors:
CToken.accrueInterest() (contracts/CToken.sol#325-381) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/CToken.sol#331)
CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/CToken.sol#293)
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CToken.sol#27-60) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/CToken.sol#36)
Comptroller._grantComp(address,uint256) (contracts/Comptroller.sol#1517-1522) uses a dangerous strict equality:
        - require(bool,string)(amountLeft == 0,insufficient comp for grant) (contracts/Comptroller.sol#1520)
Comptroller.getHypotheticalAccountLiquidityInternal(address,CToken,uint256,uint256) (contracts/Comptroller.sol#811-882) uses a dangerous strict equality:
        - vars.oraclePriceMantissa == 0 (contracts/Comptroller.sol#838)
Comptroller.liquidateBorrowAllowed(address,address,address,address,uint256) (contracts/Comptroller.sol#541-578) uses a dangerous strict equality:
        - shortfall == 0 (contracts/Comptroller.sol#567)
Comp._writeCheckpoint(address,uint32,uint96,uint96) (contracts/Governance/Comp.sol#270-281) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (contracts/Governance/Comp.sol#273)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745):
        External calls:
        - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#737)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
                - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
        CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - totalSupply = totalSupply - redeemTokens (contracts/CToken.sol#548)
        CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87) can be used in cross function reentrancies:
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
CToken._addReservesFresh(uint256).actualAddAmount (contracts/CToken.sol#1055) is a local variable never initialized
Comptroller.borrowAllowed(address,address,uint256).err_scope_0 (contracts/Comptroller.sol#414) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
Comptroller._supportMarket(CToken) (contracts/Comptroller.sol#1039-1062) ignores return value by cToken.isCToken() (contracts/Comptroller.sol#1048)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) shadows:
        - CTokenInterface.NewReserveGuardian(address,address) (contracts/CTokenInterfaces.sol#195) (event)
Comptroller.updateCompSupplyIndex(address).compAccrued (contracts/Comptroller.sol#1313) shadows:
        - ComptrollerV3Storage.compAccrued (contracts/ComptrollerStorage.sol#120) (state variable)
Comptroller.updateCompBorrowIndex(address,ExponentialNoError.Exp).compAccrued (contracts/Comptroller.sol#1337) shadows:
        - ComptrollerV3Storage.compAccrued (contracts/ComptrollerStorage.sol#120) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CToken.sol#27-60) should emit an event for:
        - initialExchangeRateMantissa = initialExchangeRateMantissa_ (contracts/CToken.sol#39)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
CToken._setPendingAdmin(address).newPendingAdmin (contracts/CToken.sol#902) lacks a zero-check on :
        - pendingAdmin = newPendingAdmin (contracts/CToken.sol#912)
Comptroller._setBorrowCapGuardian(address).newBorrowCapGuardian (contracts/Comptroller.sol#1123) lacks a zero-check on :
        - borrowCapGuardian = newBorrowCapGuardian (contracts/Comptroller.sol#1130)
Comptroller._setSupplyCapGuardian(address).newSupplyCapGuardian (contracts/Comptroller.sol#1163) lacks a zero-check on :
        - supplyCapGuardian = newSupplyCapGuardian (contracts/Comptroller.sol#1170)
Comptroller._setSpeedGuardian(address).newSpeedGuardian (contracts/Comptroller.sol#1180) lacks a zero-check on :
        - speedGuardian = newSpeedGuardian (contracts/Comptroller.sol#1187)
Comptroller._setPauseGuardian(address).newPauseGuardian (contracts/Comptroller.sol#1198) lacks a zero-check on :
        - pauseGuardian = newPauseGuardian (contracts/Comptroller.sol#1207)
Unitroller._setPendingImplementation(address).newPendingImplementation (contracts/Unitroller.sol#39) lacks a zero-check on :
        - pendingComptrollerImplementation = newPendingImplementation (contracts/Unitroller.sol#46)
Unitroller._setPendingAdmin(address).newPendingAdmin (contracts/Unitroller.sol#84) lacks a zero-check on :
        - pendingAdmin = newPendingAdmin (contracts/Unitroller.sol#94)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Comptroller.claimComp(address[],CToken[],bool,bool) (contracts/Comptroller.sol#1469-1490) has external calls inside a loop: borrowIndex = Exp(cToken.borrowIndex()) (contracts/Comptroller.sol#1474)
Comptroller.updateCompBorrowIndex(address,ExponentialNoError.Exp) (contracts/Comptroller.sol#1330-1347) has external calls inside a loop: borrowAmount = div_(CToken(cToken).totalBorrows(),marketBorrowIndex) (contracts/Comptroller.sol#133
6)
Comptroller.distributeBorrowerComp(address,address,ExponentialNoError.Exp) (contracts/Comptroller.sol#1393-1424) has external calls inside a loop: borrowerAmount = div_(CToken(cToken).borrowBalanceStored(borrower),marketBorrowIndex) (con
tracts/Comptroller.sol#1415)
Comptroller.updateCompSupplyIndex(address) (contracts/Comptroller.sol#1306-1323) has external calls inside a loop: supplyTokens = CToken(cToken).totalSupply() (contracts/Comptroller.sol#1312)
Comptroller.distributeSupplierComp(address,address) (contracts/Comptroller.sol#1354-1385) has external calls inside a loop: supplierTokens = CToken(cToken).balanceOf(supplier) (contracts/Comptroller.sol#1376)
Comptroller.grantCompInternal(address,uint256) (contracts/Comptroller.sol#1499-1507) has external calls inside a loop: compRemaining = comp.balanceOf(address(this)) (contracts/Comptroller.sol#1501)
Comptroller.grantCompInternal(address,uint256) (contracts/Comptroller.sol#1499-1507) has external calls inside a loop: comp.transfer(user,amount) (contracts/Comptroller.sol#1503)
Comptroller.setCompSpeedInternal(CToken,uint256,uint256) (contracts/Comptroller.sol#1273-1299) has external calls inside a loop: borrowIndex = Exp(cToken.borrowIndex()) (contracts/Comptroller.sol#1292)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#629)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#630)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#631)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        State variables written after the call(s):
        - accountTokens[minter] = accountTokens[minter] + mintTokens (contracts/CToken.sol#442)
        - totalSupply = totalSupply + mintTokens (contracts/CToken.sol#441)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - accountTokens[redeemer] = accountTokens[redeemer] - redeemTokens (contracts/CToken.sol#549)
```

```
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#713)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#714)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
Reentrancy in CToken.seizeInternal(address,address,address,uint256) (contracts/CToken.sol#855-892):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        State variables written after the call(s):
        - accountTokens[borrower] = accountTokens[borrower] - seizeTokens (contracts/CToken.sol#885)
        - accountTokens[liquidator] = accountTokens[liquidator] + liquidatorSeizeTokens (contracts/CToken.sol#886)
        - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        - totalSupply = totalSupply - protocolSeizeTokens (contracts/CToken.sol#884)
Reentrancy in CToken.transferTokens(address,address,address,uint256) (contracts/CToken.sol#71-115):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/CToken.sol#73)
        State variables written after the call(s):
        - accountTokens[src] = srcTokensNew (contracts/CToken.sol#100)
        - accountTokens[dst] = dstTokensNew (contracts/CToken.sol#101)
        - transferAllowances[src][spender] = allowanceNew (contracts/CToken.sol#105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in Comptroller._grantComp(address,uint256) (contracts/Comptroller.sol#1517-1522):
        External calls:
        - amountLeft = grantCompInternal(recipient,amount) (contracts/Comptroller.sol#1519)
                - comp.transfer(user,amount) (contracts/Comptroller.sol#1503)
        Event emitted after the call(s):
        - CompGranted(recipient,amount) (contracts/Comptroller.sol#1521)
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        Event emitted after the call(s):
        - Borrow(borrower,borrowAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#642)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
                - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        Event emitted after the call(s):
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(cTokenCollateral),seizeTokens) (contracts/CToken.sol#824)
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745):
        External calls:
        - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#737)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
                - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(cTokenCollateral),seizeTokens) (contracts/CToken.sol#824)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        Event emitted after the call(s):
        - Mint(minter,actualMintAmount,mintTokens) (contracts/CToken.sol#445)
        - Transfer(address(this),minter,mintTokens) (contracts/CToken.sol#446)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        Event emitted after the call(s):
        - Redeem(redeemer,redeemAmount,redeemTokens) (contracts/CToken.sol#561)
        - Transfer(redeemer,address(this),redeemTokens) (contracts/CToken.sol#560)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
Reentrancy in CToken.seizeInternal(address,address,address,uint256) (contracts/CToken.sol#855-892):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        Event emitted after the call(s):
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
Reentrancy in CToken.transferTokens(address,address,address,uint256) (contracts/CToken.sol#71-115):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/CToken.sol#73)
        Event emitted after the call(s):
        - Transfer(src,dst,tokens) (contracts/CToken.sol#109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Comp.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (contracts/Governance/Comp.sol#167-178) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,Comp::delegateBySig: signature expired) (contracts/Governance/Comp.sol#176)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Function CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) has a dubious typecast: address<=address
Function CToken._acceptAdmin() (contracts/CToken.sol#925-945) has a dubious typecast: address<=address
Function ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) has a dubious typecast: uint256<=uint256
Function ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) has a dubious typecast: uint256<=uint256
Function Comptroller.getAccountLiquidity(address) (contracts/Comptroller.sol#753-762) has a dubious typecast: CToken<=address
Function Comptroller.getAccountLiquidityInternal(address) (contracts/Comptroller.sol#770-772) has a dubious typecast: CToken<=address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
Comp.getChainId() (contracts/Governance/Comp.sol#304-310) uses assembly
        - INLINE ASM (contracts/Governance/Comp.sol#306-308)
Unitroller.fallback() (contracts/Unitroller.sol#134-150) uses assembly
        - INLINE ASM (contracts/Unitroller.sol#138-149)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Comptroller.addToMarketInternal(CToken,address) (contracts/Comptroller.sol#163-187) compares to a boolean constant:
        -marketToJoin.accountMembership[borrower] == true (contracts/Comptroller.sol#171)
Comptroller._setMintPaused(CToken,bool) (contracts/Comptroller.sol#1215-1223) compares to a boolean constant:
        -require(bool,string)(msg.sender == admin || state == true,only admin can unpause) (contracts/Comptroller.sol#1218)
Comptroller._setBorrowPaused(CToken,bool) (contracts/Comptroller.sol#1225-1233) compares to a boolean constant:
        -require(bool,string)(msg.sender == admin || state == true,only admin can unpause) (contracts/Comptroller.sol#1228)
Comptroller._setTransferPaused(bool) (contracts/Comptroller.sol#1235-1242) compares to a boolean constant:
        -require(bool,string)(msg.sender == admin || state == true,only admin can unpause) (contracts/Comptroller.sol#1237)
Comptroller._setSeizePaused(bool) (contracts/Comptroller.sol#1244-1251) compares to a boolean constant:
        -require(bool,string)(msg.sender == admin || state == true,only admin can unpause) (contracts/Comptroller.sol#1246)
Comptroller.claimComp(address[],CToken[],bool,bool) (contracts/Comptroller.sol#1469-1490) compares to a boolean constant:
        -borrowers == true (contracts/Comptroller.sol#1473)
Comptroller.claimComp(address[],CToken[],bool,bool) (contracts/Comptroller.sol#1469-1490) compares to a boolean constant:
        -suppliers == true (contracts/Comptroller.sol#1480)
Comptroller.isDeprecated(CToken) (contracts/Comptroller.sol#1579-1584) compares to a boolean constant:
        -markets[address(cToken)].collateralFactorMantissa == 0 && borrowGuardianPaused[address(cToken)] == true && cToken.reserveFactorMantissa() == 1e18 (contracts/Comptroller.sol#1580-1583)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```

```
INFO:Detectors:
CToken._addReservesFresh(uint256) (contracts/CToken.sol#1052-1086) is never used and should be removed
CToken._addReservesInternal(uint256) (contracts/CToken.sol#1038-1044) is never used and should be removed
CToken.borrowBehalfInternal(uint256,address) (contracts/CToken.sol#583-589) is never used and should be removed
CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643) is never used and should be removed
CToken.borrowInternal(uint256) (contracts/CToken.sol#571-575) is never used and should be removed
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) is never used and should be removed
CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745) is never used and should be removed
CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451) is never used and should be removed
CToken.mintInternal(uint256) (contracts/CToken.sol#388-392) is never used and should be removed
CToken.redeemBehalfInternal(uint256,address) (contracts/CToken.sol#471-477) is never used and should be removed
CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565) is never used and should be removed
CToken.redeemInternal(uint256) (contracts/CToken.sol#458-462) is never used and should be removed
CToken.redeemUnderlyingInternal(uint256) (contracts/CToken.sol#484-488) is never used and should be removed
CToken.repayBorrowBehalfInternal(address,uint256) (contracts/CToken.sol#660-664) is never used and should be removed
CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721) is never used and should be removed
CToken.repayBorrowInternal(uint256) (contracts/CToken.sol#649-653) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#88-90) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#152-154) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#156-158) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#144-146) is never used and should be removed
ExponentialNoError.div_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#160-162) is never used and should be removed
ExponentialNoError.greaterThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#67-69) is never used and should be removed
ExponentialNoError.isZeroExp(ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#74-76) is never used and should be removed
ExponentialNoError.lessThanOrEqualExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#60-62) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#124-126) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#128-130) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#104-106) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#100-102) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.10 (contracts/CToken.sol#2) allows old versions
Pragma version^0.8.10 (contracts/CTokenInterfaces.sol#2) allows old versions
Pragma version^0.8.10 (contracts/Comptroller.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ComptrollerInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ComptrollerStorage.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20Interface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20NonStandardInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ErrorReporter.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ExponentialNoError.sol#2) allows old versions
Pragma version^0.8.10 (contracts/Governance/Comp.sol#2) allows old versions
Pragma version^0.8.10 (contracts/InterestRateModel.sol#2) allows old versions
Pragma version^0.8.10 (contracts/PriceOracle.sol#2) allows old versions
Pragma version^0.8.10 (contracts/Unitroller.sol#2) allows old versions
Pragma version0.8.10 (contracts/Whitelist.sol#3) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Unitroller.fallback() (contracts/Unitroller.sol#134-150):
        - (success) = comptrollerImplementation.delegatecall(msg.data) (contracts/Unitroller.sol#136)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function CToken._setPendingAdmin(address) (contracts/CToken.sol#902-918) is not in mixedCase
Function CToken._acceptAdmin() (contracts/CToken.sol#925-945) is not in mixedCase
Function CToken._setComptroller(ComptrollerInterface) (contracts/CToken.sol#952-969) is not in mixedCase
Function CToken._setReserveGuardian(address) (contracts/CToken.sol#976-979) is not in mixedCase
Function CToken._setReserveFactor(uint256) (contracts/CToken.sol#998-1002) is not in mixedCase
Function CToken._reduceReserves(uint256) (contracts/CToken.sol#1093-1097) is not in mixedCase
Function CToken._setInterestRateModel(InterestRateModel) (contracts/CToken.sol#1154-1158) is not in mixedCase
Variable CTokenStorage._notEntered (contracts/CTokenInterfaces.sol#11) is not in mixedCase
Constant CTokenStorage.borrowRateMaxMantissa (contracts/CTokenInterfaces.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant CTokenStorage.reserveFactorMaxMantissa (contracts/CTokenInterfaces.sol#34) is not in UPPER_CASE_WITH_UNDERSCORES
Function CTokenInterface._setPendingAdmin(address) (contracts/CTokenInterfaces.sol#260) is not in mixedCase
Function CTokenInterface._acceptAdmin() (contracts/CTokenInterfaces.sol#262) is not in mixedCase
Function CTokenInterface._setComptroller(ComptrollerInterface) (contracts/CTokenInterfaces.sol#264) is not in mixedCase
Function CTokenInterface._setReserveGuardian(address) (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Parameter CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Function CTokenInterface._setReserveFactor(uint256) (contracts/CTokenInterfaces.sol#268) is not in mixedCase
Function CTokenInterface._reduceReserves(uint256) (contracts/CTokenInterfaces.sol#270) is not in mixedCase
Function CTokenInterface._setInterestRateModel(InterestRateModel) (contracts/CTokenInterfaces.sol#272) is not in mixedCase
Function CErc20Interface._addReserves(uint256) (contracts/CTokenInterfaces.sol#311) is not in mixedCase
Function CDelegatorInterface._setImplementation(address,bool,bytes) (contracts/CTokenInterfaces.sol#333-337) is not in mixedCase
Function CDelegateInterface._becomeImplementation(bytes) (contracts/CTokenInterfaces.sol#340) is not in mixedCase
Function CDelegateInterface._resignImplementation() (contracts/CTokenInterfaces.sol#351) is not in mixedCase
Function Comptroller._setPriceOracle(PriceOracle) (contracts/Comptroller.sol#932-948) is not in mixedCase
Function Comptroller._setCloseFactor(uint256) (contracts/Comptroller.sol#956-965) is not in mixedCase
Function Comptroller._setCollateralFactor(CToken,uint256) (contracts/Comptroller.sol#974-1007) is not in mixedCase
Function Comptroller._setLiquidationIncentive(uint256) (contracts/Comptroller.sol#1015-1031) is not in mixedCase
Function Comptroller._supportMarket(CToken) (contracts/Comptroller.sol#1039-1062) is not in mixedCase
Function Comptroller._setMarketBorrowCaps(CToken[],uint256[]) (contracts/Comptroller.sol#1102-1117) is not in mixedCase
Function Comptroller._setBorrowCapGuardian(address) (contracts/Comptroller.sol#1123-1134) is not in mixedCase
Function Comptroller._setMarketSupplyCaps(CToken[],uint256[]) (contracts/Comptroller.sol#1142-1157) is not in mixedCase
Function Comptroller._setSupplyCapGuardian(address) (contracts/Comptroller.sol#1163-1174) is not in mixedCase
Function Comptroller._setSpeedGuardian(address) (contracts/Comptroller.sol#1180-1191) is not in mixedCase
Function Comptroller._setPauseGuardian(address) (contracts/Comptroller.sol#1198-1213) is not in mixedCase
Function Comptroller._setMintPaused(CToken,bool) (contracts/Comptroller.sol#1215-1223) is not in mixedCase
Function Comptroller._setBorrowPaused(CToken,bool) (contracts/Comptroller.sol#1225-1233) is not in mixedCase
Function Comptroller._setTransferPaused(bool) (contracts/Comptroller.sol#1235-1242) is not in mixedCase
Function Comptroller._setSeizePaused(bool) (contracts/Comptroller.sol#1244-1251) is not in mixedCase
Function Comptroller._become(Unitroller) (contracts/Comptroller.sol#1253-1256) is not in mixedCase
Function Comptroller._grantComp(address,uint256) (contracts/Comptroller.sol#1517-1522) is not in mixedCase
Function Comptroller._setCompSpeeds(CToken[],uint256[],uint256[]) (contracts/Comptroller.sol#1530-1542) is not in mixedCase
Function Comptroller._setContributorCompSpeed(address,uint256) (contracts/Comptroller.sol#1549-1563) is not in mixedCase
Constant Comptroller.closeFactorMinMantissa (contracts/Comptroller.sol#104) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Comptroller.closeFactorMaxMantissa (contracts/Comptroller.sol#107) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Comptroller.collateralFactorMaxMantissa (contracts/Comptroller.sol#110) is not in UPPER_CASE_WITH_UNDERSCORES
Variable ComptrollerV2Storage._mintGuardianPaused (contracts/ComptrollerStorage.sol#82) is not in mixedCase
Variable ComptrollerV2Storage._borrowGuardianPaused (contracts/ComptrollerStorage.sol#83) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncate(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#37-40) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncateAddUint(ExponentialNoError.Exp,uint256,uint256) (contracts/ExponentialNoError.sol#45-48) is not in mixedCase
Constant ExponentialNoError.expScale (contracts/ExponentialNoError.sol#12) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.doubleScale (contracts/ExponentialNoError.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.mantissaOne (contracts/ExponentialNoError.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Function Unitroller._setPendingImplementation(address) (contracts/Unitroller.sol#39-51) is not in mixedCase
Function Unitroller._acceptImplementation() (contracts/Unitroller.sol#58-76) is not in mixedCase
Function Unitroller._setPendingAdmin(address) (contracts/Unitroller.sol#84-100) is not in mixedCase
Function Unitroller._acceptAdmin() (contracts/Unitroller.sol#107-127) is not in mixedCase
Parameter Whitelist.updateWhitelist(address,bool)._address (contracts/Whitelist.sol#11) is not in mixedCase
Parameter Whitelist.updateWhitelist(address,bool)._isActive (contracts/Whitelist.sol#12) is not in mixedCase
Parameter Whitelist.getWhitelisted(address)._address (contracts/Whitelist.sol#17) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "minter (contracts/Comptroller.sol#262)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "mintAmount (contracts/Comptroller.sol#263)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "cToken (contracts/Comptroller.sol#297)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "minter (contracts/Comptroller.sol#298)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "actualMintAmount (contracts/Comptroller.sol#299)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "mintTokens (contracts/Comptroller.sol#300)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "cToken (contracts/Comptroller.sol#364)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "redeemer (contracts/Comptroller.sol#365)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "cToken (contracts/Comptroller.sol#443)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "borrower (contracts/Comptroller.sol#444)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "borrowAmount (contracts/Comptroller.sol#445)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "payer (contracts/Comptroller.sol#468)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "borrower (contracts/Comptroller.sol#469)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "repayAmount (contracts/Comptroller.sol#470)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "cToken (contracts/Comptroller.sol#499)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "payer (contracts/Comptroller.sol#500)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "borrower (contracts/Comptroller.sol#501)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "actualRepayAmount (contracts/Comptroller.sol#502)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "borrowerIndex (contracts/Comptroller.sol#503)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "liquidator (contracts/Comptroller.sol#549)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "cTokenBorrowed (contracts/Comptroller.sol#597)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "cTokenCollateral (contracts/Comptroller.sol#598)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "liquidator (contracts/Comptroller.sol#599)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "borrower (contracts/Comptroller.sol#600)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "actualRepayAmount (contracts/Comptroller.sol#601)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "seizeTokens (contracts/Comptroller.sol#602)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "seizeTokens (contracts/Comptroller.sol#629)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "cTokenCollateral (contracts/Comptroller.sol#663)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "cTokenBorrowed (contracts/Comptroller.sol#664)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "liquidator (contracts/Comptroller.sol#665)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "borrower (contracts/Comptroller.sol#666)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "seizeTokens (contracts/Comptroller.sol#667)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "cToken (contracts/Comptroller.sol#716)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "src (contracts/Comptroller.sol#717)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "dst (contracts/Comptroller.sol#718)" inComptroller (contracts/Comptroller.sol#17-1598)
Redundant expression "transferTokens (contracts/Comptroller.sol#719)" inComptroller (contracts/Comptroller.sol#17-1598)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
```

```
INFO:Detectors:
Variable CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface).seizeTokens (contracts/CToken.sol#806) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CTokenInterface.seize(address,address,uint256).seizeTokens (contracts/CTokenInterfaces.sol#256) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.seize(address,address,uint256).seizeTokens (contracts/CToken.sol#839) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.seizeInternal(address,address,address,uint256).seizeTokens (contracts/CToken.sol#855) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable Comptroller.closeFactorMaxMantissa (contracts/Comptroller.sol#107) is too similar to Comptroller.closeFactorMinMantissa (contracts/Comptroller.sol#104)
Variable Comptroller._setLiquidationIncentive(uint256).newLiquidationIncentiveMantissa (contracts/Comptroller.sol#1015) is too similar to Comptroller._setLiquidationIncentive(uint256).oldLiquidationIncentiveMantissa (contracts/Comptroller.sol#1022)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
Comp.slitherConstructorConstantVariables() (contracts/Governance/Comp.sol#4-311) uses literals with too many digits:
        - totalSupply = 20000000e18 (contracts/Governance/Comp.sol#15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
CToken (contracts/CToken.sol#17-1228) does not implement functions:
        - CToken.doTransferIn(address,uint256) (contracts/CToken.sol#1208)
        - CToken.doTransferOut(address,uint256) (contracts/CToken.sol#1215)
        - CToken.getCashPrior() (contracts/CToken.sol#1202)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is never used in CToken (contracts/CToken.sol#17-1228)
ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is never used in Comptroller (contracts/Comptroller.sol#17-1598)
ExponentialNoError.mantissaOne (contracts/ExponentialNoError.sol#15) is never used in Comptroller (contracts/Comptroller.sol#17-1598)
Comptroller.closeFactorMinMantissa (contracts/Comptroller.sol#104) is never used in Comptroller (contracts/Comptroller.sol#17-1598)
Comptroller.closeFactorMaxMantissa (contracts/Comptroller.sol#107) is never used in Comptroller (contracts/Comptroller.sol#17-1598)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
Function Comptroller._initializeMarket(address) (contracts/Comptroller.sol#1071-1094) contains magic number: 32
Function Comptroller.updateCompSupplyIndex(address) (contracts/Comptroller.sol#1306-1323) contains magic numbers: 32, 224
Function Comptroller.updateCompBorrowIndex(address,ExponentialNoError.Exp) (contracts/Comptroller.sol#1330-1347) contains magic numbers: 32, 224
Function Comp.approve(address,uint256) (contracts/Governance/Comp.sol#88-100) contains magic number: 96
Function Comp.transfer(address,uint256) (contracts/Governance/Comp.sol#117-121) contains magic number: 96
Function Comp.transferFrom(address,address,uint256) (contracts/Governance/Comp.sol#130-148) contains magic number: 96
Function Comp._writeCheckpoint(address,uint32,uint96,uint96) (contracts/Governance/Comp.sol#270-281) contains magic number: 32
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
CDelegationStorage.implementation (contracts/CTokenInterfaces.sol#318) should be constant
CErc20Storage.underlying (contracts/CTokenInterfaces.sol#279) should be constant
ComptrollerV2Storage._borrowGuardianPaused (contracts/ComptrollerStorage.sol#83) should be constant
ComptrollerV2Storage._mintGuardianPaused (contracts/ComptrollerStorage.sol#82) should be constant
ComptrollerV3Storage.compRate (contracts/ComptrollerStorage.sol#102) should be constant
ComptrollerV9Storage.reserveGuardian (contracts/ComptrollerStorage.sol#163) should be constant
UnitrollerAdminStorage.comptrollerImplementation (contracts/ComptrollerStorage.sol#21) should be constant
UnitrollerAdminStorage.pendingAdmin (contracts/ComptrollerStorage.sol#16) should be constant
UnitrollerAdminStorage.pendingComptrollerImplementation (contracts/ComptrollerStorage.sol#26) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
UnitrollerAdminStorage.admin (contracts/ComptrollerStorage.sol#11) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Detectors:
In a function CToken._acceptAdmin() (contracts/CToken.sol#925-945) variable CTokenStorage.pendingAdmin (contracts/CTokenInterfaces.sol#44) is read multiple times
In a function CToken._reduceReservesFresh(uint256) (contracts/CToken.sol#1105-1146) variable CTokenStorage.admin (contracts/CTokenInterfaces.sol#39) is read multiple times
In a function Comptroller.borrowAllowed(address,address,uint256) (contracts/Comptroller.sol#380-433) variable ComptrollerV2Storage.markets (contracts/ComptrollerStorage.sol#74) is read multiple times
In a function Comptroller._addMarketInternal(address) (contracts/Comptroller.sol#1064-1069) variable ComptrollerV5Storage.allMarkets (contracts/ComptrollerStorage.sol#99) is read multiple times
In a function Comp.getPriorVotes(address,uint256) (contracts/Governance/Comp.sol#197-229) variable Comp.checkpoints (contracts/Governance/Comp.sol#33) is read multiple times
In a function Unitroller._acceptImplementation() (contracts/Unitroller.sol#58-76) variable UnitrollerAdminStorage.pendingComptrollerImplementation (contracts/ComptrollerStorage.sol#26) is read multiple times


In a function Unitroller._acceptAdmin() (contracts/Unitroller.sol#107-127) variable UnitrollerAdminStorage.pendingAdmin (contracts/ComptrollerStorage.sol#16) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

ComptrollerStorage.sol

```
INFO:Detectors:
function CToken.nonReentrant() (contracts/CToken.sol#1222-1227) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initialize.md
INFO:Detectors:
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transfer(address,uint256) (contracts/EIP20NonStandardInterface.sol#35)
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transferFrom(address,address,uint256) (contracts/EIP20NonStandardInterface.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
INFO:Detectors:
CToken.accrueInterest() (contracts/CToken.sol#325-381) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/CToken.sol#331)
CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/CToken.sol#293)
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CToken.sol#27-60) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/CToken.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745):
        External calls:
        - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#737)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
                - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
        CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - totalSupply = totalSupply - redeemTokens (contracts/CToken.sol#548)
        CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87) can be used in cross function reentrancies:
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
CToken._addReservesFresh(uint256).actualAddAmount (contracts/CToken.sol#1055) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) shadows:
        - CTokenInterface.NewReserveGuardian(address,address) (contracts/CTokenInterfaces.sol#195) (event)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CToken.sol#27-60) should emit an event for:
        - initialExchangeRateMantissa = initialExchangeRateMantissa_ (contracts/CToken.sol#39)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
CToken._setPendingAdmin(address).newPendingAdmin (contracts/CToken.sol#902) lacks a zero-check on :
        - pendingAdmin = newPendingAdmin (contracts/CToken.sol#912)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#629)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#630)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#631)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        State variables written after the call(s):
        - accountTokens[minter] = accountTokens[minter] + mintTokens (contracts/CToken.sol#442)
        - totalSupply = totalSupply + mintTokens (contracts/CToken.sol#441)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - accountTokens[redeemer] = accountTokens[redeemer] - redeemTokens (contracts/CToken.sol#549)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#713)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#714)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
```

```
Reentrancy in CToken.seizeInternal(address,address,address,uint256) (contracts/CToken.sol#855-892):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        State variables written after the call(s):
        - accountTokens[borrower] = accountTokens[borrower] - seizeTokens (contracts/CToken.sol#885)
        - accountTokens[liquidator] = accountTokens[liquidator] + liquidatorSeizeTokens (contracts/CToken.sol#886)
        - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        - totalSupply = totalSupply - protocolSeizeTokens (contracts/CToken.sol#884)
Reentrancy in CToken.transferTokens(address,address,address,uint256) (contracts/CToken.sol#71-115):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/CToken.sol#73)
        State variables written after the call(s):
        - accountTokens[src] = srcTokensNew (contracts/CToken.sol#100)
        - accountTokens[dst] = dstTokensNew (contracts/CToken.sol#101)
        - transferAllowances[src][spender] = allowanceNew (contracts/CToken.sol#105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        Event emitted after the call(s):
        - Borrow(borrower,borrowAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#642)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
            - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
            - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
            - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
            - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        Event emitted after the call(s):
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
            - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
            - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
            - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
            - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
            - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(cTokenCollateral),seizeTokens) (contracts/CToken.sol#824)
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745):
        External calls:
        - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#737)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
            - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
            - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
            - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
            - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(cTokenCollateral),seizeTokens) (contracts/CToken.sol#824)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        Event emitted after the call(s):
        - Mint(minter,actualMintAmount,mintTokens) (contracts/CToken.sol#445)
        - Transfer(address(this),minter,mintTokens) (contracts/CToken.sol#446)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        Event emitted after the call(s):
        - Redeem(redeemer,redeemAmount,redeemTokens) (contracts/CToken.sol#561)
        - Transfer(redeemer,address(this),redeemTokens) (contracts/CToken.sol#560)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
Reentrancy in CToken.seizeInternal(address,address,address,uint256) (contracts/CToken.sol#855-892):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        Event emitted after the call(s):
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
Reentrancy in CToken.transferTokens(address,address,address,uint256) (contracts/CToken.sol#71-115):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/CToken.sol#73)
        Event emitted after the call(s):
        - Transfer(src,dst,tokens) (contracts/CToken.sol#109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Function CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) has a dubious typecast: address<=address
Function CToken._acceptAdmin() (contracts/CToken.sol#925-945) has a dubious typecast: address<=address
Function ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) has a dubious typecast: uint256<=uint256
Function ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) has a dubious typecast: uint256<=uint256
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
CToken._addReservesFresh(uint256) (contracts/CToken.sol#1052-1086) is never used and should be removed
CToken._addReservesInternal(uint256) (contracts/CToken.sol#1031-1044) is never used and should be removed
CToken.borrowBehalfInternal(uint256,address) (contracts/CToken.sol#583-589) is never used and should be removed
CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643) is never used and should be removed
CToken.borrowInternal(uint256) (contracts/CToken.sol#571-575) is never used and should be removed
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) is never used and should be removed
CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745) is never used and should be removed
CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451) is never used and should be removed
CToken.mintInternal(uint256) (contracts/CToken.sol#388-392) is never used and should be removed
CToken.redeemBehalfInternal(uint256,address) (contracts/CToken.sol#471-477) is never used and should be removed
CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565) is never used and should be removed
CToken.redeemInternal(uint256) (contracts/CToken.sol#458-462) is never used and should be removed
CToken.redeemUnderlyingInternal(uint256) (contracts/CToken.sol#484-488) is never used and should be removed
CToken.repayBorrowBehalfInternal(address,uint256) (contracts/CToken.sol#660-664) is never used and should be removed
CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721) is never used and should be removed
CToken.repayBorrowInternal(uint256) (contracts/CToken.sol#649-653) is never used and should be removed
ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) is never used and should be removed
ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#92-94) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#88-90) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#152-154) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#156-158) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#140-142) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#144-146) is never used and should be removed
ExponentialNoError.div_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#160-162) is never used and should be removed
ExponentialNoError.div_(uint256,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#148-150) is never used and should be removed
ExponentialNoError.div_(uint256,uint256) (contracts/ExponentialNoError.sol#164-166) is never used and should be removed
ExponentialNoError.fraction(uint256,uint256) (contracts/ExponentialNoError.sol#168-170) is never used and should be removed
ExponentialNoError.greaterThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#67-69) is never used and should be removed
ExponentialNoError.isZeroExp(ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#74-76) is never used and should be removed
ExponentialNoError.lessThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#53-55) is never used and should be removed
ExponentialNoError.lessThanOrEqualExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#60-62) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#124-126) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#128-130) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#112-114) is never used and should be removed
ExponentialNoError.mul_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#132-134) is never used and should be removed
ExponentialNoError.safe224(uint256,string) (contracts/ExponentialNoError.sol#78-81) is never used and should be removed
ExponentialNoError.safe32(uint256,string) (contracts/ExponentialNoError.sol#83-86) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#104-106) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#100-102) is never used and should be removed
ExponentialNoError.sub_(uint256,uint256) (contracts/ExponentialNoError.sol#108-110) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
INFO:Detectors:
Pragma version^0.8.10 (contracts/CToken.sol#2) allows old versions
Pragma version^0.8.10 (contracts/CTokenInterfaces.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ComptrollerInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ComptrollerStorage.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20Interface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20NonStandardInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ErrorReporter.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ExponentialNoError.sol#2) allows old versions
Pragma version^0.8.10 (contracts/InterestRateModel.sol#2) allows old versions
Pragma version^0.8.10 (contracts/PriceOracle.sol#2) allows old versions
Pragma version0.8.10 (contracts/Whitelist.sol#3) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function CToken._setPendingAdmin(address) (contracts/CToken.sol#902-918) is not in mixedCase
Function CToken._acceptAdmin() (contracts/CToken.sol#925-945) is not in mixedCase
Function CToken._setComptroller(ComptrollerInterface) (contracts/CToken.sol#952-969) is not in mixedCase
Function CToken._setReserveGuardian(address) (contracts/CToken.sol#976-979) is not in mixedCase
Function CToken._setReserveFactor(uint256) (contracts/CToken.sol#998-1002) is not in mixedCase
Function CToken._reduceReserves(uint256) (contracts/CToken.sol#1093-1097) is not in mixedCase
Function CToken._setInterestRateModel(InterestRateModel) (contracts/CToken.sol#1154-1158) is not in mixedCase
Variable CTokenStorage._notEntered (contracts/CTokenInterfaces.sol#13) is not in mixedCase
Constant CTokenStorage.borrowRateMaxMantissa (contracts/CTokenInterfaces.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant CTokenStorage.reserveFactorMaxMantissa (contracts/CTokenInterfaces.sol#34) is not in UPPER_CASE_WITH_UNDERSCORES
Function CTokenInterface._setPendingAdmin(address) (contracts/CTokenInterfaces.sol#260) is not in mixedCase
Function CTokenInterface._acceptAdmin() (contracts/CTokenInterfaces.sol#262) is not in mixedCase
Function CTokenInterface._setComptroller(ComptrollerInterface) (contracts/CTokenInterfaces.sol#264) is not in mixedCase
Function CTokenInterface._setReserveGuardian(address) (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Parameter CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Function CTokenInterface._setReserveFactor(uint256) (contracts/CTokenInterfaces.sol#268) is not in mixedCase
Function CTokenInterface._reduceReserves(uint256) (contracts/CTokenInterfaces.sol#270) is not in mixedCase
Function CTokenInterface._setInterestRateModel(InterestRateModel) (contracts/CTokenInterfaces.sol#272) is not in mixedCase
Function CErc20Interface._addReserves(uint256) (contracts/CTokenInterfaces.sol#311) is not in mixedCase
Function CDelegatorInterface._setImplementation(address,bool,bytes) (contracts/CTokenInterfaces.sol#333-337) is not in mixedCase
Function CDelegateInterface._becomeImplementation(bytes) (contracts/CTokenInterfaces.sol#346) is not in mixedCase
Function CDelegateInterface._resignImplementation() (contracts/CTokenInterfaces.sol#351) is not in mixedCase
Variable ComptrollerV2Storage._mintGuardianPaused (contracts/ComptrollerStorage.sol#82) is not in mixedCase
Variable ComptrollerV2Storage._borrowGuardianPaused (contracts/ComptrollerStorage.sol#83) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncate(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#37-40) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncateAddUInt(ExponentialNoError.Exp,uint256,uint256) (contracts/ExponentialNoError.sol#45-48) is not in mixedCase
Constant ExponentialNoError.expScale (contracts/ExponentialNoError.sol#12) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.doubleScale (contracts/ExponentialNoError.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.mantissaOne (contracts/ExponentialNoError.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter Whitelist.updateWhitelist(address,bool)._address (contracts/Whitelist.sol#11) is not in mixedCase
Parameter Whitelist.updateWhitelist(address,bool)._isActive (contracts/Whitelist.sol#12) is not in mixedCase
Parameter Whitelist.getWhitelisted(address)._address (contracts/Whitelist.sol#17) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable CToken.seizeInternal(address,address,address,uint256).seizeTokens (contracts/CToken.sol#855) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.seize(address,address,uint256).seizeTokens (contracts/CToken.sol#839) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface).seizeTokens (contracts/CToken.sol#806) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CTokenInterface.seize(address,address,uint256).seizeTokens (contracts/CTokenInterfaces.sol#256) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
CToken (contracts/CToken.sol#17-1228) does not implement functions:
        - CToken.doTransferIn(address,uint256) (contracts/CToken.sol#1208)
        - CToken.doTransferOut(address,uint256) (contracts/CToken.sol#1215)
        - CToken.getCashPrior() (contracts/CToken.sol#1202)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is never used in CToken (contracts/CToken.sol#17-1228)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
CDelegationStorage.implementation (contracts/CTokenInterfaces.sol#318) should be constant
CErc20Storage.underlying (contracts/CTokenInterfaces.sol#279) should be constant
ComptrollerV1Storage.closeFactorMantissa (contracts/ComptrollerStorage.sol#38) should be constant
ComptrollerV1Storage.liquidationIncentiveMantissa (contracts/ComptrollerStorage.sol#43) should be constant
ComptrollerV1Storage.maxAssets (contracts/ComptrollerStorage.sol#48) should be constant
ComptrollerV1Storage.oracle (contracts/ComptrollerStorage.sol#33) should be constant
ComptrollerV2Storage._borrowGuardianPaused (contracts/ComptrollerStorage.sol#83) should be constant
ComptrollerV2Storage._mintGuardianPaused (contracts/ComptrollerStorage.sol#82) should be constant
ComptrollerV2Storage.pauseGuardian (contracts/ComptrollerStorage.sol#81) should be constant
ComptrollerV2Storage.seizeGuardianPaused (contracts/ComptrollerStorage.sol#85) should be constant
ComptrollerV2Storage.transferGuardianPaused (contracts/ComptrollerStorage.sol#84) should be constant
ComptrollerV3Storage.compRate (contracts/ComptrollerStorage.sol#102) should be constant
ComptrollerV4Storage.borrowCapGuardian (contracts/ComptrollerStorage.sol#125) should be constant
ComptrollerV8Storage.supplyCapGuardian (contracts/ComptrollerStorage.sol#154) should be constant
ComptrollerV9Storage.reserveGuardian (contracts/ComptrollerStorage.sol#163) should be constant
ComptrollerV9Storage.speedGuardian (contracts/ComptrollerStorage.sol#165) should be constant
UnitrollerAdminStorage.admin (contracts/ComptrollerStorage.sol#11) should be constant
UnitrollerAdminStorage.comptrollerImplementation (contracts/ComptrollerStorage.sol#21) should be constant
UnitrollerAdminStorage.pendingAdmin (contracts/ComptrollerStorage.sol#16) should be constant
UnitrollerAdminStorage.pendingComptrollerImplementation (contracts/ComptrollerStorage.sol#26) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
In a function CToken._acceptAdmin() (contracts/CToken.sol#925-945) variable CTokenStorage.pendingAdmin (contracts/CTokenInterfaces.sol#44) is read multiple times
In a function CToken._reduceReservesFresh(uint256) (contracts/CToken.sol#1105-1146) variable CTokenStorage.admin (contracts/CTokenInterfaces.sol#39) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

CToken.sol

```
INFO:Detectors:
Function CToken.nonReentrant() (contracts/CToken.sol#1222-1227) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initialize.md
INFO:Detectors:
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transfer(address,uint256) (contracts/EIP20NonStandardInterface.sol#35)
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transferFrom(address,address,uint256) (contracts/EIP20NonStandardInterface.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
INFO:Detectors:
CToken.accrueInterest() (contracts/CToken.sol#325-381) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/CToken.sol#331)
CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/CToken.sol#293)
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CToken.sol#27-60) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/CToken.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745):
        External calls:
        - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#737)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
                - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
        CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - totalSupply = totalSupply - redeemTokens (contracts/CToken.sol#548)
        CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87) can be used in cross function reentrancies:
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
CToken._addReservesFresh(uint256).actualAddAmount (contracts/CToken.sol#1055) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) shadows:
        - CTokenInterfaceNewReserveGuardian(address,address) (contracts/CTokenInterfaces.sol#195) (event)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CToken.sol#27-60) should emit an event for:
        - initialExchangeRateMantissa = initialExchangeRateMantissa_ (contracts/CToken.sol#39)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
CToken._setPendingAdmin(address).newPendingAdmin (contracts/CToken.sol#902) lacks a zero-check on :
        - pendingAdmin = newPendingAdmin (contracts/CToken.sol#912)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#629)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#630)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#631)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        State variables written after the call(s):
        - accountTokens[minter] = accountTokens[minter] + mintTokens (contracts/CToken.sol#442)
        - totalSupply = totalSupply + mintTokens (contracts/CToken.sol#441)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - accountTokens[redeemer] = accountTokens[redeemer] - redeemTokens (contracts/CToken.sol#549)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#713)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#714)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
Reentrancy in CToken.seizeInternal(address,address,address,uint256) (contracts/CToken.sol#855-892):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        State variables written after the call(s):
        - accountTokens[borrower] = accountTokens[borrower] - seizeTokens (contracts/CToken.sol#885)
        - accountTokens[liquidator] = accountTokens[liquidator] + liquidatorSeizeTokens (contracts/CToken.sol#886)
        - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        - totalSupply = totalSupply - protocolSeizeTokens (contracts/CToken.sol#884)
Reentrancy in CToken.transferTokens(address,address,address,uint256) (contracts/CToken.sol#71-115):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/CToken.sol#73)
        State variables written after the call(s):
        - accountTokens[src] = srcTokensNew (contracts/CToken.sol#100)
        - accountTokens[dst] = dstTokensNew (contracts/CToken.sol#101)
        - transferAllowances[src][spender] = allowanceNew (contracts/CToken.sol#105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        Event emitted after the call(s):
        - Borrow(borrower,borrowAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#642)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
                - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        Event emitted after the call(s):
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(cTokenCollateral),seizeTokens) (contracts/CToken.sol#824)
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745):
        External calls:
        - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#737)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
                - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(cTokenCollateral),seizeTokens) (contracts/CToken.sol#824)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        Event emitted after the call(s):
        - Mint(minter,actualMintAmount,mintTokens) (contracts/CToken.sol#445)
        - Transfer(address(this),minter,mintTokens) (contracts/CToken.sol#446)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        Event emitted after the call(s):
        - Redeem(redeemer,redeemAmount,redeemTokens) (contracts/CToken.sol#561)
        - Transfer(redeemer,address(this),redeemTokens) (contracts/CToken.sol#560)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
Reentrancy in CToken.seizeInternal(address,address,address,uint256) (contracts/CToken.sol#855-892):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        Event emitted after the call(s):
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
Reentrancy in CToken.transferTokens(address,address,address,uint256) (contracts/CToken.sol#71-115):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/CToken.sol#73)
        Event emitted after the call(s):
        - Transfer(src,dst,tokens) (contracts/CToken.sol#109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Function CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) has a dubious typecast: address<-address
Function CToken._acceptAdmin() (contracts/CToken.sol#925-945) has a dubious typecast: address<-address
Function ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) has a dubious typecast: uint256<-uint256
Function ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) has a dubious typecast: uint256<-uint256
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
CToken._addReservesFresh(uint256) (contracts/CToken.sol#1052-1086) is never used and should be removed
CToken._addReservesInternal(uint256) (contracts/CToken.sol#1038-1044) is never used and should be removed
CToken.borrowBehalfInternal(uint256,address) (contracts/CToken.sol#583-589) is never used and should be removed
CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643) is never used and should be removed
```

```
CToken.borrowInternal(uint256) (contracts/CToken.sol#571-575) is never used and should be removed
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) is never used and should be removed
CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745) is never used and should be removed
CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451) is never used and should be removed
CToken.mintInternal(uint256) (contracts/CToken.sol#388-392) is never used and should be removed
CToken.redeemBehalfInternal(uint256,address) (contracts/CToken.sol#471-477) is never used and should be removed
CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565) is never used and should be removed
CToken.redeemInternal(uint256) (contracts/CToken.sol#458-462) is never used and should be removed
CToken.redeemUnderlyingInternal(uint256) (contracts/CToken.sol#484-488) is never used and should be removed
CToken.repayBorrowBehalfInternal(address,uint256) (contracts/CToken.sol#660-664) is never used and should be removed
CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721) is never used and should be removed
CToken.repayBorrowInternal(uint256) (contracts/CToken.sol#649-653) is never used and should be removed
ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) is never used and should be removed
ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#92-94) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#88-90) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#152-154) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#156-158) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#140-142) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#144-146) is never used and should be removed
ExponentialNoError.div_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#160-162) is never used and should be removed
ExponentialNoError.div_(uint256,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#148-150) is never used and should be removed
ExponentialNoError.div_(uint256,uint256) (contracts/ExponentialNoError.sol#164-166) is never used and should be removed
ExponentialNoError.fraction(uint256,uint256) (contracts/ExponentialNoError.sol#168-170) is never used and should be removed
ExponentialNoError.greaterThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#67-69) is never used and should be removed
ExponentialNoError.isZeroExp(ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#74-76) is never used and should be removed
ExponentialNoError.lessThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#53-55) is never used and should be removed
ExponentialNoError.lessThanOrEqualExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#60-62) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#124-126) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#128-130) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#112-114) is never used and should be removed
ExponentialNoError.mul_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#132-134) is never used and should be removed
ExponentialNoError.safe224(uint256,string) (contracts/ExponentialNoError.sol#78-81) is never used and should be removed
ExponentialNoError.safe32(uint256,string) (contracts/ExponentialNoError.sol#83-86) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#104-106) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#100-102) is never used and should be removed
ExponentialNoError.sub_(uint256,uint256) (contracts/ExponentialNoError.sol#108-110) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.10 (contracts/CToken.sol#2) allows old versions
Pragma version^0.8.10 (contracts/CTokenInterfaces.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ComptrollerInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20Interface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20NonStandardInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ErrorReporter.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ExponentialNoError.sol#2) allows old versions
Pragma version^0.8.10 (contracts/InterestRateModel.sol#2) allows old versions
Pragma version0.8.10 (contracts/Whitelist.sol#3) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function CToken._setPendingAdmin(address) (contracts/CToken.sol#902-918) is not in mixedCase
Function CToken._acceptAdmin() (contracts/CToken.sol#925-945) is not in mixedCase
Function CToken._setComptroller(ComptrollerInterface) (contracts/CToken.sol#952-969) is not in mixedCase
Function CToken._setReserveGuardian(address) (contracts/CToken.sol#976-979) is not in mixedCase
Function CToken._setReserveFactor(uint256) (contracts/CToken.sol#998-1002) is not in mixedCase
Function CToken._reduceReserves(uint256) (contracts/CToken.sol#1093-1097) is not in mixedCase
Function CToken._setInterestRateModel(InterestRateModel) (contracts/CToken.sol#1154-1158) is not in mixedCase
Variable CTokenStorage._notEntered (contracts/CTokenInterfaces.sol#23) is not in mixedCase
Constant CTokenStorage.borrowRateMaxMantissa (contracts/CTokenInterfaces.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant CTokenStorage.reserveFactorMaxMantissa (contracts/CTokenInterfaces.sol#34) is not in UPPER_CASE_WITH_UNDERSCORES
Function CTokenInterface._setPendingAdmin(address) (contracts/CTokenInterfaces.sol#260) is not in mixedCase
Function CTokenInterface._acceptAdmin() (contracts/CTokenInterfaces.sol#262) is not in mixedCase
Function CTokenInterface._setComptroller(ComptrollerInterface) (contracts/CTokenInterfaces.sol#264) is not in mixedCase
Function CTokenInterface._setReserveGuardian(address) (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Parameter CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Function CTokenInterface._setReserveFactor(uint256) (contracts/CTokenInterfaces.sol#268) is not in mixedCase
Function CTokenInterface._reduceReserves(uint256) (contracts/CTokenInterfaces.sol#270) is not in mixedCase
Function CTokenInterface._setInterestRateModel(InterestRateModel) (contracts/CTokenInterfaces.sol#272) is not in mixedCase
Function CErc20Interface._addReserves(uint256) (contracts/CTokenInterfaces.sol#311) is not in mixedCase
Function CDelegatorInterface._setImplementation(address,bool,bytes) (contracts/CTokenInterfaces.sol#333-337) is not in mixedCase
Function CDelegateInterface._becomeImplementation(bytes) (contracts/CTokenInterfaces.sol#346) is not in mixedCase
Function CDelegateInterface._resignImplementation() (contracts/CTokenInterfaces.sol#351) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncate(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#37-40) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncateAddUInt(ExponentialNoError.Exp,uint256,uint256) (contracts/ExponentialNoError.sol#45-48) is not in mixedCase
Constant ExponentialNoError.expScale (contracts/ExponentialNoError.sol#12) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.doubleScale (contracts/ExponentialNoError.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.mantissaOne (contracts/ExponentialNoError.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter Whitelist.updateWhitelist(address,bool)._address (contracts/Whitelist.sol#11) is not in mixedCase
Parameter Whitelist.updateWhitelist(address,bool)._isActive (contracts/Whitelist.sol#12) is not in mixedCase
Parameter Whitelist.getWhitelisted(address)._address (contracts/Whitelist.sol#17) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable CToken.seizeInternal(address,address,address,uint256).seizeTokens (contracts/CToken.sol#855) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.seize(address,address,uint256).seizeTokens (contracts/CToken.sol#839) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface).seizeTokens (contracts/CToken.sol#806) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CTokenInterface.seize(address,address,uint256).seizeTokens (contracts/CTokenInterfaces.sol#256) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
CToken (contracts/CToken.sol#17-1228) does not implement functions:
        - CToken.doTransferIn(address,uint256) (contracts/CToken.sol#1208)
        - CToken.doTransferOut(address,uint256) (contracts/CToken.sol#1215)
        - CToken.getCashPrior() (contracts/CToken.sol#1202)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is never used in CToken (contracts/CToken.sol#17-1228)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
CDelegationStorage.implementation (contracts/CTokenInterfaces.sol#318) should be constant
CErc20Storage.underlying (contracts/CTokenInterfaces.sol#279) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
In a function CToken._acceptAdmin() (contracts/CToken.sol#925-945) variable CTokenStorage.pendingAdmin (contracts/CTokenInterfaces.sol#44) is read multiple times
In a function CToken._reduceReservesFresh(uint256) (contracts/CToken.sol#1105-1146) variable CTokenStorage.admin (contracts/CTokenInterfaces.sol#39) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

ERC20.sol

```
INFO:Detectors:
NonStandardToken (contracts/ERC20.sol#80-116) has incorrect ERC20 function interface:ERC20NS.transfer(address,uint256) (contracts/ERC20.sol#25)
NonStandardToken (contracts/ERC20.sol#80-116) has incorrect ERC20 function interface:ERC20NS.transferFrom(address,address,uint256) (contracts/ERC20.sol#27)
NonStandardToken (contracts/ERC20.sol#80-116) has incorrect ERC20 function interface:NonStandardToken.transfer(address,uint256) (contracts/ERC20.sol#98-102)
NonStandardToken (contracts/ERC20.sol#80-116) has incorrect ERC20 function interface:NonStandardToken.transferFrom(address,address,uint256) (contracts/ERC20.sol#104-109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
INFO:Detectors:
SafeMath.add(uint256,uint256) (contracts/SafeMath.sol#29-35) is never used and should be removed
SafeMath.div(uint256,uint256) (contracts/SafeMath.sol#137-139) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/SafeMath.sol#152-159) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/SafeMath.sol#172-174) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/SafeMath.sol#187-190) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/SafeMath.sol#88-101) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (contracts/SafeMath.sol#111-124) is never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/SafeMath.sol#61-63) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.10 (contracts/ERC20.sol#1) allows old versions
Pragma version^0.8.10 (contracts/SafeMath.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter StandardToken.approve(address,uint256)._spender (contracts/ERC20.sol#68) is not in mixedCase
Parameter NonStandardToken.approve(address,uint256)._spender (contracts/ERC20.sol#111) is not in mixedCase
Parameter ERC20Harness.harnessSetFailTransferFromAddress(address,bool)._fail (contracts/ERC20.sol#132) is not in mixedCase
Parameter ERC20Harness.harnessSetFailTransferToAddress(address,bool)._fail (contracts/ERC20.sol#136) is not in mixedCase
Parameter ERC20Harness.harnessSetBalance(address,uint256)._account (contracts/ERC20.sol#140) is not in mixedCase
Parameter ERC20Harness.harnessSetBalance(address,uint256)._amount (contracts/ERC20.sol#140) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
NonStandardToken.decimals (contracts/ERC20.sol#84) should be immutable
NonStandardToken.totalSupply (contracts/ERC20.sol#86) should be immutable
StandardToken.decimals (contracts/ERC20.sol#40) should be immutable
StandardToken.totalSupply (contracts/ERC20.sol#41) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

ErrorReporter.sol

```
INFO:Detectors:
Function ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) has a dubious typecast: uint256<=uint256
Function ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) has a dubious typecast: uint256<=uint256
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) is never used and should be removed
ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.10 (contracts/ErrorReporter.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

Exponential.sol

```
INFO:Detectors:
Exponential.divScalarByExpTruncate(uint256,ExponentialNoError.Exp).fraction (contracts/Exponential.sol#124) shadows:
        - ExponentialNoError.fraction(uint256,uint256) (contracts/ExponentialNoError.sol#168-170) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
CarefulMath.addThenSubUInt(uint256,uint256,uint256) (contracts/CarefulMath.sol#76-84) is never used and should be removed
CarefulMath.addUInt(uint256,uint256) (contracts/CarefulMath.sol#63-71) is never used and should be removed
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) is never used and should be removed
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) is never used and should be removed
CarefulMath.subUInt(uint256,uint256) (contracts/CarefulMath.sol#52-58) is never used and should be removed
Exponential.addExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/Exponential.sol#37-41) is never used and should be removed
Exponential.divExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/Exponential.sol#180-182) is never used and should be removed
Exponential.divScalar(ExponentialNoError.Exp,uint256) (contracts/Exponential.sol#91-98) is never used and should be removed
Exponential.divScalarByExp(uint256,ExponentialNoError.Exp) (contracts/Exponential.sol#103-118) is never used and should be removed
Exponential.divScalarByExpTruncate(uint256,ExponentialNoError.Exp) (contracts/Exponential.sol#123-130) is never used and should be removed
Exponential.getExp(uint256,uint256) (contracts/Exponential.sol#20-32) is never used and should be removed
Exponential.mulExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/Exponential.sol#135-155) is never used and should be removed
Exponential.mulExp(uint256,uint256) (contracts/Exponential.sol#160-162) is never used and should be removed
Exponential.mulExp3(ExponentialNoError.Exp,ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/Exponential.sol#167-173) is never used and should be removed
Exponential.mulScalar(ExponentialNoError.Exp,uint256) (contracts/Exponential.sol#55-62) is never used and should be removed
Exponential.mulScalarTruncate(ExponentialNoError.Exp,uint256) (contracts/Exponential.sol#67-74) is never used and should be removed
Exponential.mulScalarTruncateAddUInt(ExponentialNoError.Exp,uint256,uint256) (contracts/Exponential.sol#79-86) is never used and should be removed
Exponential.subExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/Exponential.sol#46-50) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#92-94) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#88-90) is never used and should be removed
ExponentialNoError.add_(uint256,uint256) (contracts/ExponentialNoError.sol#96-98) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#152-154) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#156-158) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#140-142) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#144-146) is never used and should be removed
ExponentialNoError.div_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#160-162) is never used and should be removed
ExponentialNoError.div_(uint256,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#148-150) is never used and should be removed
ExponentialNoError.div_(uint256,uint256) (contracts/ExponentialNoError.sol#164-166) is never used and should be removed
ExponentialNoError.fraction(uint256,uint256) (contracts/ExponentialNoError.sol#168-170) is never used and should be removed
ExponentialNoError.greaterThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#67-69) is never used and should be removed
ExponentialNoError.isZeroExp(ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#74-76) is never used and should be removed
ExponentialNoError.lessThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#53-55) is never used and should be removed
ExponentialNoError.lessThanOrEqualExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#60-62) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#124-126) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#128-130) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#112-114) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#116-118) is never used and should be removed
ExponentialNoError.mul_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#132-134) is never used and should be removed
ExponentialNoError.mul_(uint256,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#120-122) is never used and should be removed
ExponentialNoError.mul_(uint256,uint256) (contracts/ExponentialNoError.sol#136-138) is never used and should be removed
ExponentialNoError.mul_ScalarTruncate(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#37-40) is never used and should be removed
ExponentialNoError.mul_ScalarTruncateAddUInt(ExponentialNoError.Exp,uint256,uint256) (contracts/ExponentialNoError.sol#45-48) is never used and should be removed
ExponentialNoError.safe224(uint256,string) (contracts/ExponentialNoError.sol#78-81) is never used and should be removed
ExponentialNoError.safe32(uint256,string) (contracts/ExponentialNoError.sol#83-86) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#104-106) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#100-102) is never used and should be removed
ExponentialNoError.sub_(uint256,uint256) (contracts/ExponentialNoError.sol#108-110) is never used and should be removed
ExponentialNoError.truncate(ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#29-32) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.10 (contracts/CarefulMath.sol#2) allows old versions
Pragma version^0.8.10 (contracts/Exponential.sol#1) allows old versions
Pragma version^0.8.10 (contracts/ExponentialNoError.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function ExponentialNoError.mul_ScalarTruncate(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#37-40) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncateAddUInt(ExponentialNoError.Exp,uint256,uint256) (contracts/ExponentialNoError.sol#45-48) is not in mixedCase
Constant ExponentialNoError.expScale (contracts/ExponentialNoError.sol#12) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.doubleScale (contracts/ExponentialNoError.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.mantissaOne (contracts/ExponentialNoError.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
ExponentialNoError.mantissaOne (contracts/ExponentialNoError.sol#15) is never used in Exponential (contracts/Exponential.sol#14-184)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
```

ExponentialNoError.sol

```
INFO:Detectors:
ExponentialNoError.add_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#92-94) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#88-90) is never used and should be removed
ExponentialNoError.add_(uint256,uint256) (contracts/ExponentialNoError.sol#96-98) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#152-154) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#156-158) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#140-142) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#144-146) is never used and should be removed
ExponentialNoError.div_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#160-162) is never used and should be removed
ExponentialNoError.div_(uint256,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#148-150) is never used and should be removed
ExponentialNoError.div_(uint256,uint256) (contracts/ExponentialNoError.sol#164-166) is never used and should be removed
ExponentialNoError.fraction(uint256,uint256) (contracts/ExponentialNoError.sol#168-170) is never used and should be removed
ExponentialNoError.greaterThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#67-69) is never used and should be removed
ExponentialNoError.isZeroExp(ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#74-76) is never used and should be removed
ExponentialNoError.lessThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#53-55) is never used and should be removed
ExponentialNoError.lessThanOrEqualExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#60-62) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#124-126) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#128-130) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#112-114) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#116-118) is never used and should be removed
ExponentialNoError.mul_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#132-134) is never used and should be removed
ExponentialNoError.mul_(uint256,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#120-122) is never used and should be removed
ExponentialNoError.mul_(uint256,uint256) (contracts/ExponentialNoError.sol#136-138) is never used and should be removed
ExponentialNoError.mul_ScalarTruncate(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#37-40) is never used and should be removed
ExponentialNoError.mul_ScalarTruncateAddUInt(ExponentialNoError.Exp,uint256,uint256) (contracts/ExponentialNoError.sol#45-48) is never used and should be removed
ExponentialNoError.safe224(uint256,string) (contracts/ExponentialNoError.sol#78-81) is never used and should be removed
ExponentialNoError.safe32(uint256,string) (contracts/ExponentialNoError.sol#83-86) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#104-106) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#100-102) is never used and should be removed
ExponentialNoError.sub_(uint256,uint256) (contracts/ExponentialNoError.sol#108-110) is never used and should be removed
ExponentialNoError.truncate(ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#29-32) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.10 (contracts/ExponentialNoError.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function ExponentialNoError.mul_ScalarTruncate(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#37-40) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncateAddUInt(ExponentialNoError.Exp,uint256,uint256) (contracts/ExponentialNoError.sol#45-48) is not in mixedCase
Constant ExponentialNoError.expScale (contracts/ExponentialNoError.sol#12) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.doubleScale (contracts/ExponentialNoError.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.mantissaOne (contracts/ExponentialNoError.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is never used in ExponentialNoError (contracts/ExponentialNoError.sol#11-171)
ExponentialNoError.mantissaOne (contracts/ExponentialNoError.sol#15) is never used in ExponentialNoError (contracts/ExponentialNoError.sol#11-171)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
```

Fauceteer.sol

```
INFO:Detectors:
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transfer(address,uint256) (contracts/EIP20NonStandardInterface.sol#35)
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transferFrom(address,address,uint256) (contracts/EIP20NonStandardInterface.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
INFO:Detectors:
Fauceteer.drip(EIP20NonStandardInterface) (contracts/Fauceteer.sol#17-38) uses assembly
        - INLINE ASM (contracts/Fauceteer.sol#23-35)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Pragma version^0.8.10 (contracts/EIP20NonStandardInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/Fauceteer.sol#1) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function Fauceteer.drip(EIP20NonStandardInterface) (contracts/Fauceteer.sol#17-38) contains magic numbers: 10000, 32
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
```

InterestRateModel.sol

```
INFO:Detectors:
Pragma version^0.8.10 (contracts/InterestRateModel.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

JumpRateModel.sol

```
INFO:Detectors:
JumpRateModel.getSupplyRate(uint256,uint256,uint256,uint256) (contracts/JumpRateModel.sol#99-109) performs a multiplication on the result of a division:
        - rateToPool = (borrowRate * oneMinusReserveFactor) / BASE (contracts/JumpRateModel.sol#107)
        - (utilizationRate(cash,borrows,reserves) * rateToPool) / BASE (contracts/JumpRateModel.sol#108)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Pragma version^0.8.10 (contracts/InterestRateModel.sol#2) allows old versions
Pragma version^0.8.10 (contracts/JumpRateModel.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
JumpRateModel.baseRatePerBlock (contracts/JumpRateModel.sol#28) should be immutable
JumpRateModel.jumpMultiplierPerBlock (contracts/JumpRateModel.sol#33) should be immutable
JumpRateModel.kink (contracts/JumpRateModel.sol#38) should be immutable
JumpRateModel.multiplierPerBlock (contracts/JumpRateModel.sol#23) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Detectors:
In a function JumpRateModel.constructor(uint256,uint256,uint256,uint256) (contracts/JumpRateModel.sol#47-54) variable JumpRateModel.blocksPerYear (contracts/JumpRateModel.sol#18) is read multiple times
In a function JumpRateModel.getBorrowRate(uint256,uint256,uint256) (contracts/JumpRateModel.sol#79-89) variable JumpRateModel.BASE (contracts/JumpRateModel.sol#13) is read multiple times
In a function JumpRateModel.getBorrowRate(uint256,uint256,uint256) (contracts/JumpRateModel.sol#79-89) variable JumpRateModel.kink (contracts/JumpRateModel.sol#38) is read multiple times
In a function JumpRateModel.getSupplyRate(uint256,uint256,uint256,uint256) (contracts/JumpRateModel.sol#99-109) variable JumpRateModel.BASE (contracts/JumpRateModel.sol#13) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

JumpRateModelV2.sol

```
INFO:Detectors:
BaseJumpRateModelV2.getSupplyRate(uint256,uint256,uint256,uint256) (contracts/BaseJumpRateModelV2.sol#122-132) performs a multiplication on the result of a division:
        - rateToPool = (borrowRate * oneMinusReserveFactor) / BASE (contracts/BaseJumpRateModelV2.sol#130)
        - (utilizationRate(cash,borrows,reserves) * rateToPool) / BASE (contracts/BaseJumpRateModelV2.sol#131)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Pragma version^0.8.10 (contracts/BaseJumpRateModelV2.sol#2) allows old versions
Pragma version^0.8.10 (contracts/InterestRateModel.sol#2) allows old versions
Pragma version^0.8.10 (contracts/JumpRateModelV2.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
BaseJumpRateModelV2.owner (contracts/BaseJumpRateModelV2.sol#19) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

Maximillion.sol

```
INFO:Detectors:
Maximillion.repayBehalfExplicit(address,CEther) (contracts/Maximillion.sol#38-47) sends eth to arbitrary user
        Dangerous calls:
        - cether_.repayBorrowBehalf{value: borrows}(borrower) (contracts/Maximillion.sol#42)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transfer(address,uint256) (contracts/EIP20NonStandardInterface.sol#35)
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transferFrom(address,address,uint256) (contracts/EIP20NonStandardInterface.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
INFO:Detectors:
CToken.accrueInterest() (contracts/CToken.sol#325-381) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/CToken.sol#331)
CEther.doTransferIn(address,uint256) (contracts/CEther.sol#166-171) uses a dangerous strict equality:
        - require(bool,string)(msg.value == amount,value mismatch) (contracts/CEther.sol#169)
CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/CToken.sol#293)
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CToken.sol#27-60) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/CToken.sol#36)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == NO_ERROR,LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/CToken.sol#811)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) uses a dangerous strict equality:
        - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745):
        External calls:
        - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#737)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
                - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
        CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - totalSupply = totalSupply - redeemTokens (contracts/CToken.sol#548)
        CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87) can be used in cross function reentrancies:
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
CToken._addReservesFresh(uint256).actualAddAmount (contracts/CToken.sol#1055) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) shadows:
        - CTokenInterfaceNewReserveGuardian(address,address) (contracts/CTokenInterfaces.sol#195) (event)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
```

```
INFO:Detectors:
CToken._setPendingAdmin(address).newPendingAdmin (contracts/CToken.sol#902) lacks a zero-check on :
        - pendingAdmin = newPendingAdmin (contracts/CToken.sol#912)
CEther.constructor(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8,address).admin_ (contracts/CEther.sol#29) lacks a zero-check on :
        - admin = admin_ (contracts/CEther.sol#37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#629)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#630)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#631)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        State variables written after the call(s):
        - accountTokens[minter] = accountTokens[minter] + mintTokens (contracts/CToken.sol#442)
        - totalSupply = totalSupply + mintTokens (contracts/CToken.sol#441)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - accountTokens[redeemer] = accountTokens[redeemer] - redeemTokens (contracts/CToken.sol#549)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#713)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#714)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
Reentrancy in CToken.seizeInternal(address,address,address,uint256) (contracts/CToken.sol#855-892):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        State variables written after the call(s):
        - accountTokens[borrower] = accountTokens[borrower] - seizeTokens (contracts/CToken.sol#885)
        - accountTokens[liquidator] = accountTokens[liquidator] + liquidatorSeizeTokens (contracts/CToken.sol#886)
        - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        - totalSupply = totalSupply - protocolSeizeTokens (contracts/CToken.sol#884)
Reentrancy in CToken.transferTokens(address,address,address,uint256) (contracts/CToken.sol#71-115):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/CToken.sol#73)
        State variables written after the call(s):
        - accountTokens[src] = srcTokensNew (contracts/CToken.sol#100)
        - accountTokens[dst] = dstTokensNew (contracts/CToken.sol#101)
        - transferAllowances[src][spender] = allowanceNew (contracts/CToken.sol#105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        Event emitted after the call(s):
        - Borrow(borrower,borrowAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#642)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
                - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        Event emitted after the call(s):
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(cTokenCollateral),seizeTokens) (contracts/CToken.sol#824)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        Event emitted after the call(s):
        - Mint(minter,actualMintAmount,mintTokens) (contracts/CToken.sol#445)
        - Transfer(address(this),minter,mintTokens) (contracts/CToken.sol#446)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        Event emitted after the call(s):
        - Redeem(redeemer,redeemAmount,redeemTokens) (contracts/CToken.sol#561)
        - Transfer(redeemer,address(this),redeemTokens) (contracts/CToken.sol#560)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Function CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) has a dubious typecast: address<=address
Function CToken._acceptAdmin() (contracts/CToken.sol#925-945) has a dubious typecast: address<=address
Function ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) has a dubious typecast: uint256<=uint256
Function ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) has a dubious typecast: uint256<=uint256
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) is never used and should be removed
ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#92-94) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#88-90) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#152-154) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#156-158) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#140-142) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#144-146) is never used and should be removed
ExponentialNoError.div_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#160-162) is never used and should be removed
ExponentialNoError.fraction(uint256,uint256) (contracts/ExponentialNoError.sol#168-170) is never used and should be removed
ExponentialNoError.greaterThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#67-69) is never used and should be removed
ExponentialNoError.isZeroExp(ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#74-76) is never used and should be removed
ExponentialNoError.lessThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#53-55) is never used and should be removed
ExponentialNoError.lessThanOrEqualExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#60-62) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#124-126) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#128-130) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#112-114) is never used and should be removed
ExponentialNoError.mul_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#132-134) is never used and should be removed
ExponentialNoError.safe224(uint256,string) (contracts/ExponentialNoError.sol#78-81) is never used and should be removed
ExponentialNoError.safe32(uint256,string) (contracts/ExponentialNoError.sol#83-86) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#104-106) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#100-102) is never used and should be removed
ExponentialNoError.sub_(uint256,uint256) (contracts/ExponentialNoError.sol#108-110) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.10 (contracts/CEther.sol#2) allows old versions
Pragma version^0.8.10 (contracts/CToken.sol#2) allows old versions
Pragma version^0.8.10 (contracts/CTokenInterfaces.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ComptrollerInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20Interface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20NonStandardInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ErrorReporter.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ExponentialNoError.sol#2) allows old versions
Pragma version^0.8.10 (contracts/InterestRateModel.sol#2) allows old versions
Pragma version^0.8.10 (contracts/Maximillion.sol#2) allows old versions
Pragma version0.8.10 (contracts/Whitelist.sol#3) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function CEther._addReserves() (contracts/CEther.sol#138-140) is not in mixedCase
Function CToken._setPendingAdmin(address) (contracts/CToken.sol#902-918) is not in mixedCase
Function CToken._acceptAdmin() (contracts/CToken.sol#925-945) is not in mixedCase
Function CToken._setComptroller(ComptrollerInterface) (contracts/CToken.sol#952-969) is not in mixedCase
Function CToken._setReserveGuardian(address) (contracts/CToken.sol#976-979) is not in mixedCase
Function CToken._setReserveFactor(uint256) (contracts/CToken.sol#998-1002) is not in mixedCase
Function CToken._reduceReserves(uint256) (contracts/CToken.sol#1091-1097) is not in mixedCase
```

```
Function CToken._setInterestRateModel(InterestRateModel) (contracts/CToken.sol#1154-1158) is not in mixedCase
Variable CTokenStorage._notEntered (contracts/CTokenInterfaces.sol#13) is not in mixedCase
Constant CTokenStorage.borrowRateMaxMantissa (contracts/CTokenInterfaces.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant CTokenStorage.reserveFactorMaxMantissa (contracts/CTokenInterfaces.sol#34) is not in UPPER_CASE_WITH_UNDERSCORES
Function CTokenInterface._setPendingAdmin(address) (contracts/CTokenInterfaces.sol#260) is not in mixedCase
Function CTokenInterface._acceptAdmin() (contracts/CTokenInterfaces.sol#262) is not in mixedCase
Function CTokenInterface._setComptroller(ComptrollerInterface) (contracts/CTokenInterfaces.sol#264) is not in mixedCase
Function CTokenInterface._setReserveGuardian(address) (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Parameter CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Function CTokenInterface._setReserveFactor(uint256) (contracts/CTokenInterfaces.sol#268) is not in mixedCase
Function CTokenInterface._reduceReserves(uint256) (contracts/CTokenInterfaces.sol#270) is not in mixedCase
Function CTokenInterface._setInterestRateModel(InterestRateModel) (contracts/CTokenInterfaces.sol#272) is not in mixedCase
Function CErc20Interface._addReserves(uint256) (contracts/CTokenInterfaces.sol#311) is not in mixedCase
Function CDelegatorInterface._setImplementation(address,bool,bytes) (contracts/CTokenInterfaces.sol#333-337) is not in mixedCase
Function CDelegateInterface._becomeImplementation(bytes) (contracts/CTokenInterfaces.sol#345) is not in mixedCase
Function CDelegateInterface._resignImplementation() (contracts/CTokenInterfaces.sol#351) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncate(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#37-40) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncateAddUInt(ExponentialNoError.Exp,uint256,uint256) (contracts/ExponentialNoError.sol#45-48) is not in mixedCase
Constant ExponentialNoError.expScale (contracts/ExponentialNoError.sol#12) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.doubleScale (contracts/ExponentialNoError.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.mantissaOne (contracts/ExponentialNoError.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter Whitelist.updateWhitelist(address,bool)._address (contracts/Whitelist.sol#11) is not in mixedCase
Parameter Whitelist.updateWhitelist(address,bool)._isActive (contracts/Whitelist.sol#12) is not in mixedCase
Parameter Whitelist.getWhitelisted(address)._address (contracts/Whitelist.sol#17) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable CTokenInterface.seize(address,address,uint256).seizeTokens (contracts/CTokenInterfaces.sol#256) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface).seizeTokens (contracts/CToken.sol#806) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.seizeInternal(address,address,address,uint256).seizeTokens (contracts/CToken.sol#855) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.seize(address,address,uint256).seizeTokens (contracts/CToken.sol#839) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is never used in CEther (contracts/CEther.sol#11-177)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
CDelegationStorage.implementation (contracts/CTokenInterfaces.sol#338) should be constant
CErc20Storage.underlying (contracts/CTokenInterfaces.sol#279) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
Maximillion.cEther (contracts/Maximillion.sol#14) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

PriceOracle.sol

```
INFO:Detectors:
Function CToken.nonReentrant() (contracts/CToken.sol#1222-1227) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initialize.md
INFO:Detectors:
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transfer(address,uint256) (contracts/EIP20NonStandardInterface.sol#35)
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transferFrom(address,address,uint256) (contracts/EIP20NonStandardInterface.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
INFO:Detectors:
CToken.accrueInterest() (contracts/CToken.sol#325-381) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/CToken.sol#331)
CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/CToken.sol#293)
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CToken.sol#27-60) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/CToken.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745):
        External calls:
        - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#737)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
                - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
        CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - totalSupply = totalSupply - redeemTokens (contracts/CToken.sol#548)
        CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87) can be used in cross function reentrancies:
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
CToken._addReservesFresh(uint256).actualAddAmount (contracts/CToken.sol#1055) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) shadows:
        - CTokenInterfaceNewReserveGuardian(address,address) (contracts/CTokenInterfaces.sol#195) (event)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CToken.sol#27-60) should emit an event for:
        - initialExchangeRateMantissa = initialExchangeRateMantissa_ (contracts/CToken.sol#39)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
CToken._setPendingAdmin(address).newPendingAdmin (contracts/CToken.sol#902) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (contracts/CToken.sol#912)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#629)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#630)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#631)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        State variables written after the call(s):
        - accountTokens[minter] = accountTokens[minter] + mintTokens (contracts/CToken.sol#442)
        - totalSupply = totalSupply + mintTokens (contracts/CToken.sol#441)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - accountTokens[redeemer] = accountTokens[redeemer] - redeemTokens (contracts/CToken.sol#549)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#713)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#714)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
Reentrancy in CToken.seizeInternal(address,address,address,uint256) (contracts/CToken.sol#855-892):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        State variables written after the call(s):
        - accountTokens[borrower] = accountTokens[borrower] - seizeTokens (contracts/CToken.sol#885)
        - accountTokens[liquidator] = accountTokens[liquidator] + liquidatorSeizeTokens (contracts/CToken.sol#886)
        - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        - totalSupply = totalSupply - protocolSeizeTokens (contracts/CToken.sol#884)
```

```
Reentrancy in CToken.transferTokens(address,address,address,uint256) (contracts/CToken.sol#71-115):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/CToken.sol#73)
        State variables written after the call(s):
        - accountTokens[src] = srcTokensNew (contracts/CToken.sol#100)
        - accountTokens[dst] = dstTokensNew (contracts/CToken.sol#101)
        - transferAllowances[src][spender] = allowanceNew (contracts/CToken.sol#105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        Event emitted after the call(s):
        - Borrow(borrower,borrowAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#642)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
                - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        Event emitted after the call(s):
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(cTokenCollateral),seizeTokens) (contracts/CToken.sol#824)
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745):
        External calls:
        - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#737)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
                - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(cTokenCollateral),seizeTokens) (contracts/CToken.sol#824)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        Event emitted after the call(s):
        - Mint(minter,actualMintAmount,mintTokens) (contracts/CToken.sol#445)
        - Transfer(address(this),minter,mintTokens) (contracts/CToken.sol#446)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        Event emitted after the call(s):
        - Redeem(redeemer,redeemAmount,redeemTokens) (contracts/CToken.sol#561)
        - Transfer(redeemer,address(this),redeemTokens) (contracts/CToken.sol#560)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
Reentrancy in CToken.seizeInternal(address,address,address,uint256) (contracts/CToken.sol#855-892):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        Event emitted after the call(s):
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
Reentrancy in CToken.transferTokens(address,address,address,uint256) (contracts/CToken.sol#71-115):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/CToken.sol#73)
        Event emitted after the call(s):
        - Transfer(src,dst,tokens) (contracts/CToken.sol#109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Function CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) has a dubious typecast: address<=address
Function CToken._acceptAdmin() (contracts/CToken.sol#925-945) has a dubious typecast: address<=address
Function ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) has a dubious typecast: uint256<=uint256
Function ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) has a dubious typecast: uint256<=uint256
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
CToken._addReservesFresh(uint256) (contracts/CToken.sol#1052-1086) is never used and should be removed
CToken._addReservesInternal(uint256) (contracts/CToken.sol#1038-1044) is never used and should be removed
CToken.borrowBehalfInternal(uint256,address) (contracts/CToken.sol#583-589) is never used and should be removed
CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643) is never used and should be removed
CToken.borrowInternal(uint256) (contracts/CToken.sol#571-575) is never used and should be removed
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) is never used and should be removed
CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745) is never used and should be removed
CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451) is never used and should be removed
CToken.mintInternal(uint256) (contracts/CToken.sol#388-392) is never used and should be removed
CToken.redeemBehalfInternal(uint256,address) (contracts/CToken.sol#471-477) is never used and should be removed
CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565) is never used and should be removed
CToken.redeemInternal(uint256) (contracts/CToken.sol#458-462) is never used and should be removed
CToken.redeemUnderlyingInternal(uint256) (contracts/CToken.sol#484-488) is never used and should be removed
CToken.repayBorrowBehalfInternal(address,uint256) (contracts/CToken.sol#660-664) is never used and should be removed
CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721) is never used and should be removed
CToken.repayBorrowInternal(uint256) (contracts/CToken.sol#649-653) is never used and should be removed
ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) is never used and should be removed
ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#92-94) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#88-90) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#152-154) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#156-158) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#140-142) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#144-146) is never used and should be removed
ExponentialNoError.div_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#160-162) is never used and should be removed
ExponentialNoError.div_(uint256,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#148-150) is never used and should be removed
ExponentialNoError.div_(uint256,uint256) (contracts/ExponentialNoError.sol#164-166) is never used and should be removed
ExponentialNoError.fraction(uint256,uint256) (contracts/ExponentialNoError.sol#168-170) is never used and should be removed
ExponentialNoError.greaterThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#67-69) is never used and should be removed
ExponentialNoError.isZeroExp(ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#74-76) is never used and should be removed
ExponentialNoError.lessThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#53-55) is never used and should be removed
ExponentialNoError.lessThanOrEqualExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#60-62) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#124-126) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#128-130) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#112-114) is never used and should be removed
ExponentialNoError.mul_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#132-134) is never used and should be removed
ExponentialNoError.safe224(uint256,string) (contracts/ExponentialNoError.sol#70-81) is never used and should be removed
ExponentialNoError.safe32(uint256,string) (contracts/ExponentialNoError.sol#83-86) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#104-106) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#100-102) is never used and should be removed
ExponentialNoError.sub_(uint256,uint256) (contracts/ExponentialNoError.sol#108-110) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.10 (contracts/CToken.sol#2) allows old versions
Pragma version^0.8.10 (contracts/CTokenInterfaces.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ComptrollerInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20Interface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20NonStandardInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ErrorReporter.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ExponentialNoError.sol#2) allows old versions
Pragma version^0.8.10 (contracts/InterestRateModel.sol#2) allows old versions
Pragma version^0.8.10 (contracts/PriceOracle.sol#2) allows old versions
Pragma version0.8.10 (contracts/Whitelist.sol#3) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
INFO:Detectors:
Function CToken._setPendingAdmin(address) (contracts/CToken.sol#902-918) is not in mixedCase
Function CToken._acceptAdmin() (contracts/CToken.sol#925-945) is not in mixedCase
Function CToken._setComptroller(ComptrollerInterface) (contracts/CToken.sol#952-969) is not in mixedCase
Function CToken._setReserveGuardian(address) (contracts/CToken.sol#976-979) is not in mixedCase
Function CToken._setReserveFactor(uint256) (contracts/CToken.sol#998-1002) is not in mixedCase
Function CToken._reduceReserves(uint256) (contracts/CToken.sol#1093-1097) is not in mixedCase
Function CToken._setInterestRateModel(InterestRateModel) (contracts/CToken.sol#1154-1158) is not in mixedCase
Variable CTokenStorage._notEntered (contracts/CTokenInterfaces.sol#13) is not in mixedCase
Constant CTokenStorage.borrowRateMaxMantissa (contracts/CTokenInterfaces.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant CTokenStorage.reserveFactorMaxMantissa (contracts/CTokenInterfaces.sol#34) is not in UPPER_CASE_WITH_UNDERSCORES
Function CTokenInterface._setPendingAdmin(address) (contracts/CTokenInterfaces.sol#260) is not in mixedCase
Function CTokenInterface._acceptAdmin() (contracts/CTokenInterfaces.sol#262) is not in mixedCase
Function CTokenInterface._setComptroller(ComptrollerInterface) (contracts/CTokenInterfaces.sol#264) is not in mixedCase
Function CTokenInterface._setReserveGuardian(address) (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Parameter CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Function CTokenInterface._setReserveFactor(uint256) (contracts/CTokenInterfaces.sol#268) is not in mixedCase
Function CTokenInterface._reduceReserves(uint256) (contracts/CTokenInterfaces.sol#270) is not in mixedCase
Function CTokenInterface._setInterestRateModel(InterestRateModel) (contracts/CTokenInterfaces.sol#272) is not in mixedCase
Function CErc20Interface._addReserves(uint256) (contracts/CTokenInterfaces.sol#311) is not in mixedCase
Function CDelegatorInterface._setImplementation(address,bool,bytes) (contracts/CTokenInterfaces.sol#333-337) is not in mixedCase
Function CDelegateInterface._becomeImplementation(bytes) (contracts/CTokenInterfaces.sol#346) is not in mixedCase
Function CDelegateInterface._resignImplementation() (contracts/CTokenInterfaces.sol#351) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncate(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#37-40) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncateAddUInt(ExponentialNoError.Exp,uint256,uint256) (contracts/ExponentialNoError.sol#45-48) is not in mixedCase
Constant ExponentialNoError.expScale (contracts/ExponentialNoError.sol#12) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.doubleScale (contracts/ExponentialNoError.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.mantissaOne (contracts/ExponentialNoError.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter Whitelist.updateWhitelist(address,bool)._address (contracts/Whitelist.sol#11) is not in mixedCase
Parameter Whitelist.updateWhitelist(address,bool)._isActive (contracts/Whitelist.sol#12) is not in mixedCase
Parameter Whitelist.getWhitelisted(address)._address (contracts/Whitelist.sol#17) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable CToken.seizeInternal(address,address,address,uint256).seizeTokens (contracts/CToken.sol#855) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.seize(address,address,uint256).seizeTokens (contracts/CTokenInterfaces.sol#256) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.seize(address,address,uint256).seizeTokens (contracts/CToken.sol#839) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface).seizeTokens (contracts/CToken.sol#806) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
CToken (contracts/CToken.sol#17-1228) does not implement functions:
        - CToken.doTransferIn(address,uint256) (contracts/CToken.sol#1208)
        - CToken.doTransferOut(address,uint256) (contracts/CToken.sol#1215)
        - CToken.getCashPrior() (contracts/CToken.sol#1202)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is never used in CToken (contracts/CToken.sol#17-1228)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
CDelegationStorage.implementation (contracts/CTokenInterfaces.sol#318) should be constant
CErc20Storage.underlying (contracts/CTokenInterfaces.sol#279) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
In a function CToken._acceptAdmin() (contracts/CToken.sol#925-945) variable CTokenStorage.pendingAdmin (contracts/CTokenInterfaces.sol#44) is read multiple times
In a function CToken._reduceReservesFresh(uint256) (contracts/CToken.sol#1105-1146) variable CTokenStorage.admin (contracts/CTokenInterfaces.sol#39) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

Reservoir.sol

```
INFO:Detectors:
Reservoir.drip() (contracts/Reservoir.sol#48-69) ignores return value by token_.transfer(target_,toDrip_) (contracts/Reservoir.sol#66)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
Reservoir.mul(uint256,uint256,string) (contracts/Reservoir.sol#86-94) uses a dangerous strict equality:
        - require(bool,string)(c / a == b,errorMessage) (contracts/Reservoir.sol#92)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reservoir.constructor(uint256,EIP20Interface,address).target_ (contracts/Reservoir.sol#35) lacks a zero-check on :
        - target = target_ (contracts/Reservoir.sol#39)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Pragma version^0.8.10 (contracts/EIP20Interface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/Reservoir.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Reservoir.dripRate (contracts/Reservoir.sol#18) should be immutable
Reservoir.dripStart (contracts/Reservoir.sol#15) should be immutable
Reservoir.target (contracts/Reservoir.sol#24) should be immutable
Reservoir.token (contracts/Reservoir.sol#21) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

SimplePriceOracle.sol

```
INFO:Detectors:
Function CErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CErc20.sol#26-41) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initialize.md
INFO:Detectors:
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transfer(address,uint256) (contracts/EIP20NonStandardInterface.sol#35)
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transferFrom(address,address,uint256) (contracts/EIP20NonStandardInterface.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
INFO:Detectors:
CToken.accrueInterest() (contracts/CToken.sol#325-381) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/CToken.sol#331)
CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310) uses a dangerous strict equality:
        - totalSupply == 0 (contracts/CToken.sol#293)
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CToken.sol#27-60) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/CToken.sol#36)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) uses a dangerous strict equality:
        - require(bool,string)(amountSeized == NO_ERROR,LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/CToken.sol#811)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) uses a dangerous strict equality:
        - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745):
        External calls:
        - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#737)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
                - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
        CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - totalSupply = totalSupply - redeemTokens (contracts/CToken.sol#548)
        CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87) can be used in cross function reentrancies:
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
CToken._addReservesFresh(uint256).actualAddAmount (contracts/CToken.sol#1055) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
CErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CErc20.sol#26-41) ignores return value by EIP20Interface(underlying).totalSupply() (contracts/CErc20.sol#40)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) shadows:
        - CTokenInterface.NewReserveGuardian(address,address) (contracts/CTokenInterfaces.sol#195) (event)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
```

```
INFO:Detectors:
CToken._setPendingAdmin(address).newPendingAdmin (contracts/CToken.sol#902) lacks a zero-check on :
        - pendingAdmin = newPendingAdmin (contracts/CToken.sol#912)
CErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8).underlying_ (contracts/CErc20.sol#27) lacks a zero-check on :
        - underlying = underlying_ (contracts/CErc20.sol#39)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#629)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#630)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#631)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        State variables written after the call(s):
        - accountTokens[minter] = accountTokens[minter] + mintTokens (contracts/CToken.sol#442)
        - totalSupply = totalSupply + mintTokens (contracts/CToken.sol#441)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - accountTokens[redeemer] = accountTokens[redeemer] - redeemTokens (contracts/CToken.sol#549)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#713)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#714)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
Reentrancy in CToken.seizeInternal(address,address,address,uint256) (contracts/CToken.sol#855-892):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        State variables written after the call(s):
        - accountTokens[borrower] = accountTokens[borrower] - seizeTokens (contracts/CToken.sol#885)
        - accountTokens[liquidator] = accountTokens[liquidator] + liquidatorSeizeTokens (contracts/CToken.sol#886)
        - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        - totalSupply = totalSupply - protocolSeizeTokens (contracts/CToken.sol#884)
Reentrancy in CToken.transferTokens(address,address,address,uint256) (contracts/CToken.sol#71-115):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/CToken.sol#73)
        State variables written after the call(s):
        - accountTokens[src] = srcTokensNew (contracts/CToken.sol#100)
        - accountTokens[dst] = dstTokensNew (contracts/CToken.sol#101)
        - transferAllowances[src][spender] = allowanceNew (contracts/CToken.sol#105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        Event emitted after the call(s):
        - Borrow(borrower,borrowAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#642)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
                - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        Event emitted after the call(s):
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(cTokenCollateral),seizeTokens) (contracts/CToken.sol#824)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        Event emitted after the call(s):
        - Mint(minter,actualMintAmount,mintTokens) (contracts/CToken.sol#445)
        - Transfer(address(this),minter,mintTokens) (contracts/CToken.sol#446)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        Event emitted after the call(s):
        - Redeem(redeemer,redeemAmount,redeemTokens) (contracts/CToken.sol#561)
        - Transfer(redeemer,address(this),redeemTokens) (contracts/CToken.sol#560)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Function CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) has a dubious typecast: address<=address
Function CToken._acceptAdmin() (contracts/CToken.sol#925-945) has a dubious typecast: address<=address
Function CErc20.doTransferIn(address,uint256) (contracts/CErc20.sol#190-219) has a dubious typecast: address<=EIP20Interface
Function ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) has a dubious typecast: uint256<=uint256
Function ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) has a dubious typecast: uint256<=uint256
Function SimplePriceOracle._getUnderlyingAddress(CToken) (contracts/SimplePriceOracle.sol#11-19) has a dubious typecast: CErc20<=address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
CErc20.doTransferIn(address,uint256) (contracts/CErc20.sol#190-219) uses assembly
        - INLINE ASM (contracts/CErc20.sol#198-213)
CErc20.doTransferOut(address,uint256) (contracts/CErc20.sol#230-252) uses assembly
        - INLINE ASM (contracts/CErc20.sol#235-250)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) is never used and should be removed
ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#92-94) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#88-90) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#152-154) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#156-158) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#140-142) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#144-146) is never used and should be removed
ExponentialNoError.div_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#160-162) is never used and should be removed
ExponentialNoError.fraction(uint256,uint256) (contracts/ExponentialNoError.sol#168-170) is never used and should be removed
ExponentialNoError.greaterThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#67-69) is never used and should be removed
ExponentialNoError.isZeroExp(ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#74-76) is never used and should be removed
ExponentialNoError.lessThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#53-55) is never used and should be removed
ExponentialNoError.lessThanOrEqualExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#60-62) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#124-126) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#128-130) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#112-114) is never used and should be removed
ExponentialNoError.mul_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#132-134) is never used and should be removed
ExponentialNoError.safe224(uint256,string) (contracts/ExponentialNoError.sol#78-81) is never used and should be removed
ExponentialNoError.safe32(uint256,string) (contracts/ExponentialNoError.sol#83-86) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#104-106) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#100-102) is never used and should be removed
ExponentialNoError.sub_(uint256,uint256) (contracts/ExponentialNoError.sol#108-110) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.10 (contracts/CErc20.sol#2) allows old versions
Pragma version^0.8.10 (contracts/CToken.sol#2) allows old versions
Pragma version^0.8.10 (contracts/CTokenInterfaces.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ComptrollerInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20Interface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20NonStandardInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ErrorReporter.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ExponentialNoError.sol#2) allows old versions
Pragma version^0.8.10 (contracts/InterestRateModel.sol#2) allows old versions
Pragma version^0.8.10 (contracts/PriceOracle.sol#2) allows old versions
Pragma version^0.8.10 (contracts/SimplePriceOracle.sol#2) allows old versions
Pragma version0.8.10 (contracts/Whitelist.sol#3) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
INFO:Detectors:
Function CErc20._addReserves(uint256) (contracts/CErc20.sol#165-167) is not in mixedCase
Function CErc20._delegateCompLikeTo(address) (contracts/CErc20.sol#259-262) is not in mixedCase
Function CToken._setPendingAdmin(address) (contracts/CToken.sol#902-918) is not in mixedCase
Function CToken._acceptAdmin() (contracts/CToken.sol#925-945) is not in mixedCase
Function CToken._setComptroller(ComptrollerInterface) (contracts/CToken.sol#952-969) is not in mixedCase
Function CToken._setReserveGuardian(address) (contracts/CToken.sol#976-979) is not in mixedCase
Function CToken._setReserveFactor(uint256) (contracts/CToken.sol#998-1002) is not in mixedCase
Function CToken._reduceReserves(uint256) (contracts/CToken.sol#1093-1097) is not in mixedCase
Function CToken._setInterestRateModel(InterestRateModel) (contracts/CToken.sol#1154-1158) is not in mixedCase
Variable CTokenStorage._notEntered (contracts/CTokenInterfaces.sol#13) is not in mixedCase
Constant CTokenStorage.borrowRateMaxMantissa (contracts/CTokenInterfaces.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant CTokenStorage.reserveFactorMaxMantissa (contracts/CTokenInterfaces.sol#34) is not in UPPER_CASE_WITH_UNDERSCORES
Function CTokenInterface._setPendingAdmin(address) (contracts/CTokenInterfaces.sol#260) is not in mixedCase
Function CTokenInterface._acceptAdmin() (contracts/CTokenInterfaces.sol#262) is not in mixedCase
Function CTokenInterface._setComptroller(ComptrollerInterface) (contracts/CTokenInterfaces.sol#264) is not in mixedCase
Function CTokenInterface._setReserveGuardian(address) (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Parameter CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Function CTokenInterface._setReserveFactor(uint256) (contracts/CTokenInterfaces.sol#268) is not in mixedCase
Function CTokenInterface._reduceReserves(uint256) (contracts/CTokenInterfaces.sol#270) is not in mixedCase
Function CTokenInterface._setInterestRateModel(InterestRateModel) (contracts/CTokenInterfaces.sol#272) is not in mixedCase
Function CErc20Interface._addReserves(uint256) (contracts/CTokenInterfaces.sol#311) is not in mixedCase
Function CDelegatorInterface._setImplementation(address,bool,bytes) (contracts/CTokenInterfaces.sol#333-337) is not in mixedCase
Function CDelegateInterface._becomeImplementation(bytes) (contracts/CTokenInterfaces.sol#345) is not in mixedCase
Function CDelegateInterface._resignImplementation() (contracts/CTokenInterfaces.sol#351) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncate(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#37-40) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncateAddUInt(ExponentialNoError.Exp,uint256,uint256) (contracts/ExponentialNoError.sol#45-48) is not in mixedCase
Constant ExponentialNoError.expScale (contracts/ExponentialNoError.sol#12) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.doubleScale (contracts/ExponentialNoError.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.mantissaOne (contracts/ExponentialNoError.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter Whitelist.updateWhitelist(address,bool)._address (contracts/Whitelist.sol#11) is not in mixedCase
Parameter Whitelist.updateWhitelist(address,bool)._isActive (contracts/Whitelist.sol#12) is not in mixedCase
Parameter Whitelist.getWhitelisted(address)._address (contracts/Whitelist.sol#17) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable CToken.seize(address,address,uint256).seizeTokens (contracts/CToken.sol#839) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface).seizeTokens (contracts/CToken.sol#806) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.seizeInternal(address,address,address,uint256).seizeTokens (contracts/CToken.sol#855) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CTokenInterface.seize(address,address,uint256).seizeTokens (contracts/CTokenInterfaces.sol#256) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
SimplePriceOracle (contracts/SimplePriceOracle.sol#7-44) does not implement functions:
        - PriceOracle.getLodePrice(address) (contracts/PriceOracle.sol#18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is never used in CErc20 (contracts/CErc20.sol#15-263)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
Function CErc20.doTransferIn(address,uint256) (contracts/CErc20.sol#190-219) contains magic number: 32
Function CErc20.doTransferOut(address,uint256) (contracts/CErc20.sol#230-252) contains magic number: 32
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
CDelegationStorage.implementation (contracts/CTokenInterfaces.sol#318) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

Timelock.sol

```
INFO:Detectors:
Function Timelock.executeTransaction(address,uint256,string,bytes,uint256) (contracts/Timelock.sol#116-147) contains a low level call to a custom address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/call_forward_to_protected.md
INFO:Detectors:
Timelock.constructor(address,uint256).admin_ (contracts/Timelock.sol#47) lacks a zero-check on :
        - admin = admin_ (contracts/Timelock.sol#51)
Timelock.setPendingAdmin(address).pendingAdmin_ (contracts/Timelock.sol#74) lacks a zero-check on :
        - pendingAdmin = pendingAdmin_ (contracts/Timelock.sol#76)
Timelock.executeTransaction(address,uint256,string,bytes,uint256).target (contracts/Timelock.sol#117) lacks a zero-check on :
        - (success,returnData) = target.call{value: value}(callData) (contracts/Timelock.sol#141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Timelock.executeTransaction(address,uint256,string,bytes,uint256) (contracts/Timelock.sol#116-147):
        External calls:
        - (success,returnData) = target.call{value: value}(callData) (contracts/Timelock.sol#141)
        Event emitted after the call(s):
        - ExecuteTransaction(txHash,target,value,signature,data,eta) (contracts/Timelock.sol#144)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Timelock.queueTransaction(address,uint256,string,bytes,uint256) (contracts/Timelock.sol#81-99) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(eta >= getBlockTimestamp().add(delay),Timelock::queueTransaction: Estimated execution block must satisfy delay.) (contracts/Timelock.sol#89-92)
Timelock.executeTransaction(address,uint256,string,bytes,uint256) (contracts/Timelock.sol#116-147) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(getBlockTimestamp() >= eta,Timelock::executeTransaction: Transaction hasn't surpassed time lock.) (contracts/Timelock.sol#127)
        - require(bool,string)(getBlockTimestamp() <= eta.add(GRACE_PERIOD),Timelock::executeTransaction: Transaction is stale.) (contracts/Timelock.sol#128)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
SafeMath.add(uint256,uint256,string) (contracts/SafeMath.sol#45-51) is never used and should be removed
SafeMath.div(uint256,uint256) (contracts/SafeMath.sol#137-139) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/SafeMath.sol#152-159) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/SafeMath.sol#172-174) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/SafeMath.sol#187-190) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/SafeMath.sol#88-101) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (contracts/SafeMath.sol#111-124) is never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/SafeMath.sol#61-63) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (contracts/SafeMath.sol#73-78) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.10 (contracts/SafeMath.sol#2) allows old versions
Pragma version^0.8.10 (contracts/Timelock.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
low level call in Timelock.executeTransaction(address,uint256,string,bytes,uint256) (contracts/Timelock.sol#116-147):
        - (success,returnData) = target.call{value: value}(callData) (contracts/Timelock.sol#141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

Unitroller.sol

```
INFO:Detectors:
Unitroller.fallback() (contracts/Unitroller.sol#134-150) uses delegatecall to a input-controlled function id
        - (success) = comptrollerImplementation.delegatecall(msg.data) (contracts/Unitroller.sol#136)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-delegatecall
INFO:Detectors:
Function CToken.nonReentrant() (contracts/CToken.sol#1222-1227) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initialize.md
INFO:Detectors:
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transfer(address,uint256) (contracts/EIP20NonStandardInterface.sol#35)
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transferFrom(address,address,uint256) (contracts/EIP20NonStandardInterface.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
INFO:Detectors:
CToken.accrueInterest() (contracts/CToken.sol#325-381) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/CToken.sol#331)
CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/CToken.sol#293)
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CToken.sol#27-60) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/CToken.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745):
        External calls:
        - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#737)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
                - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
        CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalReserves = totalReservesNew (contracts/CToken.sol#883)
```

```
        CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - totalSupply = totalSupply - redeemTokens (contracts/CToken.sol#548)
        CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87) can be used in cross function reentrancies:
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
CToken._addReservesFresh(uint256).actualAddAmount (contracts/CToken.sol#1055) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) shadows:
        - CTokenInterfaceNewReserveGuardian(address,address) (contracts/CTokenInterfaces.sol#195) (event)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CToken.sol#27-60) should emit an event for:
        - initialExchangeRateMantissa = initialExchangeRateMantissa_ (contracts/CToken.sol#39)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
CToken._setPendingAdmin(address).newPendingAdmin (contracts/CToken.sol#902) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (contracts/CToken.sol#912)
Unitroller._setPendingImplementation(address).newPendingImplementation (contracts/Unitroller.sol#39) lacks a zero-check on :
                - pendingComptrollerImplementation = newPendingImplementation (contracts/Unitroller.sol#46)
Unitroller._setPendingAdmin(address).newPendingAdmin (contracts/Unitroller.sol#84) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (contracts/Unitroller.sol#94)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#629)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#630)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#631)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        State variables written after the call(s):
        - accountTokens[minter] = accountTokens[minter] + mintTokens (contracts/CToken.sol#442)
        - totalSupply = totalSupply + mintTokens (contracts/CToken.sol#441)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - accountTokens[redeemer] = accountTokens[redeemer] - redeemTokens (contracts/CToken.sol#549)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#713)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#714)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
Reentrancy in CToken.seizeInternal(address,address,address,uint256) (contracts/CToken.sol#855-892):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        State variables written after the call(s):
        - accountTokens[borrower] = accountTokens[borrower] - seizeTokens (contracts/CToken.sol#885)
        - accountTokens[liquidator] = accountTokens[liquidator] + liquidatorSeizeTokens (contracts/CToken.sol#886)
        - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        - totalSupply = totalSupply - protocolSeizeTokens (contracts/CToken.sol#884)
Reentrancy in CToken.transferTokens(address,address,address,uint256) (contracts/CToken.sol#71-115):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/CToken.sol#73)
        State variables written after the call(s):
        - accountTokens[src] = srcTokensNew (contracts/CToken.sol#100)
        - accountTokens[dst] = dstTokensNew (contracts/CToken.sol#101)
        - transferAllowances[src][spender] = allowanceNew (contracts/CToken.sol#105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        Event emitted after the call(s):
        - Borrow(borrower,borrowAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#642)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
                - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        Event emitted after the call(s):
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(cTokenCollateral),seizeTokens) (contracts/CToken.sol#824)
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745):
        External calls:
        - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#737)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
                - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(cTokenCollateral),seizeTokens) (contracts/CToken.sol#824)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        Event emitted after the call(s):
        - Mint(minter,actualMintAmount,mintTokens) (contracts/CToken.sol#445)
        - Transfer(address(this),minter,mintTokens) (contracts/CToken.sol#446)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        Event emitted after the call(s):
        - Redeem(redeemer,redeemAmount,redeemTokens) (contracts/CToken.sol#561)
        - Transfer(redeemer,address(this),redeemTokens) (contracts/CToken.sol#560)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
Reentrancy in CToken.seizeInternal(address,address,address,uint256) (contracts/CToken.sol#855-892):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        Event emitted after the call(s):
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
Reentrancy in CToken.transferTokens(address,address,address,uint256) (contracts/CToken.sol#71-115):
```

Whitelist.sol

```
INFO:Detectors:
Pragma version0.8.10 (contracts/Whitelist.sol#3) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter Whitelist.updateWhitelist(address,bool)._address (contracts/Whitelist.sol#11) is not in mixedCase
Parameter Whitelist.updateWhitelist(address,bool)._isActive (contracts/Whitelist.sol#12) is not in mixedCase
Parameter Whitelist.getWhitelisted(address)._address (contracts/Whitelist.sol#17) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

WhitePaperInterestRateModel.sol

```
INFO:Detectors:
WhitePaperInterestRateModel.getSupplyRate(uint256,uint256,uint256,uint256) (contracts/WhitePaperInterestRateModel.sol#79-89) performs a multiplication on the result of a division:
        - rateToPool = (borrowRate * oneMinusReserveFactor) / BASE (contracts/WhitePaperInterestRateModel.sol#87)
        - (utilizationRate(cash,borrows,reserves) * rateToPool) / BASE (contracts/WhitePaperInterestRateModel.sol#88)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Pragma version^0.8.10 (contracts/InterestRateModel.sol#2) allows old versions
Pragma version^0.8.10 (contracts/WhitePaperInterestRateModel.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
WhitePaperInterestRateModel.baseRatePerBlock (contracts/WhitePaperInterestRateModel.sol#29) should be immutable
WhitePaperInterestRateModel.multiplierPerBlock (contracts/WhitePaperInterestRateModel.sol#24) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Detectors:
In a function WhitePaperInterestRateModel.getSupplyRate(uint256,uint256,uint256,uint256) (contracts/WhitePaperInterestRateModel.sol#79-89) variable WhitePaperInterestRateModel.BASE (contracts/WhitePaperInterestRateModel.sol#14) is read m
ultiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

Comp.sol

```
INFO:Detectors:
Comp._writeCheckpoint(address,uint32,uint96,uint96) (contracts/Governance/Comp.sol#270-281) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (contracts/Governance/Comp.sol#273)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Comp.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (contracts/Governance/Comp.sol#167-178) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,Comp::delegateBySig: signature expired) (contracts/Governance/Comp.sol#176)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Comp.getChainId() (contracts/Governance/Comp.sol#304-310) uses assembly
        - INLINE ASM (contracts/Governance/Comp.sol#306-308)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Pragma version^0.8.10 (contracts/Governance/Comp.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Comp.slitherConstructorConstantVariables() (contracts/Governance/Comp.sol#4-311) uses literals with too many digits:
        - totalSupply = 20000000e18 (contracts/Governance/Comp.sol#15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
Function Comp.approve(address,uint256) (contracts/Governance/Comp.sol#88-100) contains magic number: 96
Function Comp.transfer(address,uint256) (contracts/Governance/Comp.sol#117-121) contains magic number: 96
Function Comp.transferFrom(address,address,uint256) (contracts/Governance/Comp.sol#130-148) contains magic number: 96
Function Comp._writeCheckpoint(address,uint32,uint96,uint96) (contracts/Governance/Comp.sol#270-281) contains magic number: 32
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
In a function Comp.getPriorVotes(address,uint256) (contracts/Governance/Comp.sol#197-229) variable Comp.checkpoints (contracts/Governance/Comp.sol#33) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

GovernorAlpha.sol

```
INFO:Detectors:
GovernorAlpha.execute(uint256) (contracts/Governance/GovernorAlpha.sol#235-252) sends eth to arbitrary user
        Dangerous calls:
        - timelock.executeTransaction{value: proposal.values[i]}(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/Governance/GovernorAlpha.sol#243-249)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in GovernorAlpha.queue(uint256) (contracts/Governance/GovernorAlpha.sol#213-225):
        External calls:
        - _queueOrRevert(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],eta) (contracts/Governance/GovernorAlpha.sol#221)
                - timelock.queueTransaction(target,value,signature,data,eta) (contracts/Governance/GovernorAlpha.sol#232)
        State variables written after the call(s):
        - proposal.eta = eta (contracts/Governance/GovernorAlpha.sol#223)
        GovernorAlpha.proposals (contracts/Governance/GovernorAlpha.sol#99) can be used in cross function reentrancies:
        - GovernorAlpha._castVote(address,uint256,bool) (contracts/Governance/GovernorAlpha.sol#331-349)
        - GovernorAlpha.cancel(uint256) (contracts/Governance/GovernorAlpha.sol#254-277)
        - GovernorAlpha.execute(uint256) (contracts/Governance/GovernorAlpha.sol#235-252)
        - GovernorAlpha.getActions(uint256) (contracts/Governance/GovernorAlpha.sol#279-288)
        - GovernorAlpha.getReceipt(uint256,address) (contracts/Governance/GovernorAlpha.sol#290-292)
        - GovernorAlpha.proposals (contracts/Governance/GovernorAlpha.sol#99)
        - GovernorAlpha.propose(address[],uint256[],string[],bytes[],string) (contracts/Governance/GovernorAlpha.sol#142-211)
        - GovernorAlpha.queue(uint256) (contracts/Governance/GovernorAlpha.sol#213-225)
        - GovernorAlpha.state(uint256) (contracts/Governance/GovernorAlpha.sol#294-314)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
GovernorAlpha._queueOrRevert(address,uint256,string,bytes,uint256) (contracts/Governance/GovernorAlpha.sol#227-233) ignores return value by timelock.queueTransaction(target,value,signature,data,eta) (contracts/Governance/GovernorAlpha.so
l#232)
GovernorAlpha.execute(uint256) (contracts/Governance/GovernorAlpha.sol#235-252) ignores return value by timelock.executeTransaction{value: proposal.values[i]}(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldat
as[i],proposal.eta) (contracts/Governance/GovernorAlpha.sol#243-249)
GovernorAlpha.__queueSetTimelockPendingAdmin(address,uint256) (contracts/Governance/GovernorAlpha.sol#361-364) ignores return value by timelock.queueTransaction(address(timelock),0,setPendingAdmin(address),abi.encode(newPendingAdmin),eta
) (contracts/Governance/GovernorAlpha.sol#363)
GovernorAlpha.__executeSetTimelockPendingAdmin(address,uint256) (contracts/Governance/GovernorAlpha.sol#366-369) ignores return value by timelock.executeTransaction(address(timelock),0,setPendingAdmin(address),abi.encode(newPendingAdmin)
,eta) (contracts/Governance/GovernorAlpha.sol#368)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
GovernorAlpha.cancel(uint256).state (contracts/Governance/GovernorAlpha.sol#255) shadows:
        - GovernorAlpha.state(uint256) (contracts/Governance/GovernorAlpha.sol#294-314) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
GovernorAlpha.constructor(address,address,address).guardian_ (contracts/Governance/GovernorAlpha.sol#136) lacks a zero-check on :
                - guardian = guardian_ (contracts/Governance/GovernorAlpha.sol#139)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
GovernorAlpha._queueOrRevert(address,uint256,string,bytes,uint256) (contracts/Governance/GovernorAlpha.sol#227-233) has external calls inside a loop: require(bool,string)(! timelock.queuedTransactions(keccak256(bytes)(abi.encode(target,v
alue,signature,data,eta))),GovernorAlpha::_queueOrRevert: proposal action already queued at eta) (contracts/Governance/GovernorAlpha.sol#228-231)
GovernorAlpha._queueOrRevert(address,uint256,string,bytes,uint256) (contracts/Governance/GovernorAlpha.sol#227-233) has external calls inside a loop: timelock.queueTransaction(target,value,signature,data,eta) (contracts/Governance/Governo
rAlpha.sol#232)
GovernorAlpha.execute(uint256) (contracts/Governance/GovernorAlpha.sol#235-252) has external calls inside a loop: timelock.executeTransaction{value: proposal.values[i]}(proposal.targets[i],proposal.values[i],proposal.signatures[i],propos
al.calldatas[i],proposal.eta) (contracts/Governance/GovernorAlpha.sol#243-249)
GovernorAlpha.cancel(uint256) (contracts/Governance/GovernorAlpha.sol#254-277) has external calls inside a loop: timelock.cancelTransaction(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta)
 (contracts/Governance/GovernorAlpha.sol#267-273)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in GovernorAlpha.cancel(uint256) (contracts/Governance/GovernorAlpha.sol#254-277):
        External calls:
        - timelock.cancelTransaction(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/Governance/GovernorAlpha.sol#267-273)
        Event emitted after the call(s):
        - ProposalCanceled(proposalId) (contracts/Governance/GovernorAlpha.sol#276)
Reentrancy in GovernorAlpha.execute(uint256) (contracts/Governance/GovernorAlpha.sol#235-252):
        External calls:
        - timelock.executeTransaction{value: proposal.values[i]}(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/Governance/GovernorAlpha.sol#243-249)
        Event emitted after the call(s):
        - ProposalExecuted(proposalId) (contracts/Governance/GovernorAlpha.sol#251)
Reentrancy in GovernorAlpha.queue(uint256) (contracts/Governance/GovernorAlpha.sol#213-225):
        External calls:
        - _queueOrRevert(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],eta) (contracts/Governance/GovernorAlpha.sol#221)
                - timelock.queueTransaction(target,value,signature,data,eta) (contracts/Governance/GovernorAlpha.sol#232)
        Event emitted after the call(s):
        - ProposalQueued(proposalId,eta) (contracts/Governance/GovernorAlpha.sol#224)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
INFO:Detectors:
GovernorAlpha.propose(address[],uint256[],string[],bytes[],string) (contracts/Governance/GovernorAlpha.sol#142-211) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(newProposal.id == 0,GovernorAlpha::propose: ProposalID collision) (contracts/Governance/GovernorAlpha.sol#182)
GovernorAlpha._queueOrRevert(address,uint256,string,bytes,uint256) (contracts/Governance/GovernorAlpha.sol#227-233) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(!timelock.queuedTransactions(keccak256(bytes)(abi.encode(target,value,signature,data,eta))),GovernorAlpha::_queueOrRevert: proposal action already queued at eta) (contracts/Governance/GovernorAlpha.sol#228
-231)
GovernorAlpha.state(uint256) (contracts/Governance/GovernorAlpha.sol#294-314) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= add256(proposal.eta,timelock.GRACE_PERIOD()) (contracts/Governance/GovernorAlpha.sol#309)
GovernorAlpha.add256(uint256,uint256) (contracts/Governance/GovernorAlpha.sol#371-375) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(c >= a,addition overflow) (contracts/Governance/GovernorAlpha.sol#373)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
GovernorAlpha.getChainId() (contracts/Governance/GovernorAlpha.sol#382-388) uses assembly
        - INLINE ASM (contracts/Governance/GovernorAlpha.sol#384-386)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
GovernorAlpha._castVote(address,uint256,bool) (contracts/Governance/GovernorAlpha.sol#331-349) compares to a boolean constant:
        -require(bool,string)(receipt.hasVoted == false,GovernorAlpha::_castVote: voter already voted) (contracts/Governance/GovernorAlpha.sol#335)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Pragma version^0.8.10 (contracts/Governance/GovernorAlpha.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function GovernorAlpha.__acceptAdmin() (contracts/Governance/GovernorAlpha.sol#351-354) is not in mixedCase
Function GovernorAlpha.__abdicate() (contracts/Governance/GovernorAlpha.sol#356-359) is not in mixedCase
Function GovernorAlpha.__queueSetTimelockPendingAdmin(address,uint256) (contracts/Governance/GovernorAlpha.sol#361-364) is not in mixedCase
Function GovernorAlpha.__executeSetTimelockPendingAdmin(address,uint256) (contracts/Governance/GovernorAlpha.sol#366-369) is not in mixedCase
Function TimelockInterface.GRACE_PERIOD() (contracts/Governance/GovernorAlpha.sol#394) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
GovernorAlpha.quorumVotes() (contracts/Governance/GovernorAlpha.sol#9-11) uses literals with too many digits:
        - 400000e18 (contracts/Governance/GovernorAlpha.sol#10)
GovernorAlpha.proposalThreshold() (contracts/Governance/GovernorAlpha.sol#14-16) uses literals with too many digits:
        - 100000e18 (contracts/Governance/GovernorAlpha.sol#15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
Function GovernorAlpha.quorumVotes() (contracts/Governance/GovernorAlpha.sol#9-11) contains magic number: 400000e18
Function GovernorAlpha.proposalThreshold() (contracts/Governance/GovernorAlpha.sol#14-16) contains magic number: 100000e18
Function GovernorAlpha.proposalMaxOperations() (contracts/Governance/GovernorAlpha.sol#19-21) contains magic number: 10
Function GovernorAlpha.votingPeriod() (contracts/Governance/GovernorAlpha.sol#29-31) contains magic number: 17280
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
GovernorAlpha.comp (contracts/Governance/GovernorAlpha.sol#37) should be immutable
GovernorAlpha.timelock (contracts/Governance/GovernorAlpha.sol#34) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

GovernorBravoDelegate.sol

```
INFO:Detectors:
GovernorBravoDelegate.execute(uint256) (contracts/Governance/GovernorBravoDelegate.sol#143-151) sends eth to arbitrary user
        Dangerous calls:
        - timelock.executeTransaction{value: proposal.values[i]}(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/Governance/GovernorBravoDelegate.sol#148)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in GovernorBravoDelegate._initiate(address) (contracts/Governance/GovernorBravoDelegate.sol#371-377):
        External calls:
        - proposalCount = GovernorAlpha(governorAlpha).proposalCount() (contracts/Governance/GovernorBravoDelegate.sol#374)
        State variables written after the call(s):
        - initialProposalId = proposalCount (contracts/Governance/GovernorBravoDelegate.sol#375)
        GovernorBravoDelegateStorageV1.initialProposalId (contracts/Governance/GovernorBravoInterfaces.sol#81) can be used in cross function reentrancies:
        - GovernorBravoDelegate._initiate(address) (contracts/Governance/GovernorBravoDelegate.sol#371-377)
        - GovernorBravoDelegateStorageV1.initialProposalId (contracts/Governance/GovernorBravoInterfaces.sol#81)
        - GovernorBravoDelegate.propose(address[],uint256[],string[],bytes[],string) (contracts/Governance/GovernorBravoDelegate.sol#74-117)
        - GovernorBravoDelegate.state(uint256) (contracts/Governance/GovernorBravoDelegate.sol#209-229)
Reentrancy in GovernorBravoDelegate.queue(uint256) (contracts/Governance/GovernorBravoDelegate.sol#123-132):
        External calls:
        - queueOrRevertInternal(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],eta) (contracts/Governance/GovernorBravoDelegate.sol#128)
                - timelock.queueTransaction(target,value,signature,data,eta) (contracts/Governance/GovernorBravoDelegate.sol#136)
        State variables written after the call(s):
        - proposal.eta = eta (contracts/Governance/GovernorBravoDelegate.sol#130)
        GovernorBravoDelegateStorageV1.proposals (contracts/Governance/GovernorBravoInterfaces.sol#93) can be used in cross function reentrancies:
        - GovernorBravoDelegate.cancel(uint256) (contracts/Governance/GovernorBravoDelegate.sol#157-179)
        - GovernorBravoDelegate.castVoteInternal(address,uint256,uint8) (contracts/Governance/GovernorBravoDelegate.sol#270-291)
        - GovernorBravoDelegate.execute(uint256) (contracts/Governance/GovernorBravoDelegate.sol#143-151)
        - GovernorBravoDelegate.getActions(uint256) (contracts/Governance/GovernorBravoDelegate.sol#189-192)
        - GovernorBravoDelegate.getReceipt(uint256,address) (contracts/Governance/GovernorBravoDelegate.sol#200-202)
        - GovernorBravoDelegateStorageV1.proposals (contracts/Governance/GovernorBravoInterfaces.sol#93)
        - GovernorBravoDelegate.propose(address[],uint256[],string[],bytes[],string) (contracts/Governance/GovernorBravoDelegate.sol#74-117)
        - GovernorBravoDelegate.queue(uint256) (contracts/Governance/GovernorBravoDelegate.sol#123-132)
        - GovernorBravoDelegate.state(uint256) (contracts/Governance/GovernorBravoDelegate.sol#209-229)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
GovernorBravoDelegate.queueOrRevertInternal(address,uint256,string,bytes,uint256) (contracts/Governance/GovernorBravoDelegate.sol#134-137) ignores return value by timelock.queueTransaction(target,value,signature,data,eta) (contracts/Gove
rnance/GovernorBravoDelegate.sol#136)
GovernorBravoDelegate.execute(uint256) (contracts/Governance/GovernorBravoDelegate.sol#143-151) ignores return value by timelock.executeTransaction{value: proposal.values[i]}(proposal.targets[i],proposal.values[i],proposal.signatures[i],
proposal.calldatas[i],proposal.eta) (contracts/Governance/GovernorBravoDelegate.sol#148)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
GovernorBravoDelegate._initiate(address) (contracts/Governance/GovernorBravoDelegate.sol#371-377) should emit an event for:
        - proposalCount = GovernorAlpha(governorAlpha).proposalCount() (contracts/Governance/GovernorBravoDelegate.sol#374)
        - initialProposalId = proposalCount (contracts/Governance/GovernorBravoDelegate.sol#375)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
GovernorBravoDelegate._setWhitelistGuardian(address).account (contracts/Governance/GovernorBravoDelegate.sol#358) lacks a zero-check on :
                - whitelistGuardian = account (contracts/Governance/GovernorBravoDelegate.sol#361)
GovernorBravoDelegate._setPendingAdmin(address).newPendingAdmin (contracts/Governance/GovernorBravoDelegate.sol#384) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (contracts/Governance/GovernorBravoDelegate.sol#392)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
GovernorBravoDelegate.queueOrRevertInternal(address,uint256,string,bytes,uint256) (contracts/Governance/GovernorBravoDelegate.sol#134-137) has external calls inside a loop: require(bool,string)(!timelock.queuedTransactions(keccak256(byt
es)(abi.encode(target,value,signature,data,eta))),GovernorBravo::queueOrRevertInternal: identical proposal action already queued at eta) (contracts/Governance/GovernorBravoDelegate.sol#135)
GovernorBravoDelegate.queueOrRevertInternal(address,uint256,string,bytes,uint256) (contracts/Governance/GovernorBravoDelegate.sol#134-137) has external calls inside a loop: timelock.queueTransaction(target,value,signature,data,eta) (cont
racts/Governance/GovernorBravoDelegate.sol#136)
GovernorBravoDelegate.execute(uint256) (contracts/Governance/GovernorBravoDelegate.sol#143-151) has external calls inside a loop: timelock.executeTransaction{value: proposal.values[i]}(proposal.targets[i],proposal.values[i],proposal.sign
atures[i],proposal.calldatas[i],proposal.eta) (contracts/Governance/GovernorBravoDelegate.sol#148)
GovernorBravoDelegate.cancel(uint256) (contracts/Governance/GovernorBravoDelegate.sol#157-179) has external calls inside a loop: timelock.cancelTransaction(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[
i],proposal.eta) (contracts/Governance/GovernorBravoDelegate.sol#175)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in GovernorBravoDelegate.cancel(uint256) (contracts/Governance/GovernorBravoDelegate.sol#157-179):
        External calls:
        - timelock.cancelTransaction(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/Governance/GovernorBravoDelegate.sol#175)
        Event emitted after the call(s):
        - ProposalCanceled(proposalId) (contracts/Governance/GovernorBravoDelegate.sol#178)
Reentrancy in GovernorBravoDelegate.execute(uint256) (contracts/Governance/GovernorBravoDelegate.sol#143-151):
        External calls:
        - timelock.executeTransaction{value: proposal.values[i]}(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/Governance/GovernorBravoDelegate.sol#148)
        Event emitted after the call(s):
        - ProposalExecuted(proposalId) (contracts/Governance/GovernorBravoDelegate.sol#150)
Reentrancy in GovernorBravoDelegate.queue(uint256) (contracts/Governance/GovernorBravoDelegate.sol#123-132):
        External calls:
        - queueOrRevertInternal(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],eta) (contracts/Governance/GovernorBravoDelegate.sol#128)
                - timelock.queueTransaction(target,value,signature,data,eta) (contracts/Governance/GovernorBravoDelegate.sol#136)
        Event emitted after the call(s):
        - ProposalQueued(proposalId,eta) (contracts/Governance/GovernorBravoDelegate.sol#131)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
GovernorBravoDelegate.propose(address[],uint256[],string[],bytes[],string) (contracts/Governance/GovernorBravoDelegate.sol#74-117) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(comp.getPriorVotes(msg.sender,sub256(block.number,1)) > proposalThreshold || isWhitelisted(msg.sender),GovernorBravo::propose: proposer votes below proposal threshold) (contracts/Governance/GovernorBravoDel
egate.sol#78)
        - require(bool,string)(newProposal.id == 0,GovernorBravo::propose: ProposalID collision) (contracts/Governance/GovernorBravoDelegate.sol#97)
GovernorBravoDelegate.queueOrRevertInternal(address,uint256,string,bytes,uint256) (contracts/Governance/GovernorBravoDelegate.sol#134-137) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(!timelock.queuedTransactions(keccak256(bytes)(abi.encode(target,value,signature,data,eta))),GovernorBravo::queueOrRevertInternal: identical proposal action already queued at eta) (contracts/Governance/Gove
rnorBravoDelegate.sol#135)
GovernorBravoDelegate.state(uint256) (contracts/Governance/GovernorBravoDelegate.sol#209-229) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= add256(proposal.eta,timelock.GRACE_PERIOD()) (contracts/Governance/GovernorBravoDelegate.sol#224)
GovernorBravoDelegate.isWhitelisted(address) (contracts/Governance/GovernorBravoDelegate.sol#298-300) uses timestamp for comparisons
        Dangerous comparisons:
        - (whitelistAccountExpirations[account] > block.timestamp) (contracts/Governance/GovernorBravoDelegate.sol#299)
GovernorBravoDelegate.add256(uint256,uint256) (contracts/Governance/GovernorBravoDelegate.sol#420-424) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(c >= a,addition overflow) (contracts/Governance/GovernorBravoDelegate.sol#422)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
INFO:Detectors:
Function GovernorBravoDelegate.initialize(address,address,uint256,uint256,uint256) (contracts/Governance/GovernorBravoDelegate.sol#49-63) has a dubious typecast: address<=address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
GovernorBravoDelegate.getChainIdInternal() (contracts/Governance/GovernorBravoDelegate.sol#431-435) uses assembly
        - INLINE ASM (contracts/Governance/GovernorBravoDelegate.sol#433)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
GovernorBravoDelegate.castVoteInternal(address,uint256,uint8) (contracts/Governance/GovernorBravoDelegate.sol#270-291) compares to a boolean constant:
        -require(bool,string)(receipt.hasVoted == false,GovernorBravo::castVoteInternal: voter already voted) (contracts/Governance/GovernorBravoDelegate.sol#275)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Pragma version^0.8.10 (contracts/Governance/GovernorBravoDelegate.sol#2) allows old versions
Pragma version^0.8.10 (contracts/Governance/GovernorBravoInterfaces.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function GovernorBravoDelegate._setVotingDelay(uint256) (contracts/Governance/GovernorBravoDelegate.sol#306-313) is not in mixedCase
Function GovernorBravoDelegate._setVotingPeriod(uint256) (contracts/Governance/GovernorBravoDelegate.sol#319-326) is not in mixedCase
Function GovernorBravoDelegate._setProposalThreshold(uint256) (contracts/Governance/GovernorBravoDelegate.sol#333-340) is not in mixedCase
Function GovernorBravoDelegate._setWhitelistAccountExpiration(address,uint256) (contracts/Governance/GovernorBravoDelegate.sol#347-352) is not in mixedCase
Function GovernorBravoDelegate._setWhitelistGuardian(address) (contracts/Governance/GovernorBravoDelegate.sol#358-364) is not in mixedCase
Function GovernorBravoDelegate._initiate(address) (contracts/Governance/GovernorBravoDelegate.sol#371-377) is not in mixedCase
Function GovernorBravoDelegate._setPendingAdmin(address) (contracts/Governance/GovernorBravoDelegate.sol#384-396) is not in mixedCase
Function GovernorBravoDelegate._acceptAdmin() (contracts/Governance/GovernorBravoDelegate.sol#402-418) is not in mixedCase
Function TimelockInterface.GRACE_PERIOD() (contracts/Governance/GovernorBravoInterfaces.sol#181) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable GovernorBravoDelegate.MAX_PROPOSAL_THRESHOLD (contracts/Governance/GovernorBravoDelegate.sol#15) is too similar to GovernorBravoDelegate.MIN_PROPOSAL_THRESHOLD (contracts/Governance/GovernorBravoDelegate.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
GovernorBravoDelegate.slitherConstructorConstantVariables() (contracts/Governance/GovernorBravoDelegate.sol#6-436) uses literals with too many digits:
        - MAX_PROPOSAL_THRESHOLD = 100000e18 (contracts/Governance/GovernorBravoDelegate.sol#15)
GovernorBravoDelegate.slitherConstructorConstantVariables() (contracts/Governance/GovernorBravoDelegate.sol#6-436) uses literals with too many digits:
        - quorumVotes = 400000e18 (contracts/Governance/GovernorBravoDelegate.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
GovernorBravoDelegatorStorage.implementation (contracts/Governance/GovernorBravoInterfaces.sol#59) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
In a function GovernorBravoDelegate._acceptAdmin() (contracts/Governance/GovernorBravoDelegate.sol#402-418) variable GovernorBravoDelegatorStorage.pendingAdmin (contracts/Governance/GovernorBravoInterfaces.sol#56) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

GovernorBravoDelegator.sol

```
INFO:Detectors:
GovernorBravoDelegator.delegateTo(address,bytes) (contracts/Governance/GovernorBravoDelegator.sol#52-59) uses delegatecall to a input-controlled function id
        - (success,returnData) = callee.delegatecall(data) (contracts/Governance/GovernorBravoDelegator.sol#53)
GovernorBravoDelegator.fallback() (contracts/Governance/GovernorBravoDelegator.sol#66-78) uses delegatecall to a input-controlled function id
        - (success) = implementation.delegatecall(msg.data) (contracts/Governance/GovernorBravoDelegator.sol#68)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-delegatecall
INFO:Detectors:
Function GovernorBravoDelegator.delegateTo(address,bytes) (contracts/Governance/GovernorBravoDelegator.sol#52-59) contains a low level call to a custom address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/call_forward_to_protected.md
INFO:Detectors:
GovernorBravoDelegator.constructor(address,address,address,address,uint256,uint256,uint256).admin_ (contracts/Governance/GovernorBravoDelegator.sol#10) lacks a zero-check on :
        - admin = admin_ (contracts/Governance/GovernorBravoDelegator.sol#28)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
GovernorBravoDelegator.delegateTo(address,bytes) (contracts/Governance/GovernorBravoDelegator.sol#52-59) uses assembly
        - INLINE ASM (contracts/Governance/GovernorBravoDelegator.sol#54-58)
GovernorBravoDelegator.fallback() (contracts/Governance/GovernorBravoDelegator.sol#66-78) uses assembly
        - INLINE ASM (contracts/Governance/GovernorBravoDelegator.sol#70-77)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Pragma version^0.8.10 (contracts/Governance/GovernorBravoDelegator.sol#2) allows old versions
Pragma version^0.8.10 (contracts/Governance/GovernorBravoInterfaces.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in GovernorBravoDelegator.delegateTo(address,bytes) (contracts/Governance/GovernorBravoDelegator.sol#52-59):
        - (success,returnData) = callee.delegatecall(data) (contracts/Governance/GovernorBravoDelegator.sol#53)
Low level call in GovernorBravoDelegator.fallback() (contracts/Governance/GovernorBravoDelegator.sol#66-78):
        - (success) = implementation.delegatecall(msg.data) (contracts/Governance/GovernorBravoDelegator.sol#68)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function GovernorBravoDelegator._setImplementation(address) (contracts/Governance/GovernorBravoDelegator.sol#36-44) is not in mixedCase
Function TimelockInterface.GRACE_PERIOD() (contracts/Governance/GovernorBravoInterfaces.sol#181) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
GovernorBravoDelegateStorageV1.comp (contracts/Governance/GovernorBravoInterfaces.sol#90) should be constant
GovernorBravoDelegateStorageV1.initialProposalId (contracts/Governance/GovernorBravoInterfaces.sol#81) should be constant
GovernorBravoDelegateStorageV1.proposalCount (contracts/Governance/GovernorBravoInterfaces.sol#84) should be constant
GovernorBravoDelegateStorageV1.proposalThreshold (contracts/Governance/GovernorBravoInterfaces.sol#78) should be constant
GovernorBravoDelegateStorageV1.timelock (contracts/Governance/GovernorBravoInterfaces.sol#87) should be constant
GovernorBravoDelegateStorageV1.votingDelay (contracts/Governance/GovernorBravoInterfaces.sol#72) should be constant
GovernorBravoDelegateStorageV1.votingPeriod (contracts/Governance/GovernorBravoInterfaces.sol#75) should be constant
GovernorBravoDelegateStorageV2.whitelistGuardian (contracts/Governance/GovernorBravoInterfaces.sol#176) should be constant
GovernorBravoDelegatorStorage.admin (contracts/Governance/GovernorBravoInterfaces.sol#53) should be constant
GovernorBravoDelegatorStorage.implementation (contracts/Governance/GovernorBravoInterfaces.sol#59) should be constant
GovernorBravoDelegatorStorage.pendingAdmin (contracts/Governance/GovernorBravoInterfaces.sol#56) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
GovernorBravoDelegatorStorage.admin (contracts/Governance/GovernorBravoInterfaces.sol#53) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

CompoundLens.sol

```
INFO:Detectors:
GovernorAlpha.execute(uint256) (contracts/Governance/GovernorAlpha.sol#235-252) sends eth to arbitrary user
        Dangerous calls:
        - timelock.executeTransaction{value: proposal.values[i]}(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/Governance/GovernorAlpha.sol#243-249)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Function CompoundLens.setProposal(CompoundLens.GovProposal,GovernorAlpha,uint256) (contracts/Lens/CompoundLens.sol#352-373) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Function CompoundLens.setBravoProposal(CompoundLens.GovBravoProposal,GovernorBravoInterface,uint256) (contracts/Lens/CompoundLens.sol#424-441) is a strange setter. Nothing is set in constructor or set in a function without using function
 parameters
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/strange_setter.md
INFO:Detectors:
Function CErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CErc20.sol#26-41) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initialize.md
INFO:Detectors:
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transfer(address,uint256) (contracts/EIP20NonStandardInterface.sol#35)
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transferFrom(address,address,uint256) (contracts/EIP20NonStandardInterface.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
INFO:Detectors:
CToken.accrueInterest() (contracts/CToken.sol#325-381) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/CToken.sol#331)
CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/CToken.sol#293)
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CToken.sol#27-60) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/CToken.sol#36)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == NO_ERROR,LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/CToken.sol#811)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) uses a dangerous strict equality:
        - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
Comp._writeCheckpoint(address,uint32,uint96,uint96) (contracts/Governance/Comp.sol#270-281) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (contracts/Governance/Comp.sol#273)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745):
        External calls:
        - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#737)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
                - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
        CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalReserves = totalReservesNew (contracts/CToken.sol#883)
```

```
        CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82)
Reentrancy in GovernorAlpha.queue(uint256) (contracts/Governance/GovernorAlpha.sol#213-225):
        External calls:
        - _queueOrRevert(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],eta) (contracts/Governance/GovernorAlpha.sol#221)
                - timelock.queueTransaction(target,value,signature,data,eta) (contracts/Governance/GovernorAlpha.sol#232)
        State variables written after the call(s):
        - proposal.eta = eta (contracts/Governance/GovernorAlpha.sol#223)
        GovernorAlpha.proposals (contracts/Governance/GovernorAlpha.sol#99) can be used in cross function reentrancies:
        - GovernorAlpha._castVote(address,uint256,bool) (contracts/Governance/GovernorAlpha.sol#331-349)
        - GovernorAlpha.cancel(uint256) (contracts/Governance/GovernorAlpha.sol#254-277)
        - GovernorAlpha.execute(uint256) (contracts/Governance/GovernorAlpha.sol#235-252)
        - GovernorAlpha.getActions(uint256) (contracts/Governance/GovernorAlpha.sol#279-288)
        - GovernorAlpha.getReceipt(uint256,address) (contracts/Governance/GovernorAlpha.sol#290-292)
        - GovernorAlpha.proposals (contracts/Governance/GovernorAlpha.sol#99)
        - GovernorAlpha.propose(address[],uint256[],string[],bytes[],string) (contracts/Governance/GovernorAlpha.sol#142-211)
        - GovernorAlpha.queue(uint256) (contracts/Governance/GovernorAlpha.sol#213-225)
        - GovernorAlpha.state(uint256) (contracts/Governance/GovernorAlpha.sol#294-314)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - totalSupply = totalSupply - redeemTokens (contracts/CToken.sol#548)
        CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87) can be used in cross function reentrancies:
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
CToken._addReservesFresh(uint256).actualAddAmount (contracts/CToken.sol#1055) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
CErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CErc20.sol#26-41) ignores return value by EIP20Interface(underlying).totalSupply() (contracts/CErc20.sol#40)
GovernorAlpha._queueOrRevert(address,uint256,string,bytes,uint256) (contracts/Governance/GovernorAlpha.sol#227-233) ignores return value by timelock.queueTransaction(target,value,signature,data,eta) (contracts/Governance/GovernorAlpha.sol#232)
GovernorAlpha.execute(uint256) (contracts/Governance/GovernorAlpha.sol#235-252) ignores return value by timelock.executeTransaction{value: proposal.values[i]}(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/Governance/GovernorAlpha.sol#243-249)
GovernorAlpha.cancel(uint256) (contracts/Governance/GovernorAlpha.sol#254-277) ignores return value by timelock.cancelTransaction(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/Governance/GovernorAlpha.sol#267-273)
GovernorAlpha.__queueSetTimelockPendingAdmin(address,uint256) (contracts/Governance/GovernorAlpha.sol#361-364) ignores return value by timelock.queueTransaction(address(timelock),0,setPendingAdmin(address),abi.encode(newPendingAdmin),eta) (contracts/Governance/GovernorAlpha.sol#363)
GovernorAlpha.__executeSetTimelockPendingAdmin(address,uint256) (contracts/Governance/GovernorAlpha.sol#366-369) ignores return value by timelock.executeTransaction(address(timelock),0,setPendingAdmin(address),abi.encode(newPendingAdmin),eta) (contracts/Governance/GovernorAlpha.sol#368)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) shadows:
        - CTokenInterfaceNewReserveGuardian(address,address) (contracts/CTokenInterfaces.sol#195) (event)
GovernorAlpha.cancel(uint256).state (contracts/Governance/GovernorAlpha.sol#255) shadows:
        - GovernorAlpha.state(uint256) (contracts/Governance/GovernorAlpha.sol#294-314) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
CToken._setPendingAdmin(address).newPendingAdmin (contracts/CToken.sol#902) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (contracts/CToken.sol#912)
CErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8).underlying_ (contracts/CErc20.sol#27) lacks a zero-check on :
                - underlying = underlying_ (contracts/CErc20.sol#39)
GovernorAlpha.constructor(address,address,address).guardian_ (contracts/Governance/GovernorAlpha.sol#136) lacks a zero-check on :
                - guardian = guardian_ (contracts/Governance/GovernorAlpha.sol#139)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
GovernorAlpha._queueOrRevert(address,uint256,string,bytes,uint256) (contracts/Governance/GovernorAlpha.sol#227-233) has external calls inside a loop: require(bool,string)(! timelock.queuedTransactions(keccak256(bytes)(abi.encode(target,value,signature,data,eta))),GovernorAlpha::_queueOrRevert: proposal action already queued at eta) (contracts/Governance/GovernorAlpha.sol#228-231)
GovernorAlpha._queueOrRevert(address,uint256,string,bytes,uint256) (contracts/Governance/GovernorAlpha.sol#227-233) has external calls inside a loop: timelock.queueTransaction(target,value,signature,data,eta) (contracts/Governance/GovernorAlpha.sol#232)
GovernorAlpha.execute(uint256) (contracts/Governance/GovernorAlpha.sol#235-252) has external calls inside a loop: timelock.executeTransaction{value: proposal.values[i]}(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/Governance/GovernorAlpha.sol#243-249)
GovernorAlpha.cancel(uint256) (contracts/Governance/GovernorAlpha.sol#254-277) has external calls inside a loop: timelock.cancelTransaction(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/Governance/GovernorAlpha.sol#267-273)
CompoundLens.cTokenMetadata(CToken) (contracts/Lens/CompoundLens.sol#145-186) has external calls inside a loop: exchangeRateCurrent = cToken.exchangeRateCurrent() (contracts/Lens/CompoundLens.sol#146)
CompoundLens.cTokenMetadata(CToken) (contracts/Lens/CompoundLens.sol#145-186) has external calls inside a loop: comptroller = ComptrollerLensInterface(address(cToken.comptroller())) (contracts/Lens/CompoundLens.sol#147)
CompoundLens.cTokenMetadata(CToken) (contracts/Lens/CompoundLens.sol#145-186) has external calls inside a loop: (isListed,collateralFactorMantissa) = comptroller.markets(address(cToken)) (contracts/Lens/CompoundLens.sol#148)
CompoundLens.cTokenMetadata(CToken) (contracts/Lens/CompoundLens.sol#145-186) has external calls inside a loop: compareStrings(cToken.symbol(),1ETH) (contracts/Lens/CompoundLens.sol#152)
CompoundLens.getCompSpeeds(ComptrollerLensInterface,CToken) (contracts/Lens/CompoundLens.sol#91-121) has external calls inside a loop: (compSupplySpeedSuccess,compSupplySpeedReturnData) = address(comptroller).call(abi.encodePacked(comptroller.compSupplySpeeds.selector,abi.encode(address(cToken)))) (contracts/Lens/CompoundLens.sol#96-98)
CompoundLens.getCompSpeeds(ComptrollerLensInterface,CToken) (contracts/Lens/CompoundLens.sol#91-121) has external calls inside a loop: (compBorrowSpeedSuccess,compBorrowSpeedReturnData) = address(comptroller).call(abi.encodePacked(comptroller.compBorrowSpeeds.selector,abi.encode(address(cToken)))) (contracts/Lens/CompoundLens.sol#104-106)
CompoundLens.getCompSpeeds(ComptrollerLensInterface,CToken) (contracts/Lens/CompoundLens.sol#91-121) has external calls inside a loop: (compSpeedSuccess,compSpeedReturnData) = address(comptroller).call(abi.encodePacked(comptroller.compSpeeds.selector,abi.encode(address(cToken)))) (contracts/Lens/CompoundLens.sol#113-115)
CompoundLens.getMarketCaps(ComptrollerLensInterface,CToken) (contracts/Lens/CompoundLens.sol#123-143) has external calls inside a loop: (borrowCapSuccess,borrowCapReturnData) = address(comptroller).call(abi.encodePacked(comptroller.borrowCaps.selector,address(cToken))) (contracts/Lens/CompoundLens.sol#128-130)
CompoundLens.cTokenMetadata(CToken) (contracts/Lens/CompoundLens.sol#145-186) has external calls inside a loop: CTokenMetadata(address(cToken),exchangeRateCurrent,cToken.supplyRatePerBlock(),cToken.borrowRatePerBlock(),cToken.reserveFactorMantissa(),cToken.totalBorrows(),cToken.totalReserves(),cToken.totalSupply(),cToken.getCash(),isListed,collateralFactorMantissa,underlyingAssetAddress,cToken.decimals(),underlyingDecimals,compSupplySpeed,compBorrowSpeed,borrowCap) (contracts/Lens/CompoundLens.sol#165-185)
CompoundLens.cTokenMetadata(CToken) (contracts/Lens/CompoundLens.sol#145-186) has external calls inside a loop: underlyingAssetAddress = cErc20.underlying() (contracts/Lens/CompoundLens.sol#157)
CompoundLens.cTokenMetadata(CToken) (contracts/Lens/CompoundLens.sol#145-186) has external calls inside a loop: underlyingDecimals = EIP20Interface(cErc20.underlying()).decimals() (contracts/Lens/CompoundLens.sol#158)
CompoundLens.cTokenBalances(CToken,address) (contracts/Lens/CompoundLens.sol#206-232) has external calls inside a loop: balanceOf = cToken.balanceOf(account) (contracts/Lens/CompoundLens.sol#207)
CompoundLens.cTokenBalances(CToken,address) (contracts/Lens/CompoundLens.sol#206-232) has external calls inside a loop: borrowBalanceCurrent = cToken.borrowBalanceCurrent(account) (contracts/Lens/CompoundLens.sol#208)
CompoundLens.cTokenBalances(CToken,address) (contracts/Lens/CompoundLens.sol#206-232) has external calls inside a loop: balanceOfUnderlying = cToken.balanceOfUnderlying(account) (contracts/Lens/CompoundLens.sol#209)
CompoundLens.cTokenBalances(CToken,address) (contracts/Lens/CompoundLens.sol#206-232) has external calls inside a loop: compareStrings(cToken.symbol(),1ETH) (contracts/Lens/CompoundLens.sol#213)
CompoundLens.cTokenBalances(CToken,address) (contracts/Lens/CompoundLens.sol#206-232) has external calls inside a loop: underlying = EIP20Interface(cErc20.underlying()) (contracts/Lens/CompoundLens.sol#218)
CompoundLens.cTokenBalances(CToken,address) (contracts/Lens/CompoundLens.sol#206-232) has external calls inside a loop: tokenBalance = underlying.balanceOf(account) (contracts/Lens/CompoundLens.sol#219)
CompoundLens.cTokenBalances(CToken,address) (contracts/Lens/CompoundLens.sol#206-232) has external calls inside a loop: tokenAllowance = underlying.allowance(account,address(cToken)) (contracts/Lens/CompoundLens.sol#220)
CompoundLens.cTokenUnderlyingPrice(CToken) (contracts/Lens/CompoundLens.sol#251-257) has external calls inside a loop: comptroller = ComptrollerLensInterface(address(cToken.comptroller())) (contracts/Lens/CompoundLens.sol#252)
CompoundLens.cTokenUnderlyingPrice(CToken) (contracts/Lens/CompoundLens.sol#251-257) has external calls inside a loop: priceOracle = comptroller.oracle() (contracts/Lens/CompoundLens.sol#253)
CompoundLens.cTokenUnderlyingPrice(CToken) (contracts/Lens/CompoundLens.sol#251-257) has external calls inside a loop: CTokenUnderlyingPrice(address(cToken),priceOracle.getUnderlyingPrice(cToken)) (contracts/Lens/CompoundLens.sol#255-256)
CompoundLens.getGovReceipts(GovernorAlpha,address,uint256[]) (contracts/Lens/CompoundLens.sol#291-308) has external calls inside a loop: receipt = governor.getReceipt(proposalIds[i],voter) (contracts/Lens/CompoundLens.sol#299)
CompoundLens.getGovBravoReceipts(GovernorBravoInterface,address,uint256[]) (contracts/Lens/CompoundLens.sol#317-334) has external calls inside a loop: receipt = governor.getReceipt(proposalIds[i],voter) (contracts/Lens/CompoundLens.sol#325)
CompoundLens.getGovProposals(GovernorAlpha,uint256[]) (contracts/Lens/CompoundLens.sol#375-405) has external calls inside a loop: (targets,values,signatures,calldatas) = governor.getActions(proposalIds[i]) (contracts/Lens/CompoundLens.sol#381-386)
CompoundLens.setProposal(CompoundLens.GovProposal,GovernorAlpha,uint256) (contracts/Lens/CompoundLens.sol#352-373) has external calls inside a loop: (proposer,eta,startBlock,endBlock,forVotes,againstVotes,canceled,executed) = governor.proposals(proposalId) (contracts/Lens/CompoundLens.sol#353-363)
CompoundLens.getGovBravoProposals(GovernorBravoInterface,uint256[]) (contracts/Lens/CompoundLens.sol#443-474) has external calls inside a loop: (targets,values,signatures,calldatas) = governor.getActions(proposalIds[i]) (contracts/Lens/CompoundLens.sol#449-454)
CompoundLens.setBravoProposal(CompoundLens.GovBravoProposal,GovernorBravoInterface,uint256) (contracts/Lens/CompoundLens.sol#424-441) has external calls inside a loop: p = governor.proposals(proposalId) (contracts/Lens/CompoundLens.sol#429)
CompoundLens.getCompVotes(Comp,address,uint32[]) (contracts/Lens/CompoundLens.sol#526-539) has external calls inside a loop: res[i] = CompVotes(uint256(blockNumbers[i]),uint256(comp.getPriorVotes(account,blockNumbers[i]))) (contracts/Lens/CompoundLens.sol#533-536)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#629)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#630)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#631)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        State variables written after the call(s):
        - accountTokens[minter] = accountTokens[minter] + mintTokens (contracts/CToken.sol#442)
        - totalSupply = totalSupply + mintTokens (contracts/CToken.sol#441)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - accountTokens[redeemer] = accountTokens[redeemer] - redeemTokens (contracts/CToken.sol#549)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#713)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#714)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
Reentrancy in CToken.seizeInternal(address,address,address,uint256) (contracts/CToken.sol#855-892):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        State variables written after the call(s):
        - accountTokens[borrower] = accountTokens[borrower] - seizeTokens (contracts/CToken.sol#885)
        - accountTokens[liquidator] = accountTokens[liquidator] + liquidatorSeizeTokens (contracts/CToken.sol#886)
        - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        - totalSupply = totalSupply - protocolSeizeTokens (contracts/CToken.sol#884)
Reentrancy in CToken.transferTokens(address,address,address,uint256) (contracts/CToken.sol#471-115):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/CToken.sol#473)
        State variables written after the call(s):
        - accountTokens[src] = srcTokensNew (contracts/CToken.sol#100)
        - accountTokens[dst] = dstTokensNew (contracts/CToken.sol#101)
        - transferAllowances[src][spender] = allowanceNew (contracts/CToken.sol#105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        Event emitted after the call(s):
        - Borrow(borrower,borrowAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#642)
Reentrancy in GovernorAlpha.cancel(uint256) (contracts/Governance/GovernorAlpha.sol#254-277):
        External calls:
        - timelock.cancelTransaction(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/Governance/GovernorAlpha.sol#267-273)
        Event emitted after the call(s):
        - ProposalCanceled(proposalId) (contracts/Governance/GovernorAlpha.sol#276)
Reentrancy in GovernorAlpha.execute(uint256) (contracts/Governance/GovernorAlpha.sol#235-252):
        External calls:
        - timelock.executeTransaction{value: proposal.values[i]}(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/Governance/GovernorAlpha.sol#243-249)
        Event emitted after the call(s):
        - ProposalExecuted(proposalId) (contracts/Governance/GovernorAlpha.sol#251)
Reentrancy in CompoundLens.getCompBalanceMetadataExt(Comp,ComptrollerLensInterface,address) (contracts/Lens/CompoundLens.sol#498-519):
        External calls:
        - comptroller.claimComp(account) (contracts/Lens/CompoundLens.sol#504)
        Event emitted after the call(s):
        - balanceMetaData(balance,allocated) (contracts/Lens/CompoundLens.sol#510)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
                - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        Event emitted after the call(s):
        - ReservesAdded(address(this),protocolSeizeAmount,totalReservesNew) (contracts/CToken.sol#891)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,liquidator,liquidatorSeizeTokens) (contracts/CToken.sol#889)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
        - Transfer(borrower,address(this),protocolSeizeTokens) (contracts/CToken.sol#890)
                - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        - seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#818)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(cTokenCollateral),seizeTokens) (contracts/CToken.sol#824)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        Event emitted after the call(s):
        - Mint(minter,actualMintAmount,mintTokens) (contracts/CToken.sol#445)
        - Transfer(address(this),minter,mintTokens) (contracts/CToken.sol#446)
Reentrancy in GovernorAlpha.queue(uint256) (contracts/Governance/GovernorAlpha.sol#213-225):
        External calls:
        - _queueOrRevert(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],eta) (contracts/Governance/GovernorAlpha.sol#221)
                - timelock.queueTransaction(target,value,signature,data,eta) (contracts/Governance/GovernorAlpha.sol#232)
        Event emitted after the call(s):
        - ProposalQueued(proposalId,eta) (contracts/Governance/GovernorAlpha.sol#224)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        Event emitted after the call(s):
        - Redeem(redeemer,redeemAmount,redeemTokens) (contracts/CToken.sol#561)
        - Transfer(redeemer,address(this),redeemTokens) (contracts/CToken.sol#560)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Comp.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (contracts/Governance/Comp.sol#167-178) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,Comp::delegateBySig: signature expired) (contracts/Governance/Comp.sol#176)
GovernorAlpha.propose(address[],uint256[],string[],bytes[],string) (contracts/Governance/GovernorAlpha.sol#142-211) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(newProposal.id == 0,GovernorAlpha::propose: ProposalID collision) (contracts/Governance/GovernorAlpha.sol#182)
GovernorAlpha._queueOrRevert(address,uint256,string,bytes,uint256) (contracts/Governance/GovernorAlpha.sol#227-233) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(!timelock.queuedTransactions(keccak256(bytes)(abi.encode(target,value,signature,data,eta))),GovernorAlpha::_queueOrRevert: proposal action already queued at eta) (contracts/Governance/GovernorAlpha.sol#228
-231)
GovernorAlpha.state(uint256) (contracts/Governance/GovernorAlpha.sol#294-314) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= add256(proposal.eta,timelock.GRACE_PERIOD()) (contracts/Governance/GovernorAlpha.sol#309)
GovernorAlpha.add256(uint256,uint256) (contracts/Governance/GovernorAlpha.sol#371-375) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(c >= a,addition overflow) (contracts/Governance/GovernorAlpha.sol#373)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Function CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) has a dubious typecast: address<=address
Function CToken._acceptAdmin() (contracts/CToken.sol#925-945) has a dubious typecast: address<=address
Function CErc20.doTransferIn(address,uint256) (contracts/CErc20.sol#190-219) has a dubious typecast: address<=EIP20Interface
Function ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) has a dubious typecast: uint256<=uint256
Function ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) has a dubious typecast: uint256<=uint256
Function CompoundLens.cTokenMetadata(CToken) (contracts/Lens/CompoundLens.sol#145-186) has a dubious typecast: ComptrollerLensInterface<=address
Function CompoundLens.cTokenMetadata(CToken) (contracts/Lens/CompoundLens.sol#145-186) has a dubious typecast: CErc20<=address
Function CompoundLens.cTokenBalances(CToken,address) (contracts/Lens/CompoundLens.sol#206-232) has a dubious typecast: CErc20<=address
Function CompoundLens.cTokenUnderlyingPrice(CToken) (contracts/Lens/CompoundLens.sol#251-257) has a dubious typecast: ComptrollerLensInterface<=address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
Setter function CompoundLens.setProposal(CompoundLens.GovProposal,GovernorAlpha,uint256) (contracts/Lens/CompoundLens.sol#352-373) does not emit an event
Setter function CompoundLens.setBravoProposal(CompoundLens.GovBravoProposal,GovernorBravoInterface,uint256) (contracts/Lens/CompoundLens.sol#424-441) does not emit an event
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/event_setter.md
INFO:Detectors:
CErc20.doTransferIn(address,uint256) (contracts/CErc20.sol#190-219) uses assembly
        - INLINE ASM (contracts/CErc20.sol#198-213)
CErc20.doTransferOut(address,uint256) (contracts/CErc20.sol#230-252) uses assembly
        - INLINE ASM (contracts/CErc20.sol#235-250)
Comp.getChainId() (contracts/Governance/Comp.sol#304-310) uses assembly
        - INLINE ASM (contracts/Governance/Comp.sol#306-308)
GovernorAlpha.getChainId() (contracts/Governance/GovernorAlpha.sol#382-388) uses assembly
        - INLINE ASM (contracts/Governance/GovernorAlpha.sol#384-386)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
GovernorAlpha._castVote(address,uint256,bool) (contracts/Governance/GovernorAlpha.sol#331-349) compares to a boolean constant:
        -require(bool,string)(receipt.hasVoted == false,GovernorAlpha::_castVote: voter already voted) (contracts/Governance/GovernorAlpha.sol#335)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) is never used and should be removed
ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#92-94) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#88-90) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#152-154) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#156-158) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#140-142) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#144-146) is never used and should be removed
ExponentialNoError.div_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#160-162) is never used and should be removed
ExponentialNoError.fraction(uint256,uint256) (contracts/ExponentialNoError.sol#168-170) is never used and should be removed
ExponentialNoError.greaterThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#67-69) is never used and should be removed
ExponentialNoError.isZeroExp(ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#74-76) is never used and should be removed
ExponentialNoError.lessThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#53-55) is never used and should be removed
ExponentialNoError.lessThanOrEqualExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#60-62) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#124-126) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#128-130) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#112-114) is never used and should be removed
ExponentialNoError.mul_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#132-134) is never used and should be removed
ExponentialNoError.safe224(uint256,string) (contracts/ExponentialNoError.sol#78-81) is never used and should be removed
ExponentialNoError.safe32(uint256,string) (contracts/ExponentialNoError.sol#83-86) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#104-106) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#100-102) is never used and should be removed
ExponentialNoError.sub_(uint256,uint256) (contracts/ExponentialNoError.sol#108-110) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
INFO:Detectors:
Pragma version^0.8.10 (contracts/CErc20.sol#2) allows old versions
Pragma version^0.8.10 (contracts/CToken.sol#2) allows old versions
Pragma version^0.8.10 (contracts/CTokenInterfaces.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ComptrollerInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20Interface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20NonStandardInterface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ErrorReporter.sol#2) allows old versions
Pragma version^0.8.10 (contracts/ExponentialNoError.sol#2) allows old versions
Pragma version^0.8.10 (contracts/Governance/Comp.sol#2) allows old versions
Pragma version^0.8.10 (contracts/Governance/GovernorAlpha.sol#2) allows old versions
Pragma version^0.8.10 (contracts/InterestRateModel.sol#2) allows old versions
Pragma version^0.8.10 (contracts/Lens/CompoundLens.sol#2) allows old versions
Pragma version^0.8.10 (contracts/PriceOracle.sol#2) allows old versions
Pragma version0.8.10 (contracts/Whitelist.sol#3) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in CompoundLens.getCompSpeeds(ComptrollerLensInterface,CToken) (contracts/Lens/CompoundLens.sol#91-121):
        - (compSupplySpeedSuccess,compSupplySpeedReturnData) = address(comptroller).call(abi.encodePacked(comptroller.compSupplySpeeds.selector,abi.encode(address(cToken)))) (contracts/Lens/CompoundLens.sol#96-98)
        - (compBorrowSpeedSuccess,compBorrowSpeedReturnData) = address(comptroller).call(abi.encodePacked(comptroller.compBorrowSpeeds.selector,abi.encode(address(cToken)))) (contracts/Lens/CompoundLens.sol#104-106)
        - (compSpeedSuccess,compSpeedReturnData) = address(comptroller).call(abi.encodePacked(comptroller.compSpeeds.selector,abi.encode(address(cToken)))) (contracts/Lens/CompoundLens.sol#113-115)
Low level call in CompoundLens.getMarketCaps(ComptrollerLensInterface,CToken) (contracts/Lens/CompoundLens.sol#123-143):
        - (borrowCapSuccess,borrowCapReturnData) = address(comptroller).call(abi.encodePacked(comptroller.borrowCaps.selector,address(cToken))) (contracts/Lens/CompoundLens.sol#128-130)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Comp (contracts/Governance/Comp.sol#4-311) should inherit from CompLike (contracts/CErc20.sol#6-8)
Comp (contracts/Governance/Comp.sol#4-311) should inherit from ComInterface (contracts/Governance/GovernorAlpha.sol#425-427)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance
INFO:Detectors:
Function CErc20._addReserves(uint256) (contracts/CErc20.sol#165-167) is not in mixedCase
Function CErc20._delegateCompLikeTo(address) (contracts/CErc20.sol#259-262) is not in mixedCase
Function CToken._setPendingAdmin(address) (contracts/CToken.sol#902-918) is not in mixedCase
Function CToken._acceptAdmin() (contracts/CToken.sol#925-945) is not in mixedCase
Function CToken._setComptroller(ComptrollerInterface) (contracts/CToken.sol#952-969) is not in mixedCase
Function CToken._setReserveGuardian(address) (contracts/CToken.sol#976-979) is not in mixedCase
Function CToken._setReserveFactor(uint256) (contracts/CToken.sol#998-1002) is not in mixedCase
Function CToken._reduceReserves(uint256) (contracts/CToken.sol#1093-1097) is not in mixedCase
Function CToken._setInterestRateModel(InterestRateModel) (contracts/CToken.sol#1154-1158) is not in mixedCase
Variable CTokenStorage._notEntered (contracts/CTokenInterfaces.sol#13) is not in mixedCase
Constant CTokenStorage.borrowRateMaxMantissa (contracts/CTokenInterfaces.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant CTokenStorage.reserveFactorMaxMantissa (contracts/CTokenInterfaces.sol#34) is not in UPPER_CASE_WITH_UNDERSCORES
Function CTokenInterface._setPendingAdmin(address) (contracts/CTokenInterfaces.sol#260) is not in mixedCase
Function CTokenInterface._acceptAdmin() (contracts/CTokenInterfaces.sol#262) is not in mixedCase
Function CTokenInterface._setComptroller(ComptrollerInterface) (contracts/CTokenInterfaces.sol#264) is not in mixedCase
Function CTokenInterface._setReserveGuardian(address) (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Parameter CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Function CTokenInterface._setReserveFactor(uint256) (contracts/CTokenInterfaces.sol#268) is not in mixedCase
Function CTokenInterface._reduceReserves(uint256) (contracts/CTokenInterfaces.sol#270) is not in mixedCase
Function CTokenInterface._setInterestRateModel(InterestRateModel) (contracts/CTokenInterfaces.sol#272) is not in mixedCase
Function CErc20Interface._addReserves(uint256) (contracts/CTokenInterfaces.sol#311) is not in mixedCase
Function CDelegatorInterface._setImplementation(address,bool,bytes) (contracts/CTokenInterfaces.sol#333-337) is not in mixedCase
Function CDelegateInterface._becomeImplementation(bytes) (contracts/CTokenInterfaces.sol#346) is not in mixedCase
Function CDelegateInterface._resignImplementation() (contracts/CTokenInterfaces.sol#351) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncate(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#37-40) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncateAddUInt(ExponentialNoError.Exp,uint256,uint256) (contracts/ExponentialNoError.sol#45-48) is not in mixedCase
Constant ExponentialNoError.expScale (contracts/ExponentialNoError.sol#12) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.doubleScale (contracts/ExponentialNoError.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.mantissaOne (contracts/ExponentialNoError.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Function GovernorAlpha.__acceptAdmin() (contracts/Governance/GovernorAlpha.sol#351-354) is not in mixedCase
Function GovernorAlpha.__abdicate() (contracts/Governance/GovernorAlpha.sol#356-359) is not in mixedCase
Function GovernorAlpha.__queueSetTimelockPendingAdmin(address,uint256) (contracts/Governance/GovernorAlpha.sol#361-364) is not in mixedCase
Function GovernorAlpha.__executeSetTimelockPendingAdmin(address,uint256) (contracts/Governance/GovernorAlpha.sol#366-369) is not in mixedCase
Function TimelockInterface.GRACE_PERIOD() (contracts/Governance/GovernorAlpha.sol#394) is not in mixedCase
Event CompoundLenslodeMetaData(uint256,uint256) (contracts/Lens/CompoundLens.sol#89) is not in CapWords
Parameter Whitelist.updateWhitelist(address,bool)._address (contracts/Whitelist.sol#11) is not in mixedCase
Parameter Whitelist.updateWhitelist(address,bool)._isActive (contracts/Whitelist.sol#12) is not in mixedCase
Parameter Whitelist.getWhitelisted(address)._address (contracts/Whitelist.sol#17) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface).seizeTokens (contracts/CToken.sol#806) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.seizeInternal(address,address,address,uint256).seizeTokens (contracts/CToken.sol#859) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.seizeInternal(address,address,address,uint256).seizeTokens (contracts/CToken.sol#855) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CTokenInterface.seize(address,address,uint256).seizeTokens (contracts/CTokenInterfaces.sol#256) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
Comp.slitherConstructorConstantVariables() (contracts/Governance/Comp.sol#4-311) uses literals with too many digits:
        - totalSupply = 20000000e18 (contracts/Governance/Comp.sol#15)
GovernorAlpha.quorumVotes() (contracts/Governance/GovernorAlpha.sol#9-11) uses literals with too many digits:
        - 400000e18 (contracts/Governance/GovernorAlpha.sol#10)
GovernorAlpha.proposalThreshold() (contracts/Governance/GovernorAlpha.sol#14-16) uses literals with too many digits:
        - 100000e18 (contracts/Governance/GovernorAlpha.sol#15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is never used in CErc20 (contracts/CErc20.sol#15-263)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
Function CErc20.doTransferIn(address,uint256) (contracts/CErc20.sol#190-219) contains magic number: 32
Function CErc20.doTransferOut(address,uint256) (contracts/CErc20.sol#230-252) contains magic number: 32
Function Comp.approve(address,uint256) (contracts/Governance/Comp.sol#88-100) contains magic number: 96
Function Comp.transfer(address,uint256) (contracts/Governance/Comp.sol#117-121) contains magic number: 96
Function Comp.transferFrom(address,address,uint256) (contracts/Governance/Comp.sol#130-148) contains magic number: 96
Function Comp._writeCheckpoint(address,uint32,uint96,uint96) (contracts/Governance/Comp.sol#270-281) contains magic number: 32
Function GovernorAlpha.quorumVotes() (contracts/Governance/GovernorAlpha.sol#9-11) contains magic number: 400000e18
Function GovernorAlpha.proposalThreshold() (contracts/Governance/GovernorAlpha.sol#14-16) contains magic number: 100000e18
Function GovernorAlpha.proposalMaxOperations() (contracts/Governance/GovernorAlpha.sol#19-21) contains magic number: 10
Function GovernorAlpha.votingPeriod() (contracts/Governance/GovernorAlpha.sol#29-31) contains magic number: 17280
Function CompoundLens.cTokenMetadata(CToken) (contracts/Lens/CompoundLens.sol#145-186) contains magic number: 18
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
CDelegationStorage.implementation (contracts/CTokenInterfaces.sol#318) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
GovernorAlpha.comp (contracts/Governance/GovernorAlpha.sol#37) should be immutable
GovernorAlpha.timelock (contracts/Governance/GovernorAlpha.sol#34) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Detectors:
In a function Comp.getPriorVotes(address,uint256) (contracts/Governance/Comp.sol#197-229) variable Comp.checkpoints (contracts/Governance/Comp.sol#33) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

Denominations.sol

```
INFO:Detectors:
Pragma version^0.8.10 (contracts/Oracle/Denominations.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

PriceOracleProxyETH.sol

```
INFO:Detectors:
Function CErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CErc20.sol#26-41) is an unprotected initializer.
Function IUniswapV2Pair.initialize(address,address) (contracts/Oracle/Interfaces/UniswapV2Interface.sol#96) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initialize.md
INFO:Detectors:
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transfer(address,uint256) (contracts/EIP20NonStandardInterface.sol#35)
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#9-71) has incorrect ERC20 function interface:EIP20NonStandardInterface.transferFrom(address,address,uint256) (contracts/EIP20NonStandardInterface.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
```

```
INFO:Detectors:
CToken.accrueInterest() (contracts/CToken.sol#325-381) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/CToken.sol#331)
CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/CToken.sol#293)
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CToken.sol#27-60) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/CToken.sol#36)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == NO_ERROR,LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/CToken.sol#811)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825) uses a dangerous strict equality:
        - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#730-745):
        External calls:
        - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#737)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
                - require(bool,string)(cTokenCollateral.seize(liquidator,borrower,seizeTokens) == NO_ERROR,token seizure failed) (contracts/CToken.sol#820)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
        CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalBorrows (contracts/CTokenInterfaces.sol#77)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#744)
                - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82) can be used in cross function reentrancies:
        - CToken.accrueInterest() (contracts/CToken.sol#325-381)
        - CToken.borrowRatePerBlock() (contracts/CToken.sol#205-207)
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CToken.supplyRatePerBlock() (contracts/CToken.sol#213-215)
        - CTokenStorage.totalReserves (contracts/CTokenInterfaces.sol#82)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - totalSupply = totalSupply - redeemTokens (contracts/CToken.sol#548)
        CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87) can be used in cross function reentrancies:
        - CToken.exchangeRateStoredInternal() (contracts/CToken.sol#291-310)
        - CTokenStorage.totalSupply (contracts/CTokenInterfaces.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
PriceOracleProxyETH.getSequencerStatus(address).status (contracts/Oracle/PriceOracleProxyETH.sol#168) is a local variable never initialized
CToken._addReservesFresh(uint256).actualAddAmount (contracts/CToken.sol#1055) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
CErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/CErc20.sol#26-41) ignores return value by EIP20Interface(underlying).totalSupply() (contracts/CErc20.sol#40)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) shadows:
        - CTokenInterfaceNewReserveGuardian(address,address) (contracts/CTokenInterfaces.sol#195) (event)
Exponential.divScalarByExpTruncate(uint256,ExponentialNoError.Exp).fraction (contracts/Exponential.sol#124) shadows:
        - ExponentialNoError.fraction(uint256,uint256) (contracts/ExponentialNoError.sol#168-170) (function)
SushiOracle.price().price (contracts/Oracle/SushiOracle.sol#38) shadows:
        - SushiOracle.price() (contracts/Oracle/SushiOracle.sol#35-40) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
CToken._setPendingAdmin(address).newPendingAdmin (contracts/CToken.sol#902) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (contracts/CToken.sol#912)
CErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8).underlying_ (contracts/CErc20.sol#27) lacks a zero-check on :
                - underlying = underlying_ (contracts/CErc20.sol#39)
PriceOracleProxyETH.constructor(address,address,address,address,address,address,address).admin_ (contracts/Oracle/PriceOracleProxyETH.sol#75) lacks a zero-check on :
                - admin = admin_ (contracts/Oracle/PriceOracleProxyETH.sol#83)
PriceOracleProxyETH.constructor(address,address,address,address,address,address,address).sequencerAddress_ (contracts/Oracle/PriceOracleProxyETH.sol#77) lacks a zero-check on :
                - sequencerAddress = sequencerAddress_ (contracts/Oracle/PriceOracleProxyETH.sol#85)
PriceOracleProxyETH.constructor(address,address,address,address,address,address,address).letherAddress_ (contracts/Oracle/PriceOracleProxyETH.sol#78) lacks a zero-check on :
                - letherAddress = letherAddress_ (contracts/Oracle/PriceOracleProxyETH.sol#86)
PriceOracleProxyETH.constructor(address,address,address,address,address,address,address).lplvGLPAddress_ (contracts/Oracle/PriceOracleProxyETH.sol#79) lacks a zero-check on :
                - lplvGLPAddress = lplvGLPAddress_ (contracts/Oracle/PriceOracleProxyETH.sol#87)
PriceOracleProxyETH.constructor(address,address,address,address,address,address,address).glpOracleAddress_ (contracts/Oracle/PriceOracleProxyETH.sol#80) lacks a zero-check on :
                - glpOracleAddress = glpOracleAddress_ (contracts/Oracle/PriceOracleProxyETH.sol#88)
PriceOracleProxyETH.constructor(address,address,address,address,address,address,address).lodeOracle_ (contracts/Oracle/PriceOracleProxyETH.sol#81) lacks a zero-check on :
                - lodeOracle = lodeOracle_ (contracts/Oracle/PriceOracleProxyETH.sol#89)
PriceOracleProxyETH._setGuardian(address)._guardian (contracts/Oracle/PriceOracleProxyETH.sol#200) lacks a zero-check on :
                - guardian = _guardian (contracts/Oracle/PriceOracleProxyETH.sol#202)
PriceOracleProxyETH._setAdmin(address)._admin (contracts/Oracle/PriceOracleProxyETH.sol#210) lacks a zero-check on :
                - admin = _admin (contracts/Oracle/PriceOracleProxyETH.sol#212)
SushiOracle.constructor(address,address,address).tokenA_ (contracts/Oracle/SushiOracle.sol#23) lacks a zero-check on :
                - tokenA = tokenA_ (contracts/Oracle/SushiOracle.sol#25)
SushiOracle.constructor(address,address,address).tokenB_ (contracts/Oracle/SushiOracle.sol#23) lacks a zero-check on :
                - tokenB = tokenB_ (contracts/Oracle/SushiOracle.sol#26)
SushiOracle.constructor(address,address,address).poolContract_ (contracts/Oracle/SushiOracle.sol#23) lacks a zero-check on :
                - poolContract = poolContract_ (contracts/Oracle/SushiOracle.sol#27)
SushiOracle._setPoolContract(address).newPoolContract (contracts/Oracle/SushiOracle.sol#44) lacks a zero-check on :
                - poolContract = newPoolContract (contracts/Oracle/SushiOracle.sol#46)
SushiOracle._setAdmin(address).newAdmin (contracts/Oracle/SushiOracle.sol#50) lacks a zero-check on :
                - admin = newAdmin (contracts/Oracle/SushiOracle.sol#52)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#629)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#630)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#631)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#400-451):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#402)
        State variables written after the call(s):
        - accountTokens[minter] = accountTokens[minter] + mintTokens (contracts/CToken.sol#442)
        - totalSupply = totalSupply + mintTokens (contracts/CToken.sol#441)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#497-565):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,redeemTokens) (contracts/CToken.sol#525)
        State variables written after the call(s):
        - accountTokens[redeemer] = accountTokens[redeemer] - redeemTokens (contracts/CToken.sol#549)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#673-721):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = accountBorrowsNew (contracts/CToken.sol#713)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#714)
        - totalBorrows = totalBorrowsNew (contracts/CToken.sol#715)
Reentrancy in CToken.seizeInternal(address,address,address,uint256) (contracts/CToken.sol#855-892):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#857)
        State variables written after the call(s):
        - accountTokens[borrower] = accountTokens[borrower] - seizeTokens (contracts/CToken.sol#885)
        - accountTokens[liquidator] = accountTokens[liquidator] + liquidatorSeizeTokens (contracts/CToken.sol#886)
        - totalReserves = totalReservesNew (contracts/CToken.sol#883)
        - totalSupply = totalSupply - protocolSeizeTokens (contracts/CToken.sol#884)
Reentrancy in CToken.transferTokens(address,address,address,uint256) (contracts/CToken.sol#71-115):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/CToken.sol#73)
        State variables written after the call(s):
        - accountTokens[src] = srcTokensNew (contracts/CToken.sol#100)
        - accountTokens[dst] = dstTokensNew (contracts/CToken.sol#101)
        - transferAllowances[src][spender] = allowanceNew (contracts/CToken.sol#105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#595-643):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#597)
        Event emitted after the call(s):
        - Borrow(borrower,borrowAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#642)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
        Event emitted after the call(s):
        - RepayBorrow(payer,borrower,actualRepayAmount,accountBorrowsNew,totalBorrowsNew,borrowIndex) (contracts/CToken.sol#718)
                - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#755-825):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#762-768)
        - actualRepayAmount = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#799)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#675)
```

```
Parameter CTokenInterface._setReserveGuardian(address).NewReserveGuardian (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Function CTokenInterface._setReserveFactor(uint256) (contracts/CTokenInterfaces.sol#268) is not in mixedCase
Function CTokenInterface._reduceReserves(uint256) (contracts/CTokenInterfaces.sol#270) is not in mixedCase
Function CTokenInterface._setInterestRateModel(InterestRateModel) (contracts/CTokenInterfaces.sol#272) is not in mixedCase
Function CErc20Interface._addReserves(uint256) (contracts/CTokenInterfaces.sol#311) is not in mixedCase
Function CDelegatorInterface._setImplementation(address,bool,bytes) (contracts/CTokenInterfaces.sol#333-337) is not in mixedCase
Function CDelegateInterface._becomeImplementation(bytes) (contracts/CTokenInterfaces.sol#346) is not in mixedCase
Function CDelegateInterface._resignImplementation() (contracts/CTokenInterfaces.sol#351) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncate(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#37-40) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncateAddUInt(ExponentialNoError.Exp,uint256,uint256) (contracts/ExponentialNoError.sol#45-48) is not in mixedCase
Constant ExponentialNoError.expScale (contracts/ExponentialNoError.sol#12) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.doubleScale (contracts/ExponentialNoError.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.mantissaOne (contracts/ExponentialNoError.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (contracts/Oracle/Interfaces/UniswapV2Interface.sol#30) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (contracts/Oracle/Interfaces/UniswapV2Interface.sol#32) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (contracts/Oracle/Interfaces/UniswapV2Interface.sol#58) is not in mixedCase
Event PriceOracleProxyETHnewLodeOracle(address) (contracts/Oracle/PriceOracleProxyETH.sol#193) is not in CapWords
Event PriceOracleProxyETHnewGlpOracle(address) (contracts/Oracle/PriceOracleProxyETH.sol#194) is not in CapWords
Function PriceOracleProxyETH._setGuardian(address) (contracts/Oracle/PriceOracleProxyETH.sol#200-204) is not in mixedCase
Parameter PriceOracleProxyETH._setGuardian(address)._guardian (contracts/Oracle/PriceOracleProxyETH.sol#200) is not in mixedCase
Function PriceOracleProxyETH._setAdmin(address) (contracts/Oracle/PriceOracleProxyETH.sol#210-214) is not in mixedCase
Parameter PriceOracleProxyETH._setAdmin(address)._admin (contracts/Oracle/PriceOracleProxyETH.sol#210) is not in mixedCase
Function PriceOracleProxyETH._setLodeOracle(SushiOracleInterface) (contracts/Oracle/PriceOracleProxyETH.sol#220-225) is not in mixedCase
Parameter PriceOracleProxyETH._setLodeOracle(SushiOracleInterface)._newLodeOracle (contracts/Oracle/PriceOracleProxyETH.sol#220) is not in mixedCase
Function PriceOracleProxyETH._setGlpOracle(PlvGLPOracleInterface) (contracts/Oracle/PriceOracleProxyETH.sol#231-236) is not in mixedCase
Parameter PriceOracleProxyETH._setGlpOracle(PlvGLPOracleInterface)._newGlpOracle (contracts/Oracle/PriceOracleProxyETH.sol#231) is not in mixedCase
Function PriceOracleProxyETH._setAggregators(address[],address[],PriceOracleProxyETH.AggregatorBase[]) (contracts/Oracle/PriceOracleProxyETH.sol#244-261) is not in mixedCase
Event SushiOraclepoolContractUpdated(address) (contracts/Oracle/SushiOracle.sol#19) is not in CapWords
Event SushiOracleadminUpdated(address) (contracts/Oracle/SushiOracle.sol#21) is not in CapWords
Function SushiOracle._setPoolContract(address) (contracts/Oracle/SushiOracle.sol#44-48) is not in mixedCase
Function SushiOracle._setAdmin(address) (contracts/Oracle/SushiOracle.sol#50-54) is not in mixedCase
Parameter Whitelist.updateWhitelist(address,bool)._address (contracts/Whitelist.sol#11) is not in mixedCase
Parameter Whitelist.updateWhitelist(address,bool)._isActive (contracts/Whitelist.sol#12) is not in mixedCase
Parameter Whitelist.getWhitelisted(address)._address (contracts/Whitelist.sol#17) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable CToken.seize(address,address,uint256).seizeTokens (contracts/CToken.sol#839) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface).seizeTokens (contracts/CToken.sol#806) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CToken.seizeInternal(address,address,address,uint256).seizeTokens (contracts/CToken.sol#855) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable CTokenInterface.seize(address,address,uint256).seizeTokens (contracts/CTokenInterfaces.sol#256) is too similar to CToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/CToken.sol#855)
Variable PriceOracleProxyETH.getUnderlyingPrice(CToken).price_scope_0 (contracts/Oracle/PriceOracleProxyETH.sol#104) is too similar to PriceOracleProxyETH.getUnderlyingPrice(CToken).price_scope_1 (contracts/Oracle/PriceOracleProxyETH.sol
#109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is never used in CErc20 (contracts/CErc20.sol#15-263)
ExponentialNoError.mantissaOne (contracts/ExponentialNoError.sol#15) is never used in PriceOracleProxyETH (contracts/Oracle/PriceOracleProxyETH.sol#16-262)
PriceOracleProxyETH.lodePrice (contracts/Oracle/PriceOracleProxyETH.sol#50) is never used in PriceOracleProxyETH (contracts/Oracle/PriceOracleProxyETH.sol#16-262)
ExponentialNoError.mantissaOne (contracts/ExponentialNoError.sol#15) is never used in SushiOracle (contracts/Oracle/SushiOracle.sol#8-56)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
Function CErc20.doTransferIn(address,uint256) (contracts/CErc20.sol#190-219) contains magic number: 32
Function CErc20.doTransferOut(address,uint256) (contracts/CErc20.sol#230-252) contains magic number: 32
Function PriceOracleProxyETH.getUnderlyingPrice(CToken) (contracts/Oracle/PriceOracleProxyETH.sol#97-124) contains magic numbers: 10, 10
Function PriceOracleProxyETH.getPriceFromChainlink(AggregatorV3Interface) (contracts/Oracle/PriceOracleProxyETH.sol#133-141) contains magic number: 10
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
CDelegationStorage.implementation (contracts/CTokenInterfaces.sol#318) should be constant
PriceOracleProxyETH.lodePrice (contracts/Oracle/PriceOracleProxyETH.sol#50) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
PriceOracleProxyETH.ethUsdAggregator (contracts/Oracle/PriceOracleProxyETH.sol#63) should be immutable
PriceOracleProxyETH.letherAddress (contracts/Oracle/PriceOracleProxyETH.sol#36) should be immutable
PriceOracleProxyETH.lplpGLPAddress (contracts/Oracle/PriceOracleProxyETH.sol#39) should be immutable
PriceOracleProxyETH.sequencerAddress (contracts/Oracle/PriceOracleProxyETH.sol#45) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

SushiOracle.sol

```
INFO:Detectors:
Function IUniswapV2Pair.initialize(address,address) (contracts/Oracle/Interfaces/UniswapV2Interface.sol#96) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initialize.md
INFO:Detectors:
Exponential.divScalarByExpTruncate(uint256,ExponentialNoError.Exp).fraction (contracts/Exponential.sol#124) shadows:
        - ExponentialNoError.fraction(uint256,uint256) (contracts/ExponentialNoError.sol#168-170) (function)
SushiOracle.price().price (contracts/Oracle/SushiOracle.sol#38) shadows:
        - SushiOracle.price() (contracts/Oracle/SushiOracle.sol#35-40) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
SushiOracle.constructor(address,address,address).tokenA_ (contracts/Oracle/SushiOracle.sol#23) lacks a zero-check on :
        - tokenA = tokenA_ (contracts/Oracle/SushiOracle.sol#25)
SushiOracle.constructor(address,address,address).tokenB_ (contracts/Oracle/SushiOracle.sol#23) lacks a zero-check on :
        - tokenB = tokenB_ (contracts/Oracle/SushiOracle.sol#26)
SushiOracle.constructor(address,address,address).poolContract_ (contracts/Oracle/SushiOracle.sol#23) lacks a zero-check on :
        - poolContract = poolContract_ (contracts/Oracle/SushiOracle.sol#27)
SushiOracle._setPoolContract(address).newPoolContract (contracts/Oracle/SushiOracle.sol#44) lacks a zero-check on :
        - poolContract = newPoolContract (contracts/Oracle/SushiOracle.sol#46)
SushiOracle._setAdmin(address).newAdmin (contracts/Oracle/SushiOracle.sol#50) lacks a zero-check on :
        - admin = newAdmin (contracts/Oracle/SushiOracle.sol#52)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
CarefulMath.addThenSubUInt(uint256,uint256,uint256) (contracts/CarefulMath.sol#76-84) is never used and should be removed
CarefulMath.addUInt(uint256,uint256) (contracts/CarefulMath.sol#63-71) is never used and should be removed
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) is never used and should be removed
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) is never used and should be removed
CarefulMath.subUInt(uint256,uint256) (contracts/CarefulMath.sol#52-58) is never used and should be removed
Exponential.addExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/Exponential.sol#37-41) is never used and should be removed
Exponential.divExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/Exponential.sol#180-182) is never used and should be removed
Exponential.divScalar(ExponentialNoError.Exp,uint256) (contracts/Exponential.sol#91-98) is never used and should be removed
Exponential.divScalarByExp(uint256,ExponentialNoError.Exp) (contracts/Exponential.sol#103-118) is never used and should be removed
Exponential.divScalarByExpTruncate(uint256,ExponentialNoError.Exp) (contracts/Exponential.sol#123-130) is never used and should be removed
Exponential.getExp(uint256,uint256) (contracts/Exponential.sol#20-32) is never used and should be removed
Exponential.mulExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/Exponential.sol#135-155) is never used and should be removed
Exponential.mulExp(uint256,uint256) (contracts/Exponential.sol#160-162) is never used and should be removed
Exponential.mulExp3(ExponentialNoError.Exp,ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/Exponential.sol#167-173) is never used and should be removed
Exponential.mulScalar(ExponentialNoError.Exp,uint256) (contracts/Exponential.sol#55-62) is never used and should be removed
Exponential.mulScalarTruncate(ExponentialNoError.Exp,uint256) (contracts/Exponential.sol#67-74) is never used and should be removed
Exponential.mulScalarTruncateAddUInt(ExponentialNoError.Exp,uint256,uint256) (contracts/Exponential.sol#79-86) is never used and should be removed
Exponential.subExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/Exponential.sol#46-50) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#92-94) is never used and should be removed
ExponentialNoError.add_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#88-90) is never used and should be removed
ExponentialNoError.add_(uint256,uint256) (contracts/ExponentialNoError.sol#98-100) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#152-154) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#156-158) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#140-142) is never used and should be removed
ExponentialNoError.div_(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#144-146) is never used and should be removed
ExponentialNoError.div_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#160-162) is never used and should be removed
ExponentialNoError.div_(uint256,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#148-150) is never used and should be removed
ExponentialNoError.div_(uint256,uint256) (contracts/ExponentialNoError.sol#164-166) is never used and should be removed
ExponentialNoError.fraction(uint256,uint256) (contracts/ExponentialNoError.sol#168-170) is never used and should be removed
ExponentialNoError.greaterThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#67-69) is never used and should be removed
ExponentialNoError.isZeroExp(ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#74-76) is never used and should be removed
ExponentialNoError.lessThanExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#53-55) is never used and should be removed
ExponentialNoError.lessThanOrEqualExp(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#60-62) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#124-126) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Double,uint256) (contracts/ExponentialNoError.sol#128-130) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#112-114) is never used and should be removed
ExponentialNoError.mul_(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#116-118) is never used and should be removed
ExponentialNoError.mul_(uint256,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#132-134) is never used and should be removed
ExponentialNoError.mul_(uint256,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#120-122) is never used and should be removed
ExponentialNoError.mul_(uint256,uint256) (contracts/ExponentialNoError.sol#136-138) is never used and should be removed
ExponentialNoError.mul_ScalarTruncate(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#37-40) is never used and should be removed
ExponentialNoError.mul_ScalarTruncateAddUInt(ExponentialNoError.Exp,uint256,uint256) (contracts/ExponentialNoError.sol#45-48) is never used and should be removed
ExponentialNoError.safe224(uint256,string) (contracts/ExponentialNoError.sol#78-81) is never used and should be removed
ExponentialNoError.safe32(uint256,string) (contracts/ExponentialNoError.sol#83-86) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Double,ExponentialNoError.Double) (contracts/ExponentialNoError.sol#104-106) is never used and should be removed
ExponentialNoError.sub_(ExponentialNoError.Exp,ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#100-102) is never used and should be removed
ExponentialNoError.sub_(uint256,uint256) (contracts/ExponentialNoError.sol#108-110) is never used and should be removed
ExponentialNoError.truncate(ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#29-32) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.10 (contracts/CarefulMath.sol#2) allows old versions
Pragma version^0.8.10 (contracts/EIP20Interface.sol#2) allows old versions
Pragma version^0.8.10 (contracts/Exponential.sol#1) allows old versions
Pragma version^0.8.10 (contracts/ExponentialNoError.sol#2) allows old versions
Pragma version^0.8.10 (contracts/Oracle/Interfaces/UniswapV2Interface.sol#1) allows old versions
Pragma version^0.8.10 (contracts/Oracle/SushiOracle.sol#2) allows old versions
solc-0.8.10 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function ExponentialNoError.mul_ScalarTruncate(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#37-40) is not in mixedCase
Function ExponentialNoError.mul_ScalarTruncateAddUInt(ExponentialNoError.Exp,uint256,uint256) (contracts/ExponentialNoError.sol#45-48) is not in mixedCase
```

```
Constant ExponentialNoError.expScale (contracts/ExponentialNoError.sol#12) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.doubleScale (contracts/ExponentialNoError.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.mantissaOne (contracts/ExponentialNoError.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (contracts/Oracle/Interfaces/UniswapV2Interface.sol#30) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (contracts/Oracle/Interfaces/UniswapV2Interface.sol#32) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (contracts/Oracle/Interfaces/UniswapV2Interface.sol#58) is not in mixedCase
Event SushiOraclepoolContractUpdated(address) (contracts/Oracle/SushiOracle.sol#19) is not in CapWords
Event SushiOracleadminUpdated(address) (contracts/Oracle/SushiOracle.sol#21) is not in CapWords
Function SushiOracle._setPoolContract(address) (contracts/Oracle/SushiOracle.sol#44-48) is not in mixedCase
Function SushiOracle._setAdmin(address) (contracts/Oracle/SushiOracle.sol#50-54) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
ExponentialNoError.mantissaOne (contracts/ExponentialNoError.sol#15) is never used in SushiOracle (contracts/Oracle/SushiOracle.sol#8-56)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
```

plvGLPOracle.sol

```
INFO:Detectors:
PlvGLPOracle.constructor(address,address,address,address,uint256)._GLP (contracts/plvglp-oracle/plvGLPOracle.sol#42) lacks a zero-check on :
                - GLP = _GLP (contracts/plvglp-oracle/plvGLPOracle.sol#43)
PlvGLPOracle.constructor(address,address,address,address,uint256)._GLPManager (contracts/plvglp-oracle/plvGLPOracle.sol#42) lacks a zero-check on :
                - GLPManager = _GLPManager (contracts/plvglp-oracle/plvGLPOracle.sol#44)
PlvGLPOracle.constructor(address,address,address,address,uint256)._plvGLP (contracts/plvglp-oracle/plvGLPOracle.sol#42) lacks a zero-check on :
                - plvGLP = _plvGLP (contracts/plvglp-oracle/plvGLPOracle.sol#45)
PlvGLPOracle.constructor(address,address,address,address,uint256)._whitelist (contracts/plvglp-oracle/plvGLPOracle.sol#42) lacks a zero-check on :
                - whitelist = _whitelist (contracts/plvglp-oracle/plvGLPOracle.sol#46)
PlvGLPOracle._updateGlpAddress(address)._newGlpAddress (contracts/plvglp-oracle/plvGLPOracle.sol#181) lacks a zero-check on :
                - GLP = _newGlpAddress (contracts/plvglp-oracle/plvGLPOracle.sol#183)
PlvGLPOracle._updateGlpManagerAddress(address)._newGlpManagerAddress (contracts/plvglp-oracle/plvGLPOracle.sol#191) lacks a zero-check on :
                - GLPManager = _newGlpManagerAddress (contracts/plvglp-oracle/plvGLPOracle.sol#193)
PlvGLPOracle._updatePlvGlpAddress(address)._newPlvGlpAddress (contracts/plvglp-oracle/plvGLPOracle.sol#201) lacks a zero-check on :
                - plvGLP = _newPlvGlpAddress (contracts/plvglp-oracle/plvGLPOracle.sol#203)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
PlvGLPOracle.computeAverageIndex() (contracts/plvglp-oracle/plvGLPOracle.sol#88-105) uses timestamp for comparisons
        Dangerous comparisons:
        - latestIndexing <= windowSize (contracts/plvglp-oracle/plvGLPOracle.sol#91)
        - i < latestIndexing (contracts/plvglp-oracle/plvGLPOracle.sol#92)
        - i_scope_0 <= latestIndexing (contracts/plvglp-oracle/plvGLPOracle.sol#99)
PlvGLPOracle.checkSwing(uint256) (contracts/plvglp-oracle/plvGLPOracle.sol#122-132) uses timestamp for comparisons
        Dangerous comparisons:
        - currentIndex > maxSwing || currentIndex < minSwing (contracts/plvglp-oracle/plvGLPOracle.sol#127)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Pragma version^0.8.10 (contracts/plvglp-oracle/Interfaces/ERC20Interface.sol#2) allows old versions
Pragma version^0.8.17 (contracts/plvglp-oracle/Interfaces/GLPManagerInterface.sol#2) allows old versions
Pragma version^0.8.17 (contracts/plvglp-oracle/Interfaces/plvGLPInterface.sol#2) allows old versions
Pragma version0.8.17 (contracts/plvglp-oracle/Whitelist.sol#2) allows old versions
Pragma version0.8.17 (contracts/plvglp-oracle/plvGLPOracle.sol#2) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Contract plvGLPInterface (contracts/plvglp-oracle/Interfaces/plvGLPInterface.sol#4-6) is not in CapWords
Parameter Whitelist.updateWhitelist(address,bool)._address (contracts/plvglp-oracle/Whitelist.sol#9) is not in mixedCase
Parameter Whitelist.updateWhitelist(address,bool)._isActive (contracts/plvglp-oracle/Whitelist.sol#9) is not in mixedCase
Parameter Whitelist.getWhitelisted(address)._address (contracts/plvglp-oracle/Whitelist.sol#13) is not in mixedCase
Event PlvGLPOracleupdatePosted(uint256,uint256) (contracts/plvglp-oracle/plvGLPOracle.sol#48) is not in CapWords
Event PlvGLPOraclenewGLPAddress(address,address) (contracts/plvglp-oracle/plvGLPOracle.sol#171) is not in CapWords
Event PlvGLPOraclenewGLPManagerAddress(address,address) (contracts/plvglp-oracle/plvGLPOracle.sol#172) is not in CapWords
Event PlvGLPOraclenewPlvGLPAddress(address,address) (contracts/plvglp-oracle/plvGLPOracle.sol#173) is not in CapWords
Event PlvGLPOracleneweWindowSize(uint256,uint256) (contracts/plvglp-oracle/plvGLPOracle.sol#174) is not in CapWords
Event PlvGLPOracleneweMaxSwing(uint256,uint256) (contracts/plvglp-oracle/plvGLPOracle.sol#175) is not in CapWords
Function PlvGLPOracle._updateGlpAddress(address) (contracts/plvglp-oracle/plvGLPOracle.sol#181-185) is not in mixedCase
Parameter PlvGLPOracle._updateGlpAddress(address)._newGlpAddress (contracts/plvglp-oracle/plvGLPOracle.sol#181) is not in mixedCase
Function PlvGLPOracle._updateGlpManagerAddress(address) (contracts/plvglp-oracle/plvGLPOracle.sol#191-195) is not in mixedCase
Parameter PlvGLPOracle._updateGlpManagerAddress(address)._newGlpManagerAddress (contracts/plvglp-oracle/plvGLPOracle.sol#191) is not in mixedCase
Function PlvGLPOracle._updatePlvGlpAddress(address) (contracts/plvglp-oracle/plvGLPOracle.sol#201-205) is not in mixedCase
Parameter PlvGLPOracle._updatePlvGlpAddress(address)._newPlvGlpAddress (contracts/plvglp-oracle/plvGLPOracle.sol#201) is not in mixedCase
Function PlvGLPOracle._updateWindowSize(uint256) (contracts/plvglp-oracle/plvGLPOracle.sol#211-215) is not in mixedCase
Parameter PlvGLPOracle._updateWindowSize(uint256)._newWindowSize (contracts/plvglp-oracle/plvGLPOracle.sol#211) is not in mixedCase
Function PlvGLPOracle._updateMaxSwing(uint256) (contracts/plvglp-oracle/plvGLPOracle.sol#217-221) is not in mixedCase
Parameter PlvGLPOracle._updateMaxSwing(uint256)._newMaxSwing (contracts/plvglp-oracle/plvGLPOracle.sol#217) is not in mixedCase
Variable PlvGLPOracle.GLP (contracts/plvglp-oracle/plvGLPOracle.sol#21) is not in mixedCase
Variable PlvGLPOracle.GLPManager (contracts/plvglp-oracle/plvGLPOracle.sol#22) is not in mixedCase
Variable PlvGLPOracle.MAX_SWING (contracts/plvglp-oracle/plvGLPOracle.sol#25) is not in mixedCase
Variable PlvGLPOracle.HistoricalIndices (contracts/plvglp-oracle/plvGLPOracle.sol#37) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Function PlvGLPOracle.constructor(address,address,address,address,uint256) (contracts/plvglp-oracle/plvGLPOracle.sol#42-53) contains magic number: 1e16
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
PlvGLPOracle.whitelist (contracts/plvglp-oracle/plvGLPOracle.sol#24) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Detectors:
In a function PlvGLPOracle.computeAverageIndex() (contracts/plvglp-oracle/plvGLPOracle.sol#88-105) variable PlvGLPOracle.HistoricalIndices (contracts/plvglp-oracle/plvGLPOracle.sol#37) is read multiple times
In a function PlvGLPOracle.computeAverageIndex() (contracts/plvglp-oracle/plvGLPOracle.sol#88-105) variable PlvGLPOracle.windowSize (contracts/plvglp-oracle/plvGLPOracle.sol#19) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

Ploopy.sol

```
INFO:Detectors:
Ploopy.receiveFlashLoan(IERC20[],uint256[],uint256[],bytes) (contracts/looper/Ploopy.sol#164-218) uses arbitrary from in transferFrom: data.tokenToLoop.safeTransferFrom(data.user,msg.sender,data.borrowedAmount) (contracts/looper/Ploopy.sol#217)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#arbitrary-from-in-transferfrom
INFO:Detectors:
Ploopy.receiveFlashLoan(IERC20[],uint256[],uint256[],bytes) (contracts/looper/Ploopy.sol#164-218) ignores return value by lTokenMapping[data.tokenToLoop].transfer(data.user,lTokenMapping[data.tokenToLoop].balanceOf(address(this))) (contracts/looper/Ploopy.sol#207)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
Ploopy.loop(IERC20,uint256,uint16,uint16) (contracts/looper/Ploopy.sol#95-161) performs a multiplication on the result of a division:
        - __computedAmount = (_amount * (_tokenPriceInEth / _usdcPriceInEth)) (contracts/looper/Ploopy.sol#121)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Ploopy.receiveFlashLoan(IERC20[],uint256[],uint256[],bytes) (contracts/looper/Ploopy.sol#164-218) uses a dangerous strict equality:
        - require(bool,string)(_finalBal >= 0,lToken balance not 0 at the end of loop) (contracts/looper/Ploopy.sol#211)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) ignores return value by USDC.approve(address(REWARD_ROUTER_V2),type()(uint256).max) (contracts/looper/Ploopy.sol#49)
Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) ignores return value by USDC.approve(address(GLP),type()(uint256).max) (contracts/looper/Ploopy.sol#50)
Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) ignores return value by USDC.approve(address(GLP_MANAGER),type()(uint256).max) (contracts/looper/Ploopy.sol#51)
Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) ignores return value by sGLP.approve(address(GLP_DEPOSITOR),type()(uint256).max) (contracts/looper/Ploopy.sol#53)
Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) ignores return value by GLP.approve(address(GLP_DEPOSITOR),type()(uint256).max) (contracts/looper/Ploopy.sol#54)
Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) ignores return value by sGLP.approve(address(REWARD_ROUTER_V2),type()(uint256).max) (contracts/looper/Ploopy.sol#55)
Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) ignores return value by GLP.approve(address(REWARD_ROUTER_V2),type()(uint256).max) (contracts/looper/Ploopy.sol#56)
Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) ignores return value by USDC.approve(address(VAULT),type()(uint256).max) (contracts/looper/Ploopy.sol#58)
Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) ignores return value by USDT.approve(address(VAULT),type()(uint256).max) (contracts/looper/Ploopy.sol#59)
Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) ignores return value by WBTC.approve(address(VAULT),type()(uint256).max) (contracts/looper/Ploopy.sol#60)
Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) ignores return value by DAI.approve(address(VAULT),type()(uint256).max) (contracts/looper/Ploopy.sol#61)
Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) ignores return value by FRAX.approve(address(VAULT),type()(uint256).max) (contracts/looper/Ploopy.sol#62)
Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) ignores return value by ARB.approve(address(VAULT),type()(uint256).max) (contracts/looper/Ploopy.sol#63)
Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) ignores return value by PLVGLP.approve(address(1PLVGLP),type()(uint256).max) (contracts/looper/Ploopy.sol#65)
Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) ignores return value by USDC.approve(address(1USDC),type()(uint256).max) (contracts/looper/Ploopy.sol#66)
Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) ignores return value by USDT.approve(address(1USDT),type()(uint256).max) (contracts/looper/Ploopy.sol#67)
Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) ignores return value by WBTC.approve(address(1WBTC),type()(uint256).max) (contracts/looper/Ploopy.sol#68)
Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) ignores return value by DAI.approve(address(1DAI),type()(uint256).max) (contracts/looper/Ploopy.sol#69)
Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) ignores return value by FRAX.approve(address(1FRAX),type()(uint256).max) (contracts/looper/Ploopy.sol#70)
Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) ignores return value by ARB.approve(address(1ARB),type()(uint256).max) (contracts/looper/Ploopy.sol#71)
Ploopy.receiveFlashLoan(IERC20[],uint256[],uint256[],bytes) (contracts/looper/Ploopy.sol#164-218) ignores return value by lTokenMapping[data.tokenToLoop].mint(data.tokenToLoop.balanceOf(address(this))) (contracts/looper/Ploopy.sol#206)
Ploopy.receiveFlashLoan(IERC20[],uint256[],uint256[],bytes) (contracts/looper/Ploopy.sol#164-218) ignores return value by lTokenMapping[data.tokenToLoop].borrowBehalf(data.borrowedAmount,data.user) (contracts/looper/Ploopy.sol#215)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Function Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#64-69) contains a low level call to a custom address
Function Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#128-137) contains a low level call to a custom address
Function Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#155-162) contains a low level call to a custom address
Function Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#180-187) contains a low level call to a custom address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/call_forward_to_protected.md
```

AUTOMATED TESTING

```
INFO:Detectors:
Reentrancy in Ploopy.loop(IERC20,uint256,uint16,uint16) (contracts/looper/Ploopy.sol#95-161):
        External calls:
        - _token.safeTransferFrom(msg.sender,address(this),_amount) (contracts/looper/Ploopy.sol#105)
        Event emitted after the call(s):
        - Allowance(_tokenToBorrow.allowance(msg.sender,address(this)),loanAmount) (contracts/looper/Ploopy.sol#143)
        - BalanceOf(_tokenToBorrow.balanceOf(address(BALANCER_VAULT)),loanAmount) (contracts/looper/Ploopy.sol#138)
        - Loan(loanAmount) (contracts/looper/Ploopy.sol#137)
        - Transfer(msg.sender,address(this),_amount) (contracts/looper/Ploopy.sol#106)
        - UserDataEvent(msg.sender,_amount,address(_tokenToBorrow),loanAmount,address(_token)) (contracts/looper/Ploopy.sol#158)
Reentrancy in Ploopy.receiveFlashLoan(IERC20[],uint256[],uint256[],bytes) (contracts/looper/Ploopy.sol#164-218):
        External calls:
        - glpAmount = REWARD_ROUTER_V2.mintAndStakeGlp(address(data.borrowedToken),data.borrowedAmount,0,0) (contracts/looper/Ploopy.sol#183-188)
        - GLP_DEPOSITOR.deposit(glpAmount) (contracts/looper/Ploopy.sol#194)
        Event emitted after the call(s):
        - plvGLPBalance(_newPlvglpBal) (contracts/looper/Ploopy.sol#198)
Reentrancy in Ploopy.receiveFlashLoan(IERC20[],uint256[],uint256[],bytes) (contracts/looper/Ploopy.sol#164-218):
        External calls:
        - glpAmount = REWARD_ROUTER_V2.mintAndStakeGlp(address(data.borrowedToken),data.borrowedAmount,0,0) (contracts/looper/Ploopy.sol#183-188)
        - GLP_DEPOSITOR.deposit(glpAmount) (contracts/looper/Ploopy.sol#194)
        - lTokenMapping[data.tokenToLoop].mint(data.tokenToLoop.balanceOf(address(this))) (contracts/looper/Ploopy.sol#206)
        - lTokenMapping[data.tokenToLoop].transfer(data.user,lTokenMapping[data.tokenToLoop].balanceOf(address(this))) (contracts/looper/Ploopy.sol#207)
        Event emitted after the call(s):
        - lTokenBalance(_finalBal) (contracts/looper/Ploopy.sol#210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Function Ploopy.loop(IERC20,uint256,uint16,uint16) (contracts/looper/Ploopy.sol#95-161) has a dubious typecast: address<=address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
Address._revert(bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#231-243) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#236-239)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity are used:
        - Version used: ['0.8.17', '^0.8.0', '^0.8.1']
        - 0.8.17 (contracts/looper/Ploopy.sol#2)
        - 0.8.17 (contracts/looper/PloopyConstants.sol#2)
        - 0.8.17 (contracts/looper/interfaces/IFlashLoanRecipient.sol#15)
        - 0.8.17 (contracts/looper/interfaces/IVault.sol#15)
        - 0.8.17 (contracts/looper/interfaces/Interfaces.sol#2)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Permit.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#89-91) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#118-120) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#170-172) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#180-187) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#145-147) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#155-162) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#64-69) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#219-229) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
ReentrancyGuard._reentrancyGuardEntered() (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#74-76) is never used and should be removed
SafeERC20._callOptionalReturnBool(IERC20,bytes) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#134-142) is never used and should be removed
SafeERC20.forceApprove(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#82-89) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#45-54) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#69-75) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#60-63) is never used and should be removed
SafeERC20.safePermit(IERC20Permit,address,address,uint256,uint256,uint8,bytes32,bytes32) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#95-109) is never used and should be removed
SafeERC20.safeTransfer(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#26-28) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.17 (contracts/looper/Ploopy.sol#2) allows old versions
Pragma version0.8.17 (contracts/looper/PloopyConstants.sol#2) allows old versions
Pragma version0.8.17 (contracts/looper/interfaces/IFlashLoanRecipient.sol#15) allows old versions
Pragma version0.8.17 (contracts/looper/interfaces/IVault.sol#15) allows old versions
Pragma version0.8.17 (contracts/looper/interfaces/Interfaces.sol#2) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Permit.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in SafeERC20._callOptionalReturnBool(IERC20,bytes) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#134-142):
        - (success,returndata) = address(token).call{data} (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#139)
Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#64-69):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#67)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#128-137):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#135)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#155-162):
        - (success,returndata) = target.staticcall{data} (node_modules/@openzeppelin/contracts/utils/Address.sol#160)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#180-187):
        - (success,returndata) = target.delegatecall{data} (node_modules/@openzeppelin/contracts/utils/Address.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Event PloopyplvGLPBalance(uint256) (contracts/looper/Ploopy.sol#80) is not in CapWords
Event PloopylTokenBalance(uint256) (contracts/looper/Ploopy.sol#81) is not in CapWords
Parameter Ploopy.loop(IERC20,uint256,uint16,uint16)._token (contracts/looper/Ploopy.sol#95) is not in mixedCase
Parameter Ploopy.loop(IERC20,uint256,uint16,uint16)._amount (contracts/looper/Ploopy.sol#95) is not in mixedCase
Parameter Ploopy.loop(IERC20,uint256,uint16,uint16)._leverage (contracts/looper/Ploopy.sol#95) is not in mixedCase
Parameter Ploopy.loop(IERC20,uint256,uint16,uint16)._useWalletBalance (contracts/looper/Ploopy.sol#95) is not in mixedCase
Parameter Ploopy.getNotionalLoanAmountInIe18(uint256,uint16)._notionalTokenAmountInIe18 (contracts/looper/Ploopy.sol#221) is not in mixedCase
Parameter Ploopy.getNotionalLoanAmountInIe18(uint256,uint16)._leverage (contracts/looper/Ploopy.sol#222) is not in mixedCase
Constant PloopyConstants.sGLP (contracts/looper/PloopyConstants.sol#21) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PloopyConstants.lUSDC (contracts/looper/PloopyConstants.sol#30) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PloopyConstants.lPLVGLP (contracts/looper/PloopyConstants.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PloopyConstants.lARB (contracts/looper/PloopyConstants.sol#36) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PloopyConstants.lWBTC (contracts/looper/PloopyConstants.sol#37) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PloopyConstants.lUSDT (contracts/looper/PloopyConstants.sol#38) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PloopyConstants.lDAI (contracts/looper/PloopyConstants.sol#39) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PloopyConstants.lFRAX (contracts/looper/PloopyConstants.sol#40) is not in UPPER_CASE_WITH_UNDERSCORES
Function IERC20Permit.DOMAIN_SEPARATOR() (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Permit.sol#59) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Function Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) contains magic numbers: 6, 6, 8, 18, 18, 18, 18
Function Ploopy.loop(IERC20,uint256,uint16,uint16) (contracts/looper/Ploopy.sol#95-161) contains magic number: 1e12
Function Address._revert(bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#231-243) contains magic number: 32
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
In a function Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) variable PloopyConstants.ARB (contracts/looper/PloopyConstants.sol#10) is read multiple times
In a function Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) variable PloopyConstants.DAI (contracts/looper/PloopyConstants.sol#13) is read multiple times
In a function Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) variable PloopyConstants.FRAX (contracts/looper/PloopyConstants.sol#14) is read multiple times
In a function Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) variable PloopyConstants.GLP (contracts/looper/PloopyConstants.sol#16) is read multiple times
In a function Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) variable PloopyConstants.PLVGLP (contracts/looper/PloopyConstants.sol#15) is read multiple times
In a function Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) variable PloopyConstants.REWARD_ROUTER_V2 (contracts/looper/PloopyConstants.sol#22-23) is read multiple times
In a function Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) variable PloopyConstants.USDC (contracts/looper/PloopyConstants.sol#9) is read multiple times
In a function Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) variable PloopyConstants.USDT (contracts/looper/PloopyConstants.sol#12) is read multiple times
In a function Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) variable PloopyConstants.VAULT (contracts/looper/PloopyConstants.sol#19) is read multiple times
In a function Ploopy.constructor() (contracts/looper/Ploopy.sol#19-72) variable PloopyConstants.WBTC (contracts/looper/PloopyConstants.sol#11) is read multiple times
In a function Ploopy.loop(IERC20,uint256,uint16,uint16) (contracts/looper/Ploopy.sol#95-161) variable PloopyConstants.BALANCER_VAULT (contracts/looper/PloopyConstants.sol#8) is read multiple times
In a function Ploopy.receiveFlashLoan(IERC20[],uint256[],uint256[],bytes) (contracts/looper/Ploopy.sol#164-218) variable PloopyConstants.PLVGLP (contracts/looper/PloopyConstants.sol#15) is read multiple times
In a function Ploopy.receiveFlashLoan(IERC20[],uint256[],uint256[],bytes) (contracts/looper/Ploopy.sol#164-218) variable Ploopy.lTokenMapping (contracts/looper/Ploopy.sol#17) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

- All the reentrancies flagged by Slither were checked individually and are false positives.
- No major issues found by Slither.

7.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on the smart contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

MythX results:

- MythX flagged some integer overflows and underflows which all were false positives, as the contracts are using Solidity 0.8.19 version. After the Solidity version 0.8.0 Arithmetic operations revert to underflow and overflow by default.
- MythX also flagged some assert violations, which were all considered to be false positives.
- No major issues were found by MythX.

THANK YOU FOR CHOOSING

// HALBORN