



Marketxyz - Compound and Fuse protocols

Smart Contract Security Audit

Prepared by: Halborn

Date of Engagement: March 14th, 2022 - March 28th, 2022

Visit: Halborn.com

DOCUMENT REVISION HISTORY	3
CONTACTS	3
1 EXECUTIVE OVERVIEW	4
1.1 INTRODUCTION	5
1.2 AUDIT SUMMARY	5
1.3 TEST APPROACH & METHODOLOGY	5
RISK METHODOLOGY	6
1.4 SCOPE	8
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW	9
3 FINDINGS & TECH DETAILS	10
3.1 (HAL-01) REQUIREMENT BYPASSED - LOW	12
Description	12
Code Location	12
Risk Level	13
Recommendation	13
Remediation Plan	14
3.2 (HAL-02) REENTRANCY VULNERABILITY WITH ERC777 TOKENS - INFORMATIONAL	15
Description	15
Code Location	15
Risk Level	20
Recommendation	20
Remediation Plan	21
3.3 (HAL-03) ZERO ADDRESS OWNER NOT CHECKED - INFORMATIONAL	22
Description	22

Code Location	22
Risk Level	22
Recommendation	23
Remediation Plan	23
3.4 (HAL-04) SAME SALT GENERATED IN THE SAME BLOCK - INFORMATIONAL	24
Description	24
Code Location	24
Risk Level	25
Recommendation	25
Remediation Plan	25
4 AUTOMATED TESTING	25
4.1 STATIC ANALYSIS REPORT	27
Description	27
Slither results	27
4.2 AUTOMATED SECURITY SCAN	63
Description	63
MythX results	63

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	3/25/2022	Omar Alshaeb
1.0	Remediation Plan	07/05/2022	Omar Alshaeb
1.1	Remediation Plan Review	07/06/2022	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
Omar Alshaeb	Halborn	Omar.Alshaeb@halborn.com



EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Marketxyz engaged Halborn to conduct a security audit on their smart contracts beginning on March 14th, 2022 and ending on March 28th, 2022 . The security assessment was scoped to the smart contracts provided to the Halborn team.

1.2 AUDIT SUMMARY

The team at Halborn was provided two weeks for the engagement and assigned a full-time security engineer to audit the security of the smart contract. 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 audit 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 accepted and acknowledged by the Marketxyz 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 audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the bridge code and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- 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 hotspots or bugs. ([MythX](#))
- Static Analysis of security for scoped contract, and imported functions. ([Slither](#))
- Testnet deployment ([Brownie](#), [Remix IDE](#))

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

- 5 - Almost certain an incident will occur.
- 4 - High probability of an incident occurring.
- 3 - Potential of a security incident in the long term.
- 2 - Low probability of an incident occurring.
- 1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

- 5 - May cause devastating and unrecoverable impact or loss.
- 4 - May cause a significant level of impact or loss.

- 3 - May cause a partial impact or loss to many.
- 2 - May cause temporary impact or loss.
- 1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.



- 10 - CRITICAL
- 9 - 8 - HIGH
- 7 - 6 - MEDIUM
- 5 - 4 - LOW
- 3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

IN-SCOPE:

The security assessment was scoped to the following `compound smart contracts`:

- `CErc20.sol`
- `CErc20Delegate.sol`
- `CEther.sol`
- `CEtherDelegate.sol`
- `CToken.sol`
- `CTokenInterfaces.sol`
- `ComptrollerStorage.sol`

Commit ID: `338558451a49a2d94c6f0757b5683bfe04d5d8c2`

And the following `fuse smart contracts`:

- `MarketAdmin.sol`
- `MarketAdminDeployer.sol`
- `FusePoolDirectory.sol`
- `FuseFeeDistributor.sol`
- `oracles/AaveOracle.sol`
- `oracles/BeefyV6Oracle.sol`
- `oracles/ChainlinkPriceOracleV2.sol`
- `oracles/CurvePlainPoolOracle.sol` and `oracles/CurvePoolOracle.sol`
- `oracles/MasterPriceOracleV2.sol`
- `oracles/PreferredPriceOracle.sol`
- `oracles/UniV2LPOracle.sol`
- `oracles/UniswapTwapPriceOracleV2.sol`
- `oracles/UniswapTwapPriceOracleV2Root.sol`
- `oracles/UnwrapAssetOracle.sol`
- `oracles/ATriCrypto3Oracle.sol`

Commit ID: `0b990cecafc54c1224408724de82e974fdeea8fb`

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	1	3

LIKELIHOOD

IMPACT

(HAL-02)				
(HAL-03) (HAL-04)		(HAL-01)		

SECURITY ANALYSIS	RISK LEVEL	REMEDATION DATE
HAL01 - REQUIREMENT BYPASSED	Low	RISK ACCEPTED
HAL02 - REENTRANCY VULNERABILITY WITH ERC777 TOKENS	Informational	ACKNOWLEDGED
HAL03 - ZERO ADDRESS OWNER NOT CHECKED	Informational	ACKNOWLEDGED
HAL04 - SAME SALT GENERATED IN THE SAME BLOCK	Informational	ACKNOWLEDGED



FINDINGS & TECH DETAILS

3.1 (HAL-01) REQUIREMENT BYPASSED – LOW

Description:

The functions `deployCEther` and `deployCErc20` within the contract `FuseFeeDistributor.sol` are not checking properly the sender of the transaction is the comptroller contract. An arbitrary account can call these functions, bypassing the requirement.

Code Location:

Listing 1: FuseFeeDistributor.sol (Lines 103,124)

```
100     function deployCEther(bytes calldata constructorData) external
↳ returns (address) {
101         // Make sure comptroller == msg.sender
102         (address comptroller) = abi.decode(constructorData[0:32],
↳ (address));
103         require(comptroller == msg.sender, "Comptroller is not
↳ sender.");
104
105         // Deploy Unitroller using msg.sender, underlying, and
↳ block.number as a salt
106         bytes memory cEtherDelegatorCreationCode = hex"608060405
↳ ...";
107         cEtherDelegatorCreationCode = abi.encodePacked(
↳ cEtherDelegatorCreationCode, constructorData);
108         bytes32 salt = keccak256(abi.encodePacked(msg.sender,
↳ address(0), block.number));
109         address proxy;
110
111         assembly {
112             proxy := create2(0, add(cEtherDelegatorCreationCode,
↳ 32), mload(cEtherDelegatorCreationCode), salt)
113             if iszero(extcodesize(proxy)) {
114                 revert(0, "Failed to deploy CEther.")
115             }
116         }
117
```

```
118         return proxy;
119     }
120
121     function deployCErc20(bytes calldata constructorData) external
122     ↪ returns (address) {
123         // Make sure comptroller == msg.sender
124         ↪ (address underlying, address comptroller) = abi.decode(
125         ↪ constructorData[0:64], (address, address));
126         ↪ require(comptroller == msg.sender, "Comptroller is not
127         ↪ sender.");
128
129         // Deploy CErc20Delegator using msg.sender, underlying,
130         ↪ and block.number as a salt
131         ↪ bytes memory cErc20DelegatorCreationCode = hex"608060405
132         ↪ ...";
133         ↪ cErc20DelegatorCreationCode = abi.encodePacked(
134         ↪ cErc20DelegatorCreationCode, constructorData);
135         ↪ bytes32 salt = keccak256(abi.encodePacked(msg.sender,
136         ↪ underlying, block.number));
137         ↪ address proxy;
138
139         assembly {
140             ↪ proxy := create2(0, add(cErc20DelegatorCreationCode,
141             ↪ 32), mload(cErc20DelegatorCreationCode), salt)
142             ↪ if iszero(extcodesize(proxy)) {
143             ↪     ↪ revert(0, "Failed to deploy CErc20.")
144             ↪ }
145             ↪ }
146         ↪ return proxy;
147     }
148 }
```

Risk Level:

Likelihood - 3

Impact - 1

Recommendation:

The function should thoroughly check if the sender is the comptroller contract.

Remediation Plan:

RISK ACCEPTED: The Marketxyz team accepted the risk of this issue.

3.2 (HAL-02) REENTRANCY VULNERABILITY WITH ERC777 TOKENS – INFORMATIONAL

Description:

The functions `redeemFresh` and `borrowFresh` within the contract `CToken.sol` are not following the Checks-Effects-Interactions pattern in order to avoid having issues related to reentrancy. The tokens are being transferred before updating the state variables first.

Code Location:

Listing 2: CToken.sol (Line 712)

```

632     function redeemFresh(address payable redeemer, uint
    ↳ redeemTokensIn, uint redeemAmountIn) internal returns (uint) {
633         require(redeemTokensIn == 0 || redeemAmountIn == 0, "one
    ↳ of redeemTokensIn or redeemAmountIn must be zero");
634
635         RedeemLocalVars memory vars;
636
637         /* exchangeRate = invoke Exchange Rate Stored() */
638         (vars.mathErr, vars.exchangeRateMantissa) =
    ↳ exchangeRateStoredInternal();
639         if (vars.mathErr != MathError.NO_ERROR) {
640             return failOpaque(Error.MATH_ERROR, FailureInfo.
    ↳ REDEEM_EXCHANGE_RATE_READ_FAILED, uint(vars.mathErr));
641         }
642
643         /* If redeemTokensIn > 0: */
644         if (redeemTokensIn > 0) {
645             /*
646              * We calculate the exchange rate and the amount of
    ↳ underlying to be redeemed:
647              * redeemTokens = redeemTokensIn
648              * redeemAmount = redeemTokensIn x
    ↳ exchangeRateCurrent
649              */

```

```
650         vars.redeemTokens = redeemTokensIn;
651
652         (vars.mathErr, vars.redeemAmount) = mulScalarTruncate(
        ↳ Exp({mantissa: vars.exchangeRateMantissa}), redeemTokensIn);
653         if (vars.mathErr != MathError.NO_ERROR) {
654             return failOpaque(Error.MATH_ERROR, FailureInfo.
        ↳ REDEEM_EXCHANGE_TOKENS_CALCULATION_FAILED, uint(vars.mathErr));
655         }
656     } else {
657         /*
658          * We get the current exchange rate and calculate the
        ↳ amount to be redeemed:
659          * redeemTokens = redeemAmountIn / exchangeRate
660          * redeemAmount = redeemAmountIn
661          */
662
663         (vars.mathErr, vars.redeemTokens) =
        ↳ divScalarByExpTruncate(redeemAmountIn, Exp({mantissa: vars.
        ↳ exchangeRateMantissa}));
664         if (vars.mathErr != MathError.NO_ERROR) {
665             return failOpaque(Error.MATH_ERROR, FailureInfo.
        ↳ REDEEM_EXCHANGE_AMOUNT_CALCULATION_FAILED, uint(vars.mathErr));
666         }
667
668         vars.redeemAmount = redeemAmountIn;
669     }
670
671     /* Fail if redeem not allowed */
672     uint allowed = comptroller.redeemAllowed(address(this),
        ↳ redeemer, vars.redeemTokens);
673     if (allowed != 0) {
674         return failOpaque(Error.COMPTROLLER_REJECTION,
        ↳ FailureInfo.REDEEM_COMPTRROLLER_REJECTION, allowed);
675     }
676
677     /* Verify market's block number equals current block
        ↳ number */
678     if (accrualBlockNumber != getBlockNumber()) {
679         return fail(Error.MARKET_NOT_FRESH, FailureInfo.
        ↳ REDEEM_FRESHNESS_CHECK);
680     }
681
682     /*
```

```
683         * We calculate the new total supply and redeemer balance,
        ↳ checking for underflow:
684         * totalSupplyNew = totalSupply - redeemTokens
685         * accountTokensNew = accountTokens[redeemer] -
        ↳ redeemTokens
686         */
687         (vars.mathErr, vars.totalSupplyNew) = subUInt(totalSupply,
        ↳ vars.redeemTokens);
688         if (vars.mathErr != MathError.NO_ERROR) {
689             return failOpaque(Error.MATH_ERROR, FailureInfo.
        ↳ REDEEM_NEW_TOTAL_SUPPLY_CALCULATION_FAILED, uint(vars.mathErr));
690         }
691
692         (vars.mathErr, vars.accountTokensNew) = subUInt(
        ↳ accountTokens[redeemer], vars.redeemTokens);
693         if (vars.mathErr != MathError.NO_ERROR) {
694             return failOpaque(Error.MATH_ERROR, FailureInfo.
        ↳ REDEEM_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED, uint(vars.mathErr))
        ↳ ;
695         }
696
697         /* Fail gracefully if protocol has insufficient cash */
698         if (getCashPrior() < vars.redeemAmount) {
699             return fail(Error.TOKEN_INSUFFICIENT_CASH, FailureInfo
        ↳ .REDEEM_TRANSFER_OUT_NOT_POSSIBLE);
700         }
701
702         //////////////////////////////////////
703         // EFFECTS & INTERACTIONS
704         // (No safe failures beyond this point)
705
706         /*
707         * We invoke doTransferOut for the redeemer and the
        ↳ redeemAmount.
708         * Note: The cToken must handle variations between ERC-20
        ↳ and ETH underlying.
709         * On success, the cToken has redeemAmount less of cash.
710         * doTransferOut reverts if anything goes wrong, since we
        ↳ can't be sure if side effects occurred.
711         */
712         doTransferOut(redeemer, vars.redeemAmount);
713
714         /* We write previously calculated values into storage */
715         totalSupply = vars.totalSupplyNew;
```

```

716     accountTokens[redeemer] = vars.accountTokensNew;
717
718     /* We emit a Transfer event, and a Redeem event */
719     emit Transfer(redeemer, address(this), vars.redeemTokens);
720     emit Redeem(redeemer, vars.redeemAmount, vars.redeemTokens
    ↪ );
721
722     /* We call the defense hook */
723     comptroller.redeemVerify(address(this), redeemer, vars.
    ↪ redeemAmount, vars.redeemTokens);
724
725     return uint(Error.NO_ERROR);
726 }

```

Listing 3: CToken.sol (Line 812)

```

755     function borrowFresh(address payable borrower, uint
    ↪ borrowAmount) internal returns (uint) {
756         /* Fail if borrow not allowed */
757         uint allowed = comptroller.borrowAllowed(address(this),
    ↪ borrower, borrowAmount);
758         if (allowed != 0) {
759             return failOpaque(Error.COMPTROLLER_REJECTION,
    ↪ FailureInfo.BORROW_COMPTRROLLER_REJECTION, allowed);
760         }
761
762         /* Verify market's block number equals current block
    ↪ number */
763         if (accrualBlockNumber != getBlockNumber()) {
764             return fail(Error.MARKET_NOT_FRESH, FailureInfo.
    ↪ BORROW_FRESHNESS_CHECK);
765         }
766
767         /* Fail gracefully if protocol has insufficient underlying
    ↪ cash */
768         uint cashPrior = getCashPrior();
769
770         if (cashPrior < borrowAmount) {
771             return fail(Error.TOKEN_INSUFFICIENT_CASH, FailureInfo
    ↪ .BORROW_CASH_NOT_AVAILABLE);
772         }
773
774         BorrowLocalVars memory vars;
775

```

```
776     /*
777     * We calculate the new borrower and total borrow balances
778     *   , failing on overflow:
779     *   accountBorrowsNew = accountBorrows + borrowAmount
780     *   totalBorrowsNew = totalBorrows + borrowAmount
781     */
782     (vars.mathErr, vars.accountBorrows) =
783     ↪ borrowBalanceStoredInternal(borrower);
784     if (vars.mathErr != MathError.NO_ERROR) {
785         return failOpaque(Error.MATH_ERROR, FailureInfo.
786         ↪ BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED, uint(vars.mathErr))
787     };
788     }
789     (vars.mathErr, vars.accountBorrowsNew) = addUInt(vars.
790     ↪ accountBorrows, borrowAmount);
791     if (vars.mathErr != MathError.NO_ERROR) {
792         return failOpaque(Error.MATH_ERROR, FailureInfo.
793         ↪ BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED, uint(vars.
794         ↪ mathErr));
795     }
796     // Check min borrow for this user for this asset
797     allowed = comptroller.borrowWithinLimits(address(this),
798     ↪ vars.accountBorrowsNew);
799     if (allowed != 0) {
800         return failOpaque(Error.COMPTROLLER_REJECTION,
801         ↪ FailureInfo.BORROW_COMPROLLER_REJECTION, allowed);
802     }
803     (vars.mathErr, vars.totalBorrowsNew) = addUInt(
804     ↪ totalBorrows, borrowAmount);
805     if (vars.mathErr != MathError.NO_ERROR) {
806         return failOpaque(Error.MATH_ERROR, FailureInfo.
807         ↪ BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED, uint(vars.mathErr));
808     }
809     }
810     ////////////////////////////////////////////////////
811     // EFFECTS & INTERACTIONS
812     // (No safe failures beyond this point)
813     /*
814     * We invoke doTransferOut for the borrower and the
815     ↪ borrowAmount.
```

```
808     * Note: The cToken must handle variations between ERC-20
      ↳ and ETH underlying.
809     * On success, the cToken borrowAmount less of cash.
810     * doTransferOut reverts if anything goes wrong, since we
      ↳ can't be sure if side effects occurred.
811     */
812     doTransferOut(borrower, borrowAmount);
813
814     /* We write the previously calculated values into storage
      ↳ */
815     accountBorrows[borrower].principal = vars.
      ↳ accountBorrowsNew;
816     accountBorrows[borrower].interestIndex = borrowIndex;
817     totalBorrows = vars.totalBorrowsNew;
818
819     /* We emit a Borrow event */
820     emit Borrow(borrower, borrowAmount, vars.accountBorrowsNew
      ↳ , vars.totalBorrowsNew);
821
822     /* We call the defense hook */
823     // unused function
824     // comptroller.borrowVerify(address(this), borrower,
      ↳ borrowAmount);
825
826     return uint(Error.NO_ERROR);
827 }
```

Risk Level:**Likelihood - 1****Impact - 2****Recommendation:**

It is strongly suggested to update the storage variables before performing the external call to transfer the tokens following the Checks-Effects-Interactions pattern.

Remediation Plan:

ACKNOWLEDGED: The [Marketxyz team](#) acknowledged this issue.

3.3 (HAL-03) ZERO ADDRESS OWNER NOT CHECKED - INFORMATIONAL

Description:

The constructor of the contract `JumpRateModelV2.sol` is not checking if the owner passed as a parameter is the zero address.

Code Location:

Listing 4: `JumpRateModelV2.sol` (Line 79)

```
71     constructor(  
72         uint256 baseRatePerYear,  
73         uint256 multiplierPerYear,  
74         uint256 jumpMultiplierPerYear,  
75         uint256 kink_,  
76         uint256 roof_,  
77         address owner_  
78     ) public {  
79         owner = owner_;  
80  
81         updateJumpRateModelInternal(  
82             baseRatePerYear,  
83             multiplierPerYear,  
84             jumpMultiplierPerYear,  
85             kink_,  
86             roof_  
87         );  
88     }
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

Every address set with user input should be checked to not be the zero address.

Remediation Plan:

ACKNOWLEDGED: The [Marketxyz team](#) acknowledged this issue.

3.4 (HAL-04) SAME SALT GENERATED IN THE SAME BLOCK – INFORMATIONAL

Description:

The function `deploy` within the contract `MarketAdminDeployer.sol` is using `block.number` to create a salt and use it to deploy a `MarketAdmin` contract. This can lead to a problem where more than one `MarketAdmin` need to be deployed in the same block due to attempting to deploy the second contract in the same address the first one was properly deployed.

Code Location:

Listing 5: `MarketAdminDeployer.sol` (Line 27)

```
18     function deploy(address comptroller, address manager)
19     external
20     returns (address)
21     {
22         if (comptroller == address(0) || manager == address(0)) {
23             revert ZeroAddressProvided();
24         }
25
26         bytes32 salt = keccak256(
27             abi.encodePacked(comptroller, manager, block.number)
28         );
29
30         bytes memory creationCode = abi.encodePacked(
31             type(MarketAdmin).creationCode,
32             abi.encode(comptroller, manager)
33         );
34
35         address deployed;
36
37         assembly {
38             deployed := create2(
39                 0,
40                 add(creationCode, 0x20),
41                 mload(creationCode),
42                 salt
```

```
43     )
44     if iszero(extcodesize(deployed)) {
45         revert(0, "Failed to deploy MarketAdmin")
46     }
47 }
48
49     emit MarketAdminDeployed(compcontroller, manager, deployed);
50
51     return deployed;
52 }
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

Use a sort of nonce to generate the salt.

Remediation Plan:

ACKNOWLEDGED: The [Marketxyz team](#) acknowledged this issue.



AUTOMATED TESTING

4.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the scoped contracts. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their ABI and binary formats, Slither was run on the all-scoped 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:

Aave0racle.sol

```

Different versions of Solidity is used:
- Version used: ["0.8.4", "0.8.0", "0.8.4"]
- 0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4)
- 0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4)
- 0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4)
- 0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
- 0.8.4 (contracts/external/latest/interfaces.sol#2)
- 0.8.4 (contracts/oracles/aave0racle.sol#5)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#23) is never used and should be removed
ERC20_burn(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#288-295) is never used and should be removed
ERC20_mint(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#257-267) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4) allows old versions
Pragma version0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4) allows old versions
Pragma version0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function Comptroller._setMinCollateralFactorEnforcement(bool) (contracts/external/latest/Interfaces.sol#56) is not in mixedCase
Function Comptroller._setMinCollateralFactor(address,bool) (contracts/external/latest/Interfaces.sol#56-59) is not in mixedCase
Function Comptroller._become(address) (contracts/external/latest/Interfaces.sol#61) is not in mixedCase
Function Comptroller._setPriceOracle(address) (contracts/external/latest/Interfaces.sol#63) is not in mixedCase
Function Comptroller._setCollateralFactor(address,uint256) (contracts/external/latest/Interfaces.sol#69-71) is not in mixedCase
Function Comptroller._acceptAdmin() (contracts/external/latest/Interfaces.sol#83) is not in mixedCase
Function Comptroller._setClassFactor(uint256) (contracts/external/latest/Interfaces.sol#85-87) is not in mixedCase
Function Comptroller._setLiquidationIncentive(uint256) (contracts/external/latest/Interfaces.sol#89-91) is not in mixedCase
Function Comptroller._supportMarket(address) (contracts/external/latest/Interfaces.sol#93) is not in mixedCase
Function Comptroller._setMarketBorrowCaps(address,uint256) (contracts/external/latest/Interfaces.sol#99-101) is not in mixedCase
Function Comptroller._setMarketSupplyCaps(address,uint256) (contracts/external/latest/Interfaces.sol#103-105) is not in mixedCase
Function Comptroller._setMinPaused(address,bool) (contracts/external/latest/Interfaces.sol#105) is not in mixedCase
Function Comptroller._setBorrowPaused(address,bool) (contracts/external/latest/Interfaces.sol#107-109) is not in mixedCase
Function Comptroller._setSeizePaused(bool) (contracts/external/latest/Interfaces.sol#111) is not in mixedCase
Function Comptroller._setRewardsDistributor(address) (contracts/external/latest/Interfaces.sol#113-117) is not in mixedCase
Function Comptroller._setSupportMarket(bool) (contracts/external/latest/Interfaces.sol#119-123) is not in mixedCase
Function Comptroller._unSupportMarket(CToken) (contracts/external/latest/Interfaces.sol#125) is not in mixedCase
Function Comptroller._setPauseGuardian(address) (contracts/external/latest/Interfaces.sol#127-129) is not in mixedCase
Function Comptroller._setBorrowGuardian(address) (contracts/external/latest/Interfaces.sol#131) is not in mixedCase
Function CToken._supportMarketAndSetCollateralFactor(CToken,uint256) (contracts/external/latest/Interfaces.sol#184-187) is not in mixedCase
Function CToken._setAdminFee(uint256) (contracts/external/latest/Interfaces.sol#189-191) is not in mixedCase
Function CToken._setClassFactor(uint256) (contracts/external/latest/Interfaces.sol#193-195) is not in mixedCase
Function CToken._setInterestRateModel(address) (contracts/external/latest/Interfaces.sol#197-199) is not in mixedCase
Function CToken._setMortgageSymbol(string) (contracts/external/latest/Interfaces.sol#201-202) is not in mixedCase
Function CToken._withdrawAdminFee(uint256) (contracts/external/latest/Interfaces.sol#204-206) is not in mixedCase
Function CToken._reduceReserves(uint256) (contracts/external/latest/Interfaces.sol#208) is not in mixedCase
Function Luniiswap2Router01.WITH (contracts/external/latest/Interfaces.sol#214) is not in mixedCase
Function Luniiswap2Pair.DOMAIN_SEPARATOR() (contracts/external/latest/Interfaces.sol#58) is not in mixedCase
Function Luniiswap2Pair.PERMIT_TYPEHASH() (contracts/external/latest/Interfaces.sol#52) is not in mixedCase
Function Luniiswap2Pair.MINIMUM_LIQUIDITY() (contracts/external/latest/Interfaces.sol#83) is not in mixedCase
Function AMMToken.UNDERLYING_ASSET_ADDRESS() (contracts/oracles/AaveOracle.sol#8) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable Luniiswap2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountDesired (contracts/external/latest/Interfaces.sol#319) is too similar to Luniiswap2Router01.addLiquidity(address,uint256).amountDesired (contracts/external/latest/Interfaces.sol#328)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

name() should be declared external:
- ERC20_name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#62-64)
symbol() should be declared external:
- ERC20_symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)
decimals() should be declared external:
- ERC20_decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#87-89)
totalSupply() should be declared external:
- ERC20_totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#94-96)
balanceOf(address) should be declared external:
- ERC20_balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#101-103)
transfer(address,uint256) should be declared external:
- ERC20_transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#113-117)
approve(address,uint256) should be declared external:
- ERC20_approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#136-140)
transferFrom(address,address,uint256) should be declared external:
- ERC20_transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#158-162)
increaseAllowance(address,uint256) should be declared external:
- ERC20_increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#181-185)
decreaseAllowance(address,uint256) should be declared external:
- ERC20_decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#201-205)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

```

BeefyV6Oracle.sol

Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol:21-23) is never used and should be removed

ERC20._burn(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol:888-295) is never used and should be removed

ERC20._mint(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol:827-267) is never used and should be removed

Reference: <https://github.com/crytic/sliether/wiki/Detector-Documentation#dead-code>

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol:4) allows old versions

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol:4) allows old versions

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/extensions/ERC20Metadata.sol:4) allows old versions

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol:4) allows old versions

Reference: <https://github.com/crytic/sliether/wiki/Detector-Documentation#incorrect-versions-of-solidity>

Function Comptroller_setWhitelistEnforcement(bool) (contracts/external/latest/Interfaces.sol:64) is not in mixedCase

Function Comptroller_setWhitelistStatuses(address[],bool[]) (contracts/external/latest/Interfaces.sol:66-59) is not in mixedCase

Function Comptroller_setPriceOracle(address) (contracts/external/latest/Interfaces.sol:63) is not in mixedCase

Function Comptroller_setCollateralFactor(address,uint256) (contracts/external/latest/Interfaces.sol:69-72) is not in mixedCase

Function Comptroller_setKeepAdmin() (contracts/external/latest/Interfaces.sol:83) is not in mixedCase

Function Comptroller_setCloseFactor(uint256) (contracts/external/latest/Interfaces.sol:88-87) is not in mixedCase

Function Comptroller_setLiquidityIncentive(uint256) (contracts/external/latest/Interfaces.sol:89-91) is not in mixedCase

Function Comptroller_setMarketBorrowCaps(address[],uint256[]) (contracts/external/latest/Interfaces.sol:95-98) is not in mixedCase

Function Comptroller_setMarketSupplyCaps(address[],uint256[]) (contracts/external/latest/Interfaces.sol:100-103) is not in mixedCase

Function Comptroller_setMinPaused(address,bool) (contracts/external/latest/Interfaces.sol:105) is not in mixedCase

Function Comptroller_setBorrowPaused(address,bool) (contracts/external/latest/Interfaces.sol:107-109) is not in mixedCase

Function Comptroller_setTransferPaused(bool) (contracts/external/latest/Interfaces.sol:111) is not in mixedCase

Function Comptroller_setSizeOfPaused(bool) (contracts/external/latest/Interfaces.sol:113) is not in mixedCase

Function Comptroller_addRewardsDistributor(address) (contracts/external/latest/Interfaces.sol:115-117) is not in mixedCase

Function Comptroller_deployMarket(bool,bytes,uint256) (contracts/external/latest/Interfaces.sol:119-123) is not in mixedCase

Function Comptroller_unsupportMarket(CToken) (contracts/external/latest/Interfaces.sol:125) is not in mixedCase

Function Comptroller_setPauseGuardian(address) (contracts/external/latest/Interfaces.sol:127-129) is not in mixedCase

Function Comptroller_setBorrowCapGuardian(address) (contracts/external/latest/Interfaces.sol:131) is not in mixedCase

Function CToken_supportMarket(CollateralFactor,CToken) (contracts/external/latest/Interfaces.sol:136-137) is not in mixedCase

Function CToken_setAdminFee(uint256) (contracts/external/latest/Interfaces.sol:139-141) is not in mixedCase

Function CToken_setReserveFactor(uint256) (contracts/external/latest/Interfaces.sol:143-145) is not in mixedCase

Function CToken_setInterestRateModel(address) (contracts/external/latest/Interfaces.sol:147-149) is not in mixedCase

Function CToken_setNameAndSymbol(string,string) (contracts/external/latest/Interfaces.sol:151-152) is not in mixedCase

Function CToken_withdrawAdminFees(uint256) (contracts/external/latest/Interfaces.sol:154-156) is not in mixedCase

Function CToken_redeem(bytes,uint256) (contracts/external/latest/Interfaces.sol:158) is not in mixedCase

Function UniswapV2Router01.WETH() (contracts/external/latest/Interfaces.sol:161) is not in mixedCase

Function UniswapV2Pair.DOMAIN_SEPARATOR() (contracts/external/latest/Interfaces.sol:163) is not in mixedCase

Function UniswapV2Pair.PERMIT_TYPEHASH() (contracts/external/latest/Interfaces.sol:165) is not in mixedCase

Function UniswapV2Pair.MINIMUM_LIQUIDITY() (contracts/external/latest/Interfaces.sol:167) is not in mixedCase

Reference: <https://github.com/crytic/sliether/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions>

Variable UniswapV2Router01._addLiquidity(address,address,uint256,uint256,uint256,address,uint256).amountDesired (contracts/external/latest/Interfaces.sol:189) is too similar to UniswapV2Router01._addLiquidity(address,address,uint256,uint256,uint256,address,uint256).amountDesired (contracts/external/latest/Interfaces.sol:190)

Reference: <https://github.com/crytic/sliether/wiki/Detector-Documentation#variable-names-are-too-similar>

name() should be declared external:

- ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol:62-64)

symbol() should be declared external:

- ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol:70-72)

decimals() should be declared external:

- ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol:87-89)

totalSupply() should be declared external:

- ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol:94-96)

balanceOf(address) should be declared external:

- ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol:101-103)

transfer(address,uint256) should be declared external:

- ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol:113-117)

approve(address,uint256) should be declared external:

- ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol:136-140)

transferFrom(address,address,uint256) should be declared external:

- ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol:158-167)

increaseAllowance(address,uint256) should be declared external:

- ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol:181-185)

decreaseAllowance(address,uint256) should be declared external:

- ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol:191-195)

Reference: <https://github.com/crytic/sliether/wiki/Detector-Documentation#public-function-that-could-be-declared-external>

CErc20Delegate.sol

UnitrollerAdminStorage.admin (contracts/ComptrollerStorage.sol:16) is never initialized. It is used in:

- UnitrollerAdminStorage.hasAdminRights() (contracts/ComptrollerStorage.sol:36-38)

Reference: <https://github.com/crytic/sliether/wiki/Detector-Documentation#uninitialized-state-variables>

CErc20Delegate._becomeImplementation(bytes) (contracts/CErc20Delegate.sol:28-30) uses a Boolean constant improperly:

- false (contracts/CErc20Delegate.sol:29)

CErc20Delegate._resignImplementation() (contracts/CErc20Delegate.sol:35-40) uses a Boolean constant improperly:

- false (contracts/CErc20Delegate.sol:37)

Reference: <https://github.com/crytic/sliether/wiki/Detector-Documentation#misuse-of-a-boolean-constant>

EP2NonStandardInterface (contracts/EP2NonStandardInterface.sol:48-70) has incorrect ERC20 function interface:EP2NonStandardInterface.transfer(address,uint256) (contracts/EP2NonStandardInterface.sol:43)

EP2NonStandardInterface (contracts/EP2NonStandardInterface.sol:48-70) has incorrect ERC20 function interface:EP2NonStandardInterface.transferFrom(address,address,uint256) (contracts/EP2NonStandardInterface.sol:43)

Reference: <https://github.com/crytic/sliether/wiki/Detector-Documentation#incorrect-erc20-interface>

CToken accrueInterest() (contracts/CToken.sol:411-432) uses a dangerous strict equality:

- accrualBlockNumber == currentBlockNumber (contracts/CToken.sol:416)

CToken accrueInterest() (contracts/CToken.sol:411-432) uses a dangerous strict equality:

- require(bool, string) {err == MathError.NO_ERROR, balance could not be calculated} (contracts/CToken.sol:429)

CToken.balanceOfUnderlying(address) (contracts/CToken.sol:427-222) uses a dangerous strict equality:

- require(bool, string) {err == MathError.NO_ERROR, balance could not be calculated} (contracts/CToken.sol:428)

CToken.borrowUnderlying(address) (contracts/CToken.sol:428-302) uses a dangerous strict equality:

- require(bool, string) {err == MathError.NO_ERROR, borrowBalanceStoredInternal failed} (contracts/CToken.sol:300)

CarefulMath.div(uint256,uint256) (contracts/CarefulMath.sol:44-47) uses a dangerous strict equality:

- b == 0 (contracts/CarefulMath.sol:46)

CToken.exchangeRateStored() (contracts/CToken.sol:385-389) uses a dangerous strict equality:

- require(bool, string) {err == MathError.NO_ERROR, exchangeRateStoredInternal failed} (contracts/CToken.sol:387)

CToken.exchangeRateStoredInternal() (contracts/CToken.sol:386-396) uses a dangerous strict equality:

- totalSupply == 0 (contracts/CToken.sol:388)

CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint256,uint256) (contracts/CToken.sol:434-75) uses a dangerous strict equality:

- require(bool, string) {err == MathError.NO_ERROR, b == 0} (contracts/CToken.sol:435)

CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint256,uint256) (contracts/CToken.sol:434-75) uses a dangerous strict equality:

- require(bool, string) {err == uint256(Error.NO_ERROR), setting im failed} (contracts/CToken.sol:459)

CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint256,uint256) (contracts/CToken.sol:434-75) uses a dangerous strict equality:

- require(bool, string) {err == uint256(Error.NO_ERROR), setting reserve factor failed} (contracts/CToken.sol:467)

CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint256,uint256) (contracts/CToken.sol:434-75) uses a dangerous strict equality:

- require(bool, string) {err == uint256(Error.NO_ERROR), setting admin fee failed} (contracts/CToken.sol:473)

CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol:491-108) uses a dangerous strict equality:

- require(bool, string) {amountSeizeError == uint256(Error.NO_ERROR), LIQUIDATE_COMPROLLER_CALCULATE_AMOUNT_SEIZE_FAILED} (contracts/CToken.sol:106)

CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol:491-108) uses a dangerous strict equality:

- require(bool, string) {seizeError == uint256(Error.NO_ERROR), token seize failed} (contracts/CToken.sol:108)

CToken.minFresh(address,uint256) (contracts/CToken.sol:811-888) uses a dangerous strict equality:

- require(bool, string) {vars.mathErr == MathError.NO_ERROR, MINT_EXCHANGE_CALCULATION_FAILED} (contracts/CToken.sol:867)

Exponential.mulExp(ExponentialNoError,Exp.ExponentialNoError,Exp) (contracts/Exponential.sol:135-165) uses a dangerous strict equality:

- assert(bool) {err2 == MathError.NO_ERROR} (contracts/Exponential.sol:152)

CarefulMath.mul(uint256,uint256) (contracts/CarefulMath.sol:24-30) uses a dangerous strict equality:

- a == 0 || b == 0 (contracts/CarefulMath.sol:25)

ExponentialNoError.mul(uint256,uint256,string) (contracts/ExponentialNoError.sol:150-157) uses a dangerous strict equality:

- a == 0 || b == 0 (contracts/ExponentialNoError.sol:151)

ExponentialNoError.mul(uint256,uint256,string) (contracts/ExponentialNoError.sol:150-157) uses a dangerous strict equality:

- require(bool, string) {c / a == b, errorMessage} (contracts/ExponentialNoError.sol:155)

CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol:878-945) uses a dangerous strict equality:

- require(bool, string) {vars.mathErr == MathError.NO_ERROR, REPAY_BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED} (contracts/CToken.sol:927)

CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol:878-945) uses a dangerous strict equality:

- require(bool, string) {vars.mathErr == MathError.NO_ERROR, REPAY_BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED} (contracts/CToken.sol:930)

CToken.seizeInternal(address,address,address,uint256) (contracts/CToken.sol:1087-1147) uses a dangerous strict equality:

- require(bool, string) {vars.mathErr == MathError.NO_ERROR, exchange rate math error} (contracts/CToken.sol:1115)

CToken.transfer(address,uint256) (contracts/CToken.sol:125-164) uses a dangerous strict equality:

- transferTokens(msg.sender,msg.sender,dst.amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol:164)

CToken.transferFrom(address,address,uint256) (contracts/CToken.sol:173-175) uses a dangerous strict equality:

- transferTokens(msg.sender,src,dst.amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol:174)

Reference: <https://github.com/crytic/sliether/wiki/Detector-Documentation#dangerous-strict-equalities>

Contract locking error found:

- Contract CErc20Delegate (contracts/CErc20Delegate.sol:10-92) has payable functions:
 - _becomeImplementation() (contracts/CErc20Delegate.sol:32)
 - _resignImplementation() (contracts/CErc20Delegate.sol:36-91)
- But does not have a function to withdraw the ether

Reference: <https://github.com/crytic/sliether/wiki/Detector-Documentation#contracts-that-lock-ether>

Reentrancy in CToken_beforeNonReentrant(bool) (contracts/CToken.sol:1569-1573):

- External calls:

```

- comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
State variables written after the call(s):
- _notInterest = false (contracts/CToken.sol#1572)
Reentrancy in CToken_reduceReserves(uint256) (contracts/CToken.sol#1285-1293):
External calls:
- error = accrueInterest() (contracts/CToken.sol#1286)
- address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
- nonReentrant(false) (contracts/CToken.sol#1288)
- comptroller_afterNonReentrant() (contracts/CToken.sol#1582)
- comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
State variables written after the call(s):
- _reduceReserveFresh(reserveAmount) (contracts/CToken.sol#1292)
- totalReserves = totalReservesNew (contracts/CToken.sol#1333)
Reentrancy in CToken_setAdminFee(uint256) (contracts/CToken.sol#1176-1184):
External calls:
- error = accrueInterest() (contracts/CToken.sol#1177)
- address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
- nonReentrant(false) (contracts/CToken.sol#1176)
- comptroller_afterNonReentrant() (contracts/CToken.sol#1582)
- comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
State variables written after the call(s):
- _setAdminFeeFresh(newAdminFeeMantissa) (contracts/CToken.sol#1183)
- adminFeeMantissa = newAdminFeeMantissa (contracts/CToken.sol#1217)
- _setAdminFeeFresh(newAdminFeeMantissa) (contracts/CToken.sol#1183)
- fuseFeeMantissa = newFuseFeeMantissa (contracts/CToken.sol#1227)
Reentrancy in CToken_setInterestRateModel(InterestRateModel) (contracts/CToken.sol#1460-1471):
External calls:
- error = accrueInterest() (contracts/CToken.sol#1464)
- address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
- _setInterestRateModelFresh(newInterestRateModel) (contracts/CToken.sol#1470)
- address(oldInterestRateModel).call(abi.encodeWithSignature(resetInterestCheckpoints())) (contracts/CToken.sol#1506)
- address(newInterestRateModel).call(abi.encodeWithSignature(checkpointInterest())) (contracts/CToken.sol#1509)
State variables written after the call(s):
- _setInterestRateModelFresh(newInterestRateModel) (contracts/CToken.sol#1470)
- interestRateModel = newInterestRateModel (contracts/CToken.sol#1508)
Reentrancy in CToken_setReserveFactor(uint256) (contracts/CToken.sol#1241-1249):
External calls:
- error = accrueInterest() (contracts/CToken.sol#1242)
- address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
- nonReentrant(false) (contracts/CToken.sol#1241)
- comptroller_afterNonReentrant() (contracts/CToken.sol#1582)
- comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
State variables written after the call(s):
- _setReserveFactorFresh(newReserveFactorMantissa) (contracts/CToken.sol#1248)
- reserveFactorMantissa = newReserveFactorMantissa (contracts/CToken.sol#1273)
Reentrancy in CToken_withdrawAdminFees(uint256) (contracts/CToken.sol#1406-1412):
External calls:
- error = accrueInterest() (contracts/CToken.sol#1406)
- address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
- nonReentrant(false) (contracts/CToken.sol#1406)
- comptroller_afterNonReentrant() (contracts/CToken.sol#1582)
- comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
State variables written after the call(s):
- _withdrawAdminFeesFresh(withdrawAmount) (contracts/CToken.sol#1411)
- totalAdminFees = totalAdminFeesNew (contracts/CToken.sol#1449)
Reentrancy in CToken_withdrawFuseFees(uint256) (contracts/CToken.sol#1348-1356):
External calls:
- error = accrueInterest() (contracts/CToken.sol#1349)
- address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
- nonReentrant(false) (contracts/CToken.sol#1348)
- comptroller_afterNonReentrant() (contracts/CToken.sol#1582)
- comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
State variables written after the call(s):
- _withdrawFuseFeesFresh(withdrawAmount) (contracts/CToken.sol#1355)
- totalFuseFees = totalFuseFeesNew (contracts/CToken.sol#1371)
Reentrancy in CToken_borrowFresh(address,uint256) (contracts/CToken.sol#755-827):
External calls:
- allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#757)
- allowed = comptroller.borrowWithinLimits(address(this),vars.accountBorrowNew) (contracts/CToken.sol#792)
State variables written after the call(s):
- accountBorrow(borrower).principal = vars.accountBorrowNew (contracts/CToken.sol#815)
- accountBorrow(borrower).interestIndex = borrowIndex (contracts/CToken.sol#816)
Reentrancy in CToken_borrowInternal(uint256) (contracts/CToken.sol#733-741):
External calls:
- error = accrueInterest() (contracts/CToken.sol#734)
- address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
- borrowFresh(msg.sender,borrowAmount) (contracts/CToken.sol#740)
- allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#757)
- allowed = comptroller.borrowWithinLimits(address(this),vars.accountBorrowNew) (contracts/CToken.sol#792)
- nonReentrant(false) (contracts/CToken.sol#733)
- comptroller_afterNonReentrant() (contracts/CToken.sol#1582)
- comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
State variables written after the call(s):
- borrowFresh(msg.sender,borrowAmount) (contracts/CToken.sol#740)
- totalBorrow = vars.totalBorrowNew (contracts/CToken.sol#817)
Reentrancy in CToken_liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#956-970):
External calls:
- error = accrueInterest() (contracts/CToken.sol#956)
- address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
- error = cTokenCollateral accrueInterest() (contracts/CToken.sol#962)
- liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#969)
- allowed = comptroller.liquidateBorrowAllowed(address(this),sizeOfToken,liquidator,borrower,sizeOfTokens) (contracts/CToken.sol#1009)
- allowed = comptroller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#983)
- allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#888)
- sizeOfError = cTokenCollateral.sizeOf(liquidator,borrower,sizeOfTokens) (contracts/CToken.sol#1036)
- nonReentrant(false) (contracts/CToken.sol#956)
- comptroller_afterNonReentrant() (contracts/CToken.sol#1582)
- comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
State variables written after the call(s):
- liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#969)
- totalBorrow = vars.totalBorrowNew (contracts/CToken.sol#935)
- liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#969)
- totalReserves = vars.totalReservesNew (contracts/CToken.sol#1132)
Reentrancy in CToken_nonReentrant(bool) (contracts/CToken.sol#1558-1562):
External calls:
- _beforeNonReentrant(localOnly) (contracts/CToken.sol#1559)
- comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
- _afterNonReentrant(localOnly) (contracts/CToken.sol#1561)
- comptroller_afterNonReentrant() (contracts/CToken.sol#1582)
State variables written after the call(s):
- _afterNonReentrant(localOnly) (contracts/CToken.sol#1561)
- notEntered = true (contracts/CToken.sol#1581)
Reentrancy in CToken_redeemFresh(address,uint256,uint256) (contracts/CToken.sol#632-726):
External calls:
- allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/CToken.sol#672)
State variables written after the call(s):
- totalSupply = vars.totalSupplyNew (contracts/CToken.sol#715)
Reentrancy in CToken_repayBorrowBehalfInternal(address,uint256) (contracts/CToken.sol#850-858):
External calls:
- error = accrueInterest() (contracts/CToken.sol#851)
- address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
- repayBorrowFresh(msg.sender,borrower,repayAmount) (contracts/CToken.sol#857)
- allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#880)
- nonReentrant(false) (contracts/CToken.sol#850)
- comptroller_afterNonReentrant() (contracts/CToken.sol#1582)
- comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
State variables written after the call(s):
- repayBorrowFresh(msg.sender,borrower,repayAmount) (contracts/CToken.sol#857)
- totalBorrow = vars.totalBorrowNew (contracts/CToken.sol#935)
Reentrancy in CToken_repayBorrowInternal(uint256) (contracts/CToken.sol#834-842):
External calls:
- error = accrueInterest() (contracts/CToken.sol#835)
- address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
- repayBorrowFresh(msg.sender,msg.sender,repayAmount) (contracts/CToken.sol#841)
- allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#880)
- nonReentrant(false) (contracts/CToken.sol#834)
- comptroller_afterNonReentrant() (contracts/CToken.sol#1582)
- comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
State variables written after the call(s):
- repayBorrowFresh(msg.sender,msg.sender,repayAmount) (contracts/CToken.sol#841)
- totalBorrow = vars.totalBorrowNew (contracts/CToken.sol#935)
Reference: https://github.com/cryptic/sliether/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
CToken.finishInterestAccrual(uint256,uint256,uint256) (contracts/CToken.sol#437-476) ignores return value by address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMan

```



```
- _setInterestRateModelFresh(newInterestRateModel) (contracts/CToken.sol#1470)
Reentrancy in CToken_setReserveFactor(uint256) (contracts/CToken.sol#1241-1249):
  External calls:
    - error = accrueInterest() (contracts/CToken.sol#1242)
    - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
    - nonReentrant(false) (contracts/CToken.sol#1582)
    - controller_afterNonReentrant() (contracts/CToken.sol#1582)
    - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  Event emitted after the call(s):
    - AccrueInterest(cashPrior, interestAccumulated, borrowIndexNew, totalBorrowsNew) (contracts/CToken.sol#470)
    - error = accrueInterest() (contracts/CToken.sol#1242)
    - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
    - _setReserveFactorFresh(newReserveFactorMantissa) (contracts/CToken.sol#1248)
    - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
    - fail(TokenErrorReporter.Error(error), FailureInfo.SET_RESERVE_FACTOR_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#1246)
    - NewReserveFactor(oldReserveFactorMantissa, newReserveFactorMantissa) (contracts/CToken.sol#1275)
    - _setReserveFactorFresh(newReserveFactorMantissa) (contracts/CToken.sol#1248)
Reentrancy in CToken_withdrawFees(uint256) (contracts/CToken.sol#1348-1352):
  External calls:
    - error = accrueInterest() (contracts/CToken.sol#1349)
    - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
    - nonReentrant(false) (contracts/CToken.sol#1582)
    - controller_afterNonReentrant() (contracts/CToken.sol#1582)
    - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  Event emitted after the call(s):
    - AccrueInterest(cashPrior, interestAccumulated, borrowIndexNew, totalBorrowsNew) (contracts/CToken.sol#470)
    - error = accrueInterest() (contracts/CToken.sol#1349)
    - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
    - _withdrawAdminFeesFresh(withdrawAmount) (contracts/CToken.sol#1411)
    - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
    - fail(TokenErrorReporter.Error(error), FailureInfo.WITHDRAW_ADMIN_FEES_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#1408)
Reentrancy in CToken_withdrawFuseFees(uint256) (contracts/CToken.sol#1348-1356):
  External calls:
    - error = accrueInterest() (contracts/CToken.sol#1349)
    - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
    - nonReentrant(false) (contracts/CToken.sol#1582)
    - controller_afterNonReentrant() (contracts/CToken.sol#1582)
    - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  Event emitted after the call(s):
    - AccrueInterest(cashPrior, interestAccumulated, borrowIndexNew, totalBorrowsNew) (contracts/CToken.sol#470)
    - error = accrueInterest() (contracts/CToken.sol#1349)
    - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
    - _withdrawFuseFeesFresh(withdrawAmount) (contracts/CToken.sol#1355)
    - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
    - fail(TokenErrorReporter.Error(error), FailureInfo.WITHDRAW_FUSE_FEES_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#1352)
Reentrancy in CToken_borrowBalanceCurrent(address) (contracts/CToken.sol#288-291):
  External calls:
    - require(bool, string)(accrueInterest() == uint256(Error_NO_ERROR), accrue interest failed) (contracts/CToken.sol#289)
    - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
    - nonReentrant(false) (contracts/CToken.sol#290)
    - controller_afterNonReentrant() (contracts/CToken.sol#1582)
    - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  Event emitted after the call(s):
    - AccrueInterest(cashPrior, interestAccumulated, borrowIndexNew, totalBorrowsNew) (contracts/CToken.sol#470)
    - require(bool, string)(accrueInterest() == uint256(Error_NO_ERROR), accrue interest failed) (contracts/CToken.sol#289)
Reentrancy in CToken_borrowFresh(address, uint256) (contracts/CToken.sol#755-827):
  External calls:
    - allowed = controller.borrowAllowed(address(this), borrower, borrowAmount) (contracts/CToken.sol#757)
    - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
    - fail(Error.TOKEN_INSUFFICIENT_CASH, FailureInfo.BORROW_CASH_NOT_AVAILABLE) (contracts/CToken.sol#771)
    - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
    - failOpaque(Error.MATH_ERROR, FailureInfo.BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED, uint256(vars.mathErr)) (contracts/CToken.sol#788)
    - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
    - fail(Error.MARKET_NOT_FRESH, FailureInfo.BORROW_FRESHNESS_CHECK) (contracts/CToken.sol#764)
    - Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
    - failOpaque(Error.MATH_ERROR, FailureInfo.BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED, uint256(vars.mathErr)) (contracts/CToken.sol#783)
    - Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
    - failOpaque(Error.COMPTROLLER_REJECTION, FailureInfo.BORROW_COMPROLLER_REJECTION_ALLOWED) (contracts/CToken.sol#759)
Reentrancy in CToken_borrowFresh(address, uint256) (contracts/CToken.sol#755-827):
  External calls:
    - allowed = controller.borrowAllowed(address(this), borrower, borrowAmount) (contracts/CToken.sol#757)
    - allowed = controller.borrowWithinLimits(address(this), vars.accountBorrowsNew) (contracts/CToken.sol#792)
  Event emitted after the call(s):
    - Borrow(borrower, borrowAmount, vars.accountBorrowsNew, vars.totalBorrowsNew) (contracts/CToken.sol#820)
    - Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
    - failOpaque(Error.COMPTROLLER_REJECTION, FailureInfo.BORROW_COMPROLLER_REJECTION_ALLOWED) (contracts/CToken.sol#794)
    - Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
    - failOpaque(Error.MATH_ERROR, FailureInfo.BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED, uint256(vars.mathErr)) (contracts/CToken.sol#799)
Reentrancy in CToken_borrowInternal(uint256) (contracts/CToken.sol#733-741):
  External calls:
    - error = accrueInterest() (contracts/CToken.sol#734)
    - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
    - nonReentrant(false) (contracts/CToken.sol#733)
    - controller_afterNonReentrant() (contracts/CToken.sol#1582)
    - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  Event emitted after the call(s):
    - AccrueInterest(cashPrior, interestAccumulated, borrowIndexNew, totalBorrowsNew) (contracts/CToken.sol#470)
    - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
    - fail(TokenErrorReporter.Error(error), FailureInfo.BORROW_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#737)
Reentrancy in CToken_borrowInternal(uint256) (contracts/CToken.sol#733-741):
  External calls:
    - error = accrueInterest() (contracts/CToken.sol#734)
    - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
    - borrowFresh(msg.sender, borrowAmount) (contracts/CToken.sol#748)
    - allowed = controller.borrowAllowed(address(this), borrower, borrowAmount) (contracts/CToken.sol#757)
    - allowed = controller.borrowWithinLimits(address(this), vars.accountBorrowsNew) (contracts/CToken.sol#792)
    - nonReentrant(false) (contracts/CToken.sol#733)
    - controller_afterNonReentrant() (contracts/CToken.sol#1582)
    - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  Event emitted after the call(s):
    - Borrow(borrower, borrowAmount, vars.accountBorrowsNew, vars.totalBorrowsNew) (contracts/CToken.sol#820)
    - borrowFresh(msg.sender, borrowAmount) (contracts/CToken.sol#748)
    - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
    - borrowFresh(msg.sender, borrowAmount) (contracts/CToken.sol#748)
    - Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
    - borrowFresh(msg.sender, borrowAmount) (contracts/CToken.sol#748)
Reentrancy in CToken_exchangeRateCurrent() (contracts/CToken.sol#345-348):
  External calls:
    - require(bool, string)(accrueInterest() == uint256(Error_NO_ERROR), accrue interest failed) (contracts/CToken.sol#346)
    - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
    - nonReentrant(false) (contracts/CToken.sol#345)
    - controller_afterNonReentrant() (contracts/CToken.sol#1582)
    - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  Event emitted after the call(s):
    - AccrueInterest(cashPrior, interestAccumulated, borrowIndexNew, totalBorrowsNew) (contracts/CToken.sol#470)
    - require(bool, string)(accrueInterest() == uint256(Error_NO_ERROR), accrue interest failed) (contracts/CToken.sol#346)
Reentrancy in CToken_initialize(ComptrollerInterface, InterestRateModel, uint256, string, string, uint8, uint256, uint256) (contracts/CToken.sol#434-761):
  External calls:
    - err = _setInterestRateModelFresh(interestRateModel) (contracts/CToken.sol#458)
    - address(oldInterestRateModel).call(abi.encodeWithSignature(resetInterestCheckpoints())) (contracts/CToken.sol#1506)
    - address(newInterestRateModel).call(abi.encodeWithSignature(checkpointInterest())) (contracts/CToken.sol#1509)
  Event emitted after the call(s):
    - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
    - err = _setReserveFactorFresh(reserveFactorMantissa) (contracts/CToken.sol#466)
    - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
    - err = _setAdminFeesFresh(adminFeeMantissa) (contracts/CToken.sol#470)
    - NewAdminFees(oldAdminFeeMantissa, newAdminFeeMantissa) (contracts/CToken.sol#1228)
    - err = _setAdminFeesFresh(adminFeeMantissa) (contracts/CToken.sol#470)
    - NewFuseFee(oldFuseFeeMantissa, newFuseFeeMantissa) (contracts/CToken.sol#1230)
    - err = _setAdminFeesFresh(adminFeeMantissa) (contracts/CToken.sol#470)
    - NewReserveFactor(oldReserveFactorMantissa, newReserveFactorMantissa) (contracts/CToken.sol#1275)
    - err = _setReserveFactorFresh(reserveFactorMantissa) (contracts/CToken.sol#466)
Reentrancy in CToken_liquidateBorrowFresh(address, address, uint256, CTokenInterface) (contracts/CToken.sol#991-1050):
  External calls:
    - allowed = controller.liquidateBorrowAllowed(address(this), address(CTokenCollateral), liquidator, borrower, repayAmount) (contracts/CToken.sol#993)
  Event emitted after the call(s):
    - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
    - fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED, FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_UINT_MAX) (contracts/CToken.sol#1010)
    - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
    - fail(Error.MARKET_NOT_FRESH, FailureInfo.LIQUIDATE_COLLATERAL_FRESHNESS_CHECK) (contracts/CToken.sol#995)
    - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
    - fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED, FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_ZERO) (contracts/CToken.sol#1005)
```

```
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
- (fail(Error.INVALID_ACCOUNT_PAIR,FailureInfo.LIQUIDATE_LIQUIDATOR_IS_BORROWER),0) (contracts/CToken.sol#1800)
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - (fail(Error.MARKET_NOT_FRESH,FailureInfo.LIQUIDATE_FRESHNESS_CHECK),0) (contracts/CToken.sol#998)
- Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
- (fail(opaqueError.COMPTROLLER_REJECTION,FailureInfo.COMPTROLLER_REJECTION_ALLOWED),0) (contracts/CToken.sol#985)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#981-1050):
  External calls:
  - allowed = controller.liquidateBorrowAllowed(address(this),address(CTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#983)
  - (repayBorrowError.actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#1815)
  - allowed = controller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#886)
  Event emitted after the call(s):
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - (fail(TokenErrorReporter.Error.repayBorrowError,FailureInfo.LIQUIDATE_REPAY_BORROW_FRESH_FAILED),0) (contracts/CToken.sol#1917)
  - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
  - (repayBorrowError.actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#1815)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - (repayBorrowError.actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#1815)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - (repayBorrowError.payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/CToken.sol#938)
  - (repayBorrowError.actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#1815)
Reentrancy in CToken.liquidateBorrow(address,address,uint256,CTokenInterface) (contracts/CToken.sol#981-1050):
  External calls:
  - allowed = controller.liquidateBorrowAllowed(address(this),address(CTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#983)
  - (repayBorrowError.actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#1815)
  - allowed = controller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#886)
  - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#1834)
  - allowed = controller.seizeAllowed(address(this),seizeToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#1889)
  Event emitted after the call(s):
  - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
  - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#1834)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#1834)
  - ReserveAdded(address(this),vars.protocolSeizeAmount,vars.totalReserveNew) (contracts/CToken.sol#1138)
  - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/CToken.sol#1138)
  - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#1834)
  - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/CToken.sol#1139)
  - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#1834)
Reentrancy in CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#981-1050):
  External calls:
  - allowed = controller.liquidateBorrowAllowed(address(this),address(CTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#983)
  - (repayBorrowError.actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/CToken.sol#1815)
  - allowed = controller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#886)
  - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/CToken.sol#1834)
  - allowed = controller.seizeAllowed(address(this),seizeToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#1889)
  - seizeError = cTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/CToken.sol#1836)
  Event emitted after the call(s):
  - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(CTokenCollateral),seizeTokens) (contracts/CToken.sol#1843)
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#955-978):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#956)
  - address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#958)
  - controller._afterNonReentrant() (contracts/CToken.sol#1571)
  - controller._beforeNonReentrant() (contracts/CToken.sol#1571)
  Event emitted after the call(s):
  - AccrueInterest(controller,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/CToken.sol#479)
  - error = accrueInterest() (contracts/CToken.sol#956)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - (fail(TokenErrorReporter.Error.error),FailureInfo.LIQUIDATE_ACCRUE_BORROW_INTEREST_FAILED),0) (contracts/CToken.sol#959)
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#955-978):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#956)
  - address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#962)
  - nonReentrant(false) (contracts/CToken.sol#958)
  - controller._afterNonReentrant() (contracts/CToken.sol#1571)
  - controller._beforeNonReentrant() (contracts/CToken.sol#1571)
  Event emitted after the call(s):
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - (fail(TokenErrorReporter.Error.error),FailureInfo.LIQUIDATE_ACCRUE_COLLATERAL_INTEREST_FAILED),0) (contracts/CToken.sol#966)
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#955-978):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#956)
  - address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#962)
  - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#969)
  - allowed = controller.seizeAllowed(address(this),seizeToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#1889)
  - allowed = controller.liquidateBorrowAllowed(address(this),address(CTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#983)
  - allowed = controller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#886)
  - seizeError = cTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/CToken.sol#1836)
  - nonReentrant(false) (contracts/CToken.sol#958)
  - controller._afterNonReentrant() (contracts/CToken.sol#1571)
  - controller._beforeNonReentrant() (contracts/CToken.sol#1571)
  Event emitted after the call(s):
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#969)
  - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
  - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#969)
  - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(CTokenCollateral),seizeTokens) (contracts/CToken.sol#1843)
  - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#969)
  - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/CToken.sol#938)
  - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#969)
  - ReserveAdded(address(this),vars.protocolSeizeAmount,vars.totalReserveNew) (contracts/CToken.sol#1140)
  - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#969)
  - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/CToken.sol#1139)
  - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#969)
  - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/CToken.sol#1139)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#484-492):
  External calls:
  - allowed = controller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#451)
  Event emitted after the call(s):
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - (fail(Error.MARKET_NOT_FRESH,FailureInfo.MINT_FRESHNESS_CHECK),0) (contracts/CToken.sol#500)
  - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
  - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
  - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
  - (fail(opaqueError.COMPTROLLER_REJECTION,FailureInfo.MINT_COMPTROLLER_REJECTION_ALLOWED),0) (contracts/CToken.sol#451)
  - (fail(opaqueError.MATH_ERROR,FailureInfo.MINT_EXCHANGE_RATE_READ_FAILED,uint256(vars.mathErr)),0) (contracts/CToken.sol#627)
  - Mint(minter,vars.actualMintAmount,vars.mintTokens) (contracts/CToken.sol#457)
  - Transfer(address(this),minter,vars.mintTokens) (contracts/CToken.sol#457)
Reentrancy in CToken.mintInternal(uint256) (contracts/CToken.sol#484-492):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#485)
  - address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#484)
  - controller._afterNonReentrant() (contracts/CToken.sol#1571)
  - controller._beforeNonReentrant() (contracts/CToken.sol#1571)
  Event emitted after the call(s):
  - AccrueInterest(controller,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/CToken.sol#479)
  - error = accrueInterest() (contracts/CToken.sol#485)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - (fail(TokenErrorReporter.Error.error),FailureInfo.MINT_ACCRUE_INTEREST_FAILED),0) (contracts/CToken.sol#488)
Reentrancy in CToken.mintInternal(uint256) (contracts/CToken.sol#484-492):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#485)
  - address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - mintFresh(msg.sender,mintAmount) (contracts/CToken.sol#491)
  - allowed = controller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#451)
  - controller.mintVerify(address(this),minter,vars.actualMintAmount,vars.mintTokens) (contracts/CToken.sol#677)
  - nonReentrant(false) (contracts/CToken.sol#484)
  - controller._afterNonReentrant() (contracts/CToken.sol#1571)
  - controller._beforeNonReentrant() (contracts/CToken.sol#1571)
  Event emitted after the call(s):
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - mintFresh(msg.sender,mintAmount) (contracts/CToken.sol#491)
  - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
  - mintFresh(msg.sender,mintAmount) (contracts/CToken.sol#491)
  - Mint(minter,vars.actualMintAmount,vars.mintTokens) (contracts/CToken.sol#457)
  - mintFresh(msg.sender,mintAmount) (contracts/CToken.sol#491)
  - Transfer(address(this),minter,vars.mintTokens) (contracts/CToken.sol#457)
  - mintFresh(msg.sender,mintAmount) (contracts/CToken.sol#491)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#632-726):
  External calls:
```

```

- allowed = controller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/CToken.sol#672)
Event emitted after the call(s):
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(Error.MARKET_NOT_FRESH,FailureInfo.REDEEM_FRESHNESS_CHECK) (contracts/CToken.sol#679)
- Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#230)
  - failOpaque(Error.MATH_ERROR,FailureInfo.REDEEM_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/CToken.sol#694)
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.REDEEM_TRANSFER_OUT_NOT_POSSIBLE) (contracts/CToken.sol#699)
- Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#230)
  - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.REDEEM_COMPPTROLLER_REJECTION,allowed) (contracts/CToken.sol#674)
- Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#230)
  - failOpaque(Error.MATH_ERROR,FailureInfo.REDEEM_NEW_TOTAL_SUPPLY_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/CToken.sol#689)
- Redeem(redeemer,vars.redeemAmount,vars.redeemTokens) (contracts/CToken.sol#720)
- Transfer(redeemer,address(this),vars.redeemTokens) (contracts/CToken.sol#719)
Reentrancy in CToken.redeemInternal(uint256) (contracts/CToken.sol#558-594):
External calls:
- error = accrueInterest() (contracts/CToken.sol#689)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#673)
- nonReentrant(false) (contracts/CToken.sol#588)
- controller_afterNonReentrant() (contracts/CToken.sol#1582)
- controller_beforeNonReentrant() (contracts/CToken.sol#1574)
Event emitted after the call(s):
- AccrueInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/CToken.sol#478)
- error = accrueInterest() (contracts/CToken.sol#689)
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(TokenErrorReporter.Error(error),FailureInfo.REDEEM_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#692)
Reentrancy in CToken.redeemInternal(uint256) (contracts/CToken.sol#558-594):
External calls:
- error = accrueInterest() (contracts/CToken.sol#689)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#673)
- redeemFresh(msg.sender,redeemTokens,0) (contracts/CToken.sol#595)
- allowed = controller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/CToken.sol#672)
- controller_redeemVerify(passed,this,redeemer,vars.redeemAmount,vars.redeemTokens) (contracts/CToken.sol#723)
- nonReentrant(false) (contracts/CToken.sol#588)
- controller_afterNonReentrant() (contracts/CToken.sol#1582)
- controller_beforeNonReentrant() (contracts/CToken.sol#1574)
Event emitted after the call(s):
- Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#230)
  - redeemFresh(msg.sender,redeemTokens,0) (contracts/CToken.sol#595)
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - redeemFresh(msg.sender,redeemTokens,0) (contracts/CToken.sol#595)
- Redeem(redeemer,vars.redeemAmount,vars.redeemTokens) (contracts/CToken.sol#720)
- redeemFresh(msg.sender,redeemTokens,0) (contracts/CToken.sol#595)
- Transfer(redeemer,address(this),vars.redeemTokens) (contracts/CToken.sol#719)
- redeemFresh(msg.sender,redeemTokens,0) (contracts/CToken.sol#595)
Reentrancy in CToken.redeemUnderlyingInternal(uint256) (contracts/CToken.sol#608-612):
External calls:
- error = accrueInterest() (contracts/CToken.sol#608)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#673)
- nonReentrant(false) (contracts/CToken.sol#604)
- controller_afterNonReentrant() (contracts/CToken.sol#1582)
- controller_beforeNonReentrant() (contracts/CToken.sol#1574)
Event emitted after the call(s):
- AccrueInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/CToken.sol#478)
- error = accrueInterest() (contracts/CToken.sol#608)
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(TokenErrorReporter.Error(error),FailureInfo.REDEEM_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#608)
Reentrancy in CToken.redeemUnderlyingInternal(uint256) (contracts/CToken.sol#608-612):
External calls:
- error = accrueInterest() (contracts/CToken.sol#608)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#673)
- redeemFresh(msg.sender,0,redeemAmount) (contracts/CToken.sol#611)
- allowed = controller.redeemAllowed(address(this),redeemer,vars.redeemAmount,vars.redeemTokens) (contracts/CToken.sol#672)
- controller_redeemVerify(address(this),redeemer,vars.redeemAmount,vars.redeemTokens) (contracts/CToken.sol#723)
- nonReentrant(false) (contracts/CToken.sol#604)
- controller_afterNonReentrant() (contracts/CToken.sol#1582)
- controller_beforeNonReentrant() (contracts/CToken.sol#1574)
Event emitted after the call(s):
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - redeemFresh(msg.sender,0,redeemAmount) (contracts/CToken.sol#611)
- Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#230)
  - redeemFresh(msg.sender,0,redeemAmount) (contracts/CToken.sol#611)
- Redeem(redeemer,vars.redeemAmount,vars.redeemTokens) (contracts/CToken.sol#611)
- Transfer(redeemer,address(this),vars.redeemTokens) (contracts/CToken.sol#719)
- redeemFresh(msg.sender,0,redeemAmount) (contracts/CToken.sol#611)
Reentrancy in CToken.repayBorrowBehalfInternal(address,uint256) (contracts/CToken.sol#850-858):
External calls:
- error = accrueInterest() (contracts/CToken.sol#851)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
- nonReentrant(false) (contracts/CToken.sol#850)
- controller_afterNonReentrant() (contracts/CToken.sol#1582)
- controller_beforeNonReentrant() (contracts/CToken.sol#1574)
Event emitted after the call(s):
- AccrueInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/CToken.sol#478)
- error = accrueInterest() (contracts/CToken.sol#851)
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(TokenErrorReporter.Error(error),FailureInfo.REPAY_BEHALF_ACCRUE_INTEREST_FAILED,0) (contracts/CToken.sol#854)
Reentrancy in CToken.repayBorrowBehalfInternal(address,uint256) (contracts/CToken.sol#850-858):
External calls:
- error = accrueInterest() (contracts/CToken.sol#851)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
- repayBorrowFresh(msg.sender,borrower,repayAmount) (contracts/CToken.sol#857)
- allowed = controller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#888)
- nonReentrant(false) (contracts/CToken.sol#850)
- controller_afterNonReentrant() (contracts/CToken.sol#1582)
- controller_beforeNonReentrant() (contracts/CToken.sol#1574)
Event emitted after the call(s):
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - repayBorrowFresh(msg.sender,borrower,repayAmount) (contracts/CToken.sol#857)
- Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#230)
  - repayBorrowFresh(msg.sender,borrower,repayAmount) (contracts/CToken.sol#857)
- RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/CToken.sol#938)
- repayBorrowFresh(msg.sender,borrower,repayAmount) (contracts/CToken.sol#857)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#878-945):
External calls:
- allowed = controller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#888)
Event emitted after the call(s):
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(Error.MARKET_NOT_FRESH,FailureInfo.REPAY_BORROW_FRESHNESS_CHECK,0) (contracts/CToken.sol#887)
- Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#230)
  - failOpaque(Error.MATH_ERROR,FailureInfo.REPAY_BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr),0) (contracts/CToken.sol#898)
- Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#230)
  - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.REPAY_BORROW_COMPPTROLLER_REJECTION,allowed,0) (contracts/CToken.sol#882)
- RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/CToken.sol#938)
Reentrancy in CToken.repayBorrowInternal(uint256) (contracts/CToken.sol#834-842):
External calls:
- error = accrueInterest() (contracts/CToken.sol#835)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
- nonReentrant(false) (contracts/CToken.sol#834)
- controller_afterNonReentrant() (contracts/CToken.sol#1582)
- controller_beforeNonReentrant() (contracts/CToken.sol#1574)
Event emitted after the call(s):
- AccrueInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/CToken.sol#478)
- error = accrueInterest() (contracts/CToken.sol#835)
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(TokenErrorReporter.Error(error),FailureInfo.REPAY_BORROW_ACCRUE_INTEREST_FAILED,0) (contracts/CToken.sol#838)
Reentrancy in CToken.repayBorrowInternal(uint256) (contracts/CToken.sol#834-842):
External calls:
- error = accrueInterest() (contracts/CToken.sol#835)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
- repayBorrowFresh(msg.sender,msg.sender,repayAmount) (contracts/CToken.sol#841)
- allowed = controller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#888)
- nonReentrant(false) (contracts/CToken.sol#834)
- controller_afterNonReentrant() (contracts/CToken.sol#1582)
- controller_beforeNonReentrant() (contracts/CToken.sol#1574)
Event emitted after the call(s):
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - repayBorrowFresh(msg.sender,msg.sender,repayAmount) (contracts/CToken.sol#841)
- Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#230)
  - repayBorrowFresh(msg.sender,msg.sender,repayAmount) (contracts/CToken.sol#841)
- RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/CToken.sol#938)
- repayBorrowFresh(msg.sender,msg.sender,repayAmount) (contracts/CToken.sol#841)
Reentrancy in CToken.repay(address,address,uint256) (contracts/CToken.sol#1861-1863):

```

```

External calls:
- seizeInternal(msg.sender, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1862)
- allowed = controller.seizeAllowed(address(this), seizeToken, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1889)
- nonReentrant(true) (contracts/CToken.sol#1863)
- controller_afterNonReentrant() (contracts/CToken.sol#1582)
- controller_beforeNonReentrant() (contracts/CToken.sol#1574)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#228)
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#226)
- Failure(uint256(err), uint256(info), 0) (contracts/CToken.sol#1862)
- seizeInternal(msg.sender, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1148)
- ReserveAdded(address(this), vars.protocolSeizeAmount, vars.totalReservesNew) (contracts/CToken.sol#1862)
- Transfer(borrower, liquidator, vars.liquidatorSeizeTokens) (contracts/CToken.sol#1138)
- Transfer(borrower, address(this), vars.protocolSeizeTokens) (contracts/CToken.sol#1862)
- Transfer(borrower, address(this), vars.protocolSeizeTokens) (contracts/CToken.sol#1139)
- seizeInternal(msg.sender, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1862)
Reentrancy in CToken: totalBorrowOvershoot() (contracts/CToken.sol#187-117):
External calls:
- allowed = controller.seizeAllowed(address(this), seizeToken, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1889)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#228)
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#226)
- Failure(uint256(err), uint256(info), 0) (contracts/CToken.sol#1862)
- Failure(Error.INVALID_ACCOUNT_PAIR, FailureInfo.LIQUIDATE_SEIZE_LIQUIDATOR_IS_BORROWER) (contracts/CToken.sol#1896)
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#228)
- Failure(Error.INVALID_ACCOUNT_PAIR, FailureInfo.LIQUIDATE_SEIZE_LIQUIDATOR_IS_BORROWER) (contracts/CToken.sol#1896)
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#228)
- Failure(Error.INVALID_ACCOUNT_PAIR, FailureInfo.LIQUIDATE_SEIZE_COMPROLLER_REJECTION_ALLOWED) (contracts/CToken.sol#1891)
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#228)
- ReserveAdded(address(this), vars.protocolSeizeAmount, vars.totalReservesNew) (contracts/CToken.sol#1148)
- Transfer(borrower, liquidator, vars.liquidatorSeizeTokens) (contracts/CToken.sol#1138)
- Transfer(borrower, address(this), vars.protocolSeizeTokens) (contracts/CToken.sol#1139)
Reentrancy in CToken: totalBorrowOvershoot() (contracts/CToken.sol#187-281):
External calls:
- require(bool, string)(accruedInterest() == uint256(Error.NO_ERROR), accrued interest failed) (contracts/CToken.sol#279)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#673)
- nonReentrant(false) (contracts/CToken.sol#278)
- controller_afterNonReentrant() (contracts/CToken.sol#1582)
- controller_beforeNonReentrant() (contracts/CToken.sol#1574)
Event emitted after the call(s):
- AccruedInterest(bool, string, InterestRateModel, borrowIndexNew, totalBorrowNew) (contracts/CToken.sol#478)
- require(bool, string)(accruedInterest() == uint256(Error.NO_ERROR), accrued interest failed) (contracts/CToken.sol#279)
Reentrancy in CToken: transfer(address, uint256) (contracts/CToken.sol#162-144):
External calls:
- transferTokens(msg.sender, msg.sender, dst, amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#163)
- allowed = controller.transferAllowed(address(this), src, dst, tokens) (contracts/CToken.sol#95)
- nonReentrant(false) (contracts/CToken.sol#162)
- controller_afterNonReentrant() (contracts/CToken.sol#1582)
- controller_beforeNonReentrant() (contracts/CToken.sol#1574)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- TransferTokens(msg.sender, msg.sender, dst, amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#163)
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#228)
- Transfer(src, dst, tokens) (contracts/CToken.sol#147)
- TransferTokens(msg.sender, msg.sender, dst, amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#163)
Reentrancy in CToken: transferFrom(address, address, uint256) (contracts/CToken.sol#173-176):
External calls:
- allowed = controller.transferAllowed(address(this), src, dst, tokens) (contracts/CToken.sol#95)
- nonReentrant(false) (contracts/CToken.sol#173)
- controller_afterNonReentrant() (contracts/CToken.sol#1582)
- controller_beforeNonReentrant() (contracts/CToken.sol#1574)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- TransferTokens(msg.sender, src, dst, amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#174)
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#228)
- TransferTokens(msg.sender, src, dst, amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#174)
- Transfer(src, dst, tokens) (contracts/CToken.sol#147)
Reentrancy in CToken: transferTokens(address, address, uint256) (contracts/CToken.sol#93-154):
External calls:
- allowed = controller.transferAllowed(address(this), src, dst, tokens) (contracts/CToken.sol#95)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- Failure(Error.MATH_ERROR, FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/CToken.sol#116)
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#228)
- Failure(Error.COMPTROLLER_REJECTION, FailureInfo.TRANSFER_COMPROLLER_REJECTION_ALLOWED) (contracts/CToken.sol#97)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- Failure(Error.MATH_ERROR, FailureInfo.TRANSFER_TOO_MUCH) (contracts/CToken.sol#131)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- Failure(Error.MATH_ERROR, FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/CToken.sol#121)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- Failure(Error.BAD_INPUT, FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/CToken.sol#102)
- Transfer(src, dst, tokens) (contracts/CToken.sol#147)
Reference: https://github.com/crytic/allther/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
CToken._functionCall(address, bytes, string) (contracts/CToken.sol#1696-1615) uses assembly
- inline asm (contracts/CToken.sol#148-148)
Reference: https://github.com/crytic/allther/wiki/Detector-Documentation#assembly-usage
CToken.doTransferIn(address, uint256) (contracts/CToken.sol#1563) is never used and should be removed
CToken.doTransferOut(address, uint256) (contracts/CToken.sol#1558) is never used and should be removed
CToken.getCashForError() (contracts/CToken.sol#1537) is never used and should be removed
ControllerErrorReporter.fail(ControllerErrorReporter.Error, ControllerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#66-73) is never used and should be removed
ControllerErrorReporter.failOpaque(ControllerErrorReporter.Error, ControllerErrorReporter.FailureInfo, uint256) (contracts/ErrorReporter.sol#78-82) is never used and should be removed
Exponential.addExp(ExponentialNoError.Exp, ExponentialNoError.Exp) (contracts/Exponential.sol#47-51) is never used and should be removed
Exponential.divExp(ExponentialNoError.Exp, ExponentialNoError.Exp) (contracts/Exponential.sol#108-152) 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.mulExp(ExponentialNoError.Exp, ExponentialNoError.Exp) (contracts/Exponential.sol#130-155) is never used and should be removed
Exponential.mulExp(uint256, uint256) (contracts/Exponential.sol#156-152) is never used and should be removed
Exponential.mulExp3(ExponentialNoError.Exp, ExponentialNoError.Exp) (contracts/Exponential.sol#167-173) is never used and should be removed
Exponential.mulScalarTruncateAddInt(ExponentialNoError.Exp, uint256, uint256) (contracts/Exponential.sol#77-86) is never used and should be removed
Exponential.subExp(ExponentialNoError.Exp, ExponentialNoError.Exp) (contracts/Exponential.sol#146-48) is never used and should be removed
ExponentialNoError.add(ExponentialNoError.Double, ExponentialNoError.Double) (contracts/ExponentialNoError.sol#91-93) is never used and should be removed
ExponentialNoError.add(ExponentialNoError.Exp, ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#87-89) is never used and should be removed
ExponentialNoError.add(ExponentialNoError.Double, ExponentialNoError.Double) (contracts/ExponentialNoError.sol#71-73) is never used and should be removed
ExponentialNoError.div(ExponentialNoError.Double, uint256) (contracts/ExponentialNoError.sol#175-177) is never used and should be removed
ExponentialNoError.div(ExponentialNoError.Exp, ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#159-161) is never used and should be removed
ExponentialNoError.div(ExponentialNoError.Exp, uint256) (contracts/ExponentialNoError.sol#163-165) is never used and should be removed
ExponentialNoError.div(uint256, ExponentialNoError.Double) (contracts/ExponentialNoError.sol#179-181) is never used and should be removed
ExponentialNoError.div(uint256, ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#167-169) is never used and should be removed
ExponentialNoError.div(uint256, uint256) (contracts/ExponentialNoError.sol#183-185) is never used and should be removed
ExponentialNoError.div(uint256, uint256, string) (contracts/ExponentialNoError.sol#187-198) is never used and should be removed
ExponentialNoError.fracton(uint256, uint256) (contracts/ExponentialNoError.sol#192-194) is never used and should be removed
ExponentialNoError.greaterThanExp(ExponentialNoError.Exp, ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#66-68) is never used and should be removed
ExponentialNoError.isZeroExp(ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#73-75) is never used and should be removed
ExponentialNoError.lessThanExp(ExponentialNoError.Exp, ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#52-54) is never used and should be removed
ExponentialNoError.lessThanEqualExp(ExponentialNoError.Exp, ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#69-71) is never used and should be removed
ExponentialNoError.mul(ExponentialNoError.Double, ExponentialNoError.Double) (contracts/ExponentialNoError.sol#134-136) is never used and should be removed
ExponentialNoError.mul(ExponentialNoError.Exp, ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#138-140) is never used and should be removed
ExponentialNoError.mul(ExponentialNoError.Exp, ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#122-124) is never used and should be removed
ExponentialNoError.mul(uint256, ExponentialNoError.Double) (contracts/ExponentialNoError.sol#142-144) is never used and should be removed
ExponentialNoError.safediv(uint256, string) (contracts/ExponentialNoError.sol#77-88) is never used and should be removed
ExponentialNoError.safediv(uint256, string) (contracts/ExponentialNoError.sol#82-88) is never used and should be removed
ExponentialNoError.sub(ExponentialNoError.Double, ExponentialNoError.Double) (contracts/ExponentialNoError.sol#109-111) is never used and should be removed
ExponentialNoError.sub(ExponentialNoError.Exp, ExponentialNoError.Exp) (contracts/ExponentialNoError.sol#105-107) is never used and should be removed
UnitrollerAdminStorage.hasAdminRights() (contracts/ControllerStorage.sol#36-38) is never used and should be removed
Reference: https://github.com/crytic/allther/wiki/Detector-Documentation#dead-code
Low level call in CToken: finishInterestAccrual(uint256, uint256, uint256, uint256) (contracts/CToken.sol#437-476):
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#673)
Low level call in CToken: _setInterestRateModelFresh(InterestRateModel) (contracts/CToken.sol#479-512):
- address(oldInterestRateModel).call(abi.encodeWithSignature(resetInterestCheckpoints())) (contracts/CToken.sol#506)
- address(newInterestRateModel).call(abi.encodeWithSignature(checkpointInterest())) (contracts/CToken.sol#509)
Low level call in CToken: _functionCall(address, bytes, string) (contracts/CToken.sol#1596-1615):
- (success, returnData) = target.call(data) (contracts/CToken.sol#1597)
Reference: https://github.com/crytic/allther/wiki/Detector-Documentation#low-level-calls
Function CERC20_delegateCompileOf(address) (contracts/CERC20.sol#177-180) is not in mixedCase
Function CERC20_delegate_becomeImplementation(bytes) (contracts/CERC20_delegate.sol#20-30) is not in mixedCase
Function CERC20_delegate_setImplementationOf(address, bool, bytes) (contracts/CERC20_delegate.sol#74-80) is not in mixedCase
Function CERC20_delegate_prepare() (contracts/CERC20_delegate.sol#86-91) is not in mixedCase
Function CToken_setAdminFee(uint256) (contracts/CToken.sol#1176-1184) is not in mixedCase

```

```

Function CToken_setReserveFactor(uint256) (contracts/CToken.sol#1241-1249) is not in mixedCase
Function CToken_reduceReserves(uint256) (contracts/CToken.sol#1285-1293) is not in mixedCase
Function CToken_withdrawAdminFees(uint256) (contracts/CToken.sol#1346-1356) is not in mixedCase
Function CToken_withdrawAdminFees(uint256) (contracts/CToken.sol#1404-1412) is not in mixedCase
Function CToken_getInterestRateModel(InterestRateModel) (contracts/CToken.sol#1403-1471) is not in mixedCase
Function CToken_setNameAndSymbol(string,string) (contracts/CToken.sol#1521-1528) is not in mixedCase
Parameter CToken_setNameAndSymbol(string,string).name (contracts/CToken.sol#1521) is not in mixedCase
Parameter CToken_setNameAndSymbol(string,string).symbol (contracts/CToken.sol#1521) is not in mixedCase
Function CTokenAdminStorage_fuseAdmin (contracts/CTokenInterfaces.sol#19) is not in UPPER_CASE_WITH_UNDERSCORES
Variable CTokenStorage_notEntered (contracts/CTokenInterfaces.sol#19) is not in mixedCase
Constant CTokenStorage.borrowRateMaxMantissa (contracts/CTokenInterfaces.sol#39) is not in UPPER_CASE_WITH_UNDERSCORES
Constant CTokenStorage.reserveFactorPlusFeeMantissa (contracts/CTokenInterfaces.sol#44) is not in UPPER_CASE_WITH_UNDERSCORES
Variable CTokenStorage_pendingAdmin (contracts/CTokenInterfaces.sol#49) is not in mixedCase
Constant CTokenStorage_protocolSizeShareMantissa (contracts/CTokenInterfaces.sol#160) is not in UPPER_CASE_WITH_UNDERSCORES
Function CTokenInterface_setReserveFactor(uint256) (contracts/CTokenInterfaces.sol#249) is not in mixedCase
Function CTokenInterface_setInterestRateModel(InterestRateModel) (contracts/CTokenInterfaces.sol#269) is not in mixedCase
Constant CTokenInterface_isCether (contracts/CTokenInterfaces.sol#153) is not in UPPER_CASE_WITH_UNDERSCORES
Constant CTokenInterface_isCether (contracts/CTokenInterfaces.sol#156) is not in UPPER_CASE_WITH_UNDERSCORES
Constant CetherInterface_isCether (contracts/CTokenInterfaces.sol#297) is not in UPPER_CASE_WITH_UNDERSCORES
Function CDelegateInterface_setImplementation(bytes,address,book,bytes) (contracts/CTokenInterfaces.sol#319) is not in mixedCase
Function CDelegateInterface_becomeImplementation(bytes) (contracts/CTokenInterfaces.sol#326) is not in mixedCase
Function CDelegateInterface_prepare() (contracts/CTokenInterfaces.sol#332) is not in mixedCase
Function ComptrollerInterface_beforeReentrant() (contracts/ComptrollerInterface.sol#76) is not in mixedCase
Function ComptrollerInterface_afterNonReentrant() (contracts/ComptrollerInterface.sol#77) is not in mixedCase
Constant ComptrollerInterface_isComptroller (contracts/ComptrollerInterface.sol#81) is not in UPPER_CASE_WITH_UNDERSCORES
Constant UnrollerAdminStorage_fuseAdmin (contracts/ComptrollerStorage.sol#11) is not in UPPER_CASE_WITH_UNDERSCORES
Variable ComptrollerV2Storage_minGuardianPaused (contracts/ComptrollerStorage.sol#145) is not in mixedCase
Variable ComptrollerV2Storage_borrowGuardianPaused (contracts/ComptrollerStorage.sol#146) is not in mixedCase
Variable ComptrollerV2Storage_notEntered (contracts/ComptrollerStorage.sol#172) is not in mixedCase
Variable ComptrollerV2Storage_notEnteredInitialized (contracts/ComptrollerStorage.sol#175) is not in mixedCase
Function ExponentialNoError_mul_scalarFuncate(ExponentialNoError.Exp,uint256) (contracts/ExponentialNoError.sol#88-99) is not in mixedCase
Function ExponentialNoError_mul_scalarFuncateAddIn(ExponentialNoError.Exp,uint256,uint256) (contracts/ExponentialNoError.sol#44-47) is not in mixedCase
Constant ExponentialNoError_expScale (contracts/ExponentialNoError.sol#11) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError_doubleScale (contracts/ExponentialNoError.sol#22) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError_halfExpScale (contracts/ExponentialNoError.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError_mantissaOne (contracts/ExponentialNoError.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant InterestRateModel_isInterestRateModel (contracts/InterestRateModel.sol#9) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PriceOracle_isPriceOracle (contracts/PriceOracle.sol#7) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "data (contracts/Cerc20Delegate.sol#22)" in Cerc20Delegate (contracts/Cerc20Delegate.sol#18-92)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#redundant-statements

Variable CToken_seize(address,address,uint256).seizeTokens (contracts/CToken.sol#1041) is too similar to CToken_seizeInternal(address,address,uint256).seizeTokens (contracts/CToken.sol#1087)
Variable CToken_liquidateBorrowFresh(address,address,uint256,CTokenInterface).seizeTokens (contracts/CToken.sol#1025) is too similar to CToken_seizeInternal(address,address,uint256).seizeTokens (contracts/CToken.sol#1087)
Variable CToken_seizeInternal(address,address,uint256).seizeTokens (contracts/CToken.sol#1087) is too similar to CToken_seizeInternal(address,address,uint256).seizeTokens (contracts/CToken.sol#1087)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#variable-names-are-too-similar

CTokenStorage_pendingAdmin (contracts/CTokenInterfaces.sol#49) is never used in Cerc20Delegate (contracts/Cerc20Delegate.sol#18-92)
ComptrollerV3Storage_maxAssets (contracts/ComptrollerStorage.sol#70) is never used in ComptrollerV3Storage (contracts/ComptrollerStorage.sol#153-176)
ComptrollerV2Storage_borrowerIndex (contracts/ComptrollerStorage.sol#111) is never used in ComptrollerV3Storage (contracts/ComptrollerStorage.sol#153-176)
ComptrollerV2Storage_maxAssets (contracts/ComptrollerStorage.sol#117) is never used in ComptrollerV3Storage (contracts/ComptrollerStorage.sol#153-176)
ComptrollerV2Storage_whitelistIndexes (contracts/ComptrollerStorage.sol#137) is never used in ComptrollerV3Storage (contracts/ComptrollerStorage.sol#153-176)
ComptrollerV3Storage_notEntered (contracts/ComptrollerStorage.sol#172) is never used in ComptrollerV3Storage (contracts/ComptrollerStorage.sol#153-176)
ComptrollerV3Storage_notEnteredInitialized (contracts/ComptrollerStorage.sol#175) is never used in ComptrollerV3Storage (contracts/ComptrollerStorage.sol#153-176)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#unused-state-variable

CTokenStorage_pendingAdmin (contracts/CTokenInterfaces.sol#49) should be constant
ComptrollerV3Storage_closeFactorMantissa (contracts/ComptrollerStorage.sol#60) should be constant
ComptrollerV3Storage_liquidationIncentiveMantissa (contracts/ComptrollerStorage.sol#65) should be constant
ComptrollerV2Storage_maxAssets (contracts/ComptrollerStorage.sol#70) should be constant
ComptrollerV2Storage_borrowGuardianPaused (contracts/ComptrollerStorage.sol#146) should be constant
ComptrollerV2Storage_minGuardianPaused (contracts/ComptrollerStorage.sol#145) should be constant
ComptrollerV2Storage_enforceWhitelist (contracts/ComptrollerStorage.sol#128) should be constant
ComptrollerV2Storage_pauseGuardian (contracts/ComptrollerStorage.sol#144) should be constant
ComptrollerV2Storage_seizeGuardianPaused (contracts/ComptrollerStorage.sol#148) should be constant
ComptrollerV2Storage_transferGuardianPaused (contracts/ComptrollerStorage.sol#147) should be constant
ComptrollerV3Storage_notEntered (contracts/ComptrollerStorage.sol#172) should be constant
ComptrollerV3Storage_notEnteredInitialized (contracts/ComptrollerStorage.sol#175) should be constant
ComptrollerV3Storage_autoImplementation (contracts/ComptrollerStorage.sol#157) should be constant
ComptrollerV3Storage_borrowCapGuardian (contracts/ComptrollerStorage.sol#160) should be constant
UnrollerAdminStorage_admin (contracts/ComptrollerStorage.sol#16) should be constant
UnrollerAdminStorage_adminRights (contracts/ComptrollerStorage.sol#21) should be constant
UnrollerAdminStorage_comptrollerImplementation (contracts/ComptrollerStorage.sol#43) should be constant
UnrollerAdminStorage_fuseAdminRights (contracts/ComptrollerStorage.sol#26) should be constant
UnrollerAdminStorage_pendingAdmin (contracts/ComptrollerStorage.sol#21) should be constant
UnrollerAdminStorage_pendingComptrollerImplementation (contracts/ComptrollerStorage.sol#48) should be constant
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

initialize(address,ComptrollerInterface,InterestRateModel,string,string,uint256,uint256) should be declared external:
- Cerc20_initialize(address,ComptrollerInterface,InterestRateModel,string,string,uint256,uint256) (contracts/Cerc20.sol#24-39)
- setInterestRateModel(InterestRateModel) should be declared external:
  - CToken_setInterestRateModel(InterestRateModel) (contracts/CToken.sol#1463-1471)
  - CTokenInterface_setInterestRateModel(InterestRateModel) (contracts/CTokenInterfaces.sol#269)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#public-function-that-could-be-declared-external

```

CEtherDelegate.sol

```

UnitrollerAdminStorage.admin (contracts/ComptrollerStorage.sol#16) is never initialized, it is used in:
- UnitrollerAdminStorage.hasAdminRights() (contracts/ComptrollerStorage.sol#26-38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

CEtherDelegate._becomeImplementation(bytes) (contracts/CEtherDelegate.sol#20-30) uses a Boolean constant improperly:
- false (contracts/CEtherDelegate.sol#25)
CEtherDelegate._resignImplementation() (contracts/CEtherDelegate.sol#35-40) uses a Boolean constant improperly:
- false (contracts/CEtherDelegate.sol#37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#misuse-of-a-boolean-constant

EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#68-70) has incorrect ERC20 function interface: EIP20NonStandardInterface.transfer(address,uint256) (contracts/EIP20NonStandardInterface.sol#34)
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#68-70) has incorrect ERC20 function interface: EIP20NonStandardInterface.transferFrom(address,address,uint256) (contracts/EIP20NonStandardInterface.sol#34)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

CToken accrueInterest() (contracts/CToken.sol#411-432) uses a dangerous strict equality:
- accrualBlockNumber == currentBlockNumber (contracts/CToken.sol#416)
CToken accrueInterest() (contracts/CToken.sol#411-432) uses a dangerous strict equality:
- require(bool, string)(mathErr == MathError.NO_ERROR, could not calculate block delta) (contracts/CToken.sol#429)
CToken.balanceOfUnderlying(address) (contracts/CToken.sol#217-222) uses a dangerous strict equality:
- require(bool, string)(err == MathError.NO_ERROR, balance could not be calculated) (contracts/CToken.sol#220)
CToken.borrowBalanceStored(address) (contracts/CToken.sol#278-302) uses a dangerous strict equality:
- require(bool, string)(err == MathError.NO_ERROR, borrowBalanceStored: borrowBalanceStoredInternal failed) (contracts/CToken.sol#300)
CarefulMath.div(uint256,uint256) (contracts/CarefulMath.sol#42-47) uses a dangerous strict equality:
- b == 0 (contracts/CarefulMath.sol#42)
CEther.doTransferIn(address,uint256) (contracts/CEther.sol#129-134) uses a dangerous strict equality:
- require(bool, string)(msg.value == amount, value mismatch) (contracts/CEther.sol#132)
CToken.exchangeRateStored() (contracts/CToken.sol#355-359) uses a dangerous strict equality:
- require(bool, string)(err == MathError.NO_ERROR, exchangeRateStored: exchangeRateStoredInternal failed) (contracts/CToken.sol#357)
CToken.exchangeRateStoredInternal() (contracts/CToken.sol#366-396) uses a dangerous strict equality:
- _totalSupply == 0 (contracts/CToken.sol#368)
CEther.getCashPrior() (contracts/CEther.sol#117-122) uses a dangerous strict equality:
- require(bool, string)(err == MathError.NO_ERROR) (contracts/CEther.sol#119)
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,uint8,uint256,uint256) (contracts/CToken.sol#34-78) uses a dangerous strict equality:
- require(bool, string)(accrualBlockNumber == 0 && borrowIndex == 0, initialized-only-once) (contracts/CToken.sol#44)
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,uint8,uint256,uint256) (contracts/CToken.sol#34-78) uses a dangerous strict equality:
- require(bool, string)(err == uint256(Error.NO_ERROR), setting irm failed) (contracts/CToken.sol#49)
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,uint8,uint256,uint256) (contracts/CToken.sol#34-78) uses a dangerous strict equality:
- require(bool, string)(err == uint256(Error.NO_ERROR), setting admin fee failed) (contracts/CToken.sol#71)
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,uint8,uint256,uint256) (contracts/CToken.sol#34-78) uses a dangerous strict equality:
- require(bool, string)(err == uint256(Error.NO_ERROR), token seizure failed) (contracts/CToken.sol#82)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#981-1050) uses a dangerous strict equality:
- require(bool, string)(amountSeizeError == uint256(Error.NO_ERROR), LIQUIDATE_COMPROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/CToken.sol#1026)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#981-1050) uses a dangerous strict equality:
- require(bool, string)(err == uint256(Error.NO_ERROR), MINT_EXCHANGE_CALCULATION_FAILED) (contracts/CToken.sol#957)
Exponential.mulExp(ExponentialNoError,ExponentialNoError) (contracts/Exponential.sol#135-155) uses a dangerous strict equality:
- assert(bool)(err == MathError.NO_ERROR) (contracts/Exponential.sol#152)
CarefulMath.mul(uint256,uint256) (contracts/CarefulMath.sol#24-36) uses a dangerous strict equality:
- a == 0 (contracts/CarefulMath.sol#25)
ExponentialNoError.mul(uint256,uint256,string) (contracts/ExponentialNoError.sol#150-157) uses a dangerous strict equality:
- a == 0 || b == 0 (contracts/ExponentialNoError.sol#151)
ExponentialNoError.mul(uint256,uint256,string) (contracts/ExponentialNoError.sol#150-157) uses a dangerous strict equality:
- require(bool, string)(c / a == b, errorMessage) (contracts/ExponentialNoError.sol#156)
CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#878-945) uses a dangerous strict equality:
- require(bool, string)(vars.mathErr == MathError.NO_ERROR, REPAY_BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED) (contracts/CToken.sol#927)
CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#878-945) uses a dangerous strict equality:
- require(bool, string)(vars.mathErr == MathError.NO_ERROR, REPAY_BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED) (contracts/CToken.sol#930)
CEther.requireNoError(uint256,string) (contracts/CEther.sol#124-144) uses a dangerous strict equality:
- errCode == uint256(Error.NO_ERROR) (contracts/CEther.sol#144)
CEther.requireNoError(uint256,string) (contracts/CEther.sol#124-144) uses a dangerous strict equality:
- require(bool, string)(err == uint256(Error.NO_ERROR), string(fullMessage)) (contracts/CEther.sol#143)
CToken.seizeInternal(address,address,uint256) (contracts/CToken.sol#1087-1147) uses a dangerous strict equality:
- require(bool, string)(vars.mathErr == MathError.NO_ERROR, exchange rate math error) (contracts/CToken.sol#1115)
CToken.transfer(address,uint256) (contracts/CToken.sol#152-164) uses a dangerous strict equality:
- transferTokens(msg.sender,msg.sender,dst,amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#163)
CToken.transfer(msg.sender,msg.sender,dst,amount) (contracts/CToken.sol#152-164) uses a dangerous strict equality:
- transferTokens(msg.sender,msg.sender,dst,amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#164)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in CToken_beforeNonReentrant(bool) (contracts/CToken.sol#1569-1573):
  External calls:
  - comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
  State variables written after the call(s):
  - _notEntered = false (contracts/CToken.sol#1572)
Reentrancy in CToken_reduceReserves(uint256) (contracts/CToken.sol#1286-1293):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#1286)
  - address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#1286)
  - comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - comptroller_afterNonReentrant() (contracts/CToken.sol#1582)
  State variables written after the call(s):
  - _reduceReservesFresh(reduceAmount) (contracts/CToken.sol#1292)
  - totalReserves = totalReservesNew (contracts/CToken.sol#1333)
Reentrancy in CToken_setAdminFee(uint256) (contracts/CToken.sol#1276-1284):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#1277)
  - address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#1276)
  - comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - comptroller_afterNonReentrant() (contracts/CToken.sol#1582)
  State variables written after the call(s):
  - _setAdminFeeFresh(newAdminFeeMantissa) (contracts/CToken.sol#1283)
  - adminMantissa = newAdminFeeMantissa (contracts/CToken.sol#1217)
  - _setAdminFeeFresh(newAdminFeeMantissa) (contracts/CToken.sol#1283)
  - fuseFeeMantissa = newFuseFeeMantissa (contracts/CToken.sol#1227)
Reentrancy in CToken_setInterestRateModel(InterestRateModel) (contracts/CToken.sol#1463-1471):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#1464)
  - address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - _setInterestRateModelFresh(newInterestRateModel) (contracts/CToken.sol#1470)
  - address(InterestRateModel).call(abi.encodeWithSignature(reserveInterestCheckpoints())) (contracts/CToken.sol#1506)
  - address(newInterestRateModel).call(abi.encodeWithSignature(checkpointInterest())) (contracts/CToken.sol#1509)
  State variables written after the call(s):
  - _setInterestRateModelFresh(newInterestRateModel) (contracts/CToken.sol#1470)
  - interestRateModel = newInterestRateModel (contracts/CToken.sol#1508)
Reentrancy in CToken_setReserveFactor(uint256) (contracts/CToken.sol#1241-1249):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#1242)
  - address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#1241)
  - comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - comptroller_afterNonReentrant() (contracts/CToken.sol#1582)
  State variables written after the call(s):
  - _setReserveFactorFresh(newReserveFactorMantissa) (contracts/CToken.sol#1248)
  - reserveFactorMantissa = newReserveFactorMantissa (contracts/CToken.sol#1273)
Reentrancy in CToken_withdrawAdminFee(uint256) (contracts/CToken.sol#1404-1412):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#1405)
  - address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#1404)
  - comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - comptroller_afterNonReentrant() (contracts/CToken.sol#1582)
  State variables written after the call(s):
  - _withdrawAdminFeeFresh(withdrawAmount) (contracts/CToken.sol#1411)
  - totalAdminFees = totalAdminFeesNew (contracts/CToken.sol#1409)
Reentrancy in CToken_withdrawFuseFees(uint256) (contracts/CToken.sol#1366-1369):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#1369)
  - address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#1366)
  - comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - comptroller_afterNonReentrant() (contracts/CToken.sol#1582)
  State variables written after the call(s):
  - _withdrawFuseFeesFresh(withdrawAmount) (contracts/CToken.sol#1365)
  - totalFuseFees = totalFuseFeesNew (contracts/CToken.sol#1391)
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#755-827):
  External calls:
  - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#757)
  - allowed = comptroller.borrowWithinLimits(address(this),vars.accountBorrowNew) (contracts/CToken.sol#792)
  State variables written after the call(s):
  - accountBorrow(borrower).principal = vars.accountBorrowNew (contracts/CToken.sol#815)

```

```

- accountBorrows[borrower].interestIndex = borrowIndex (contracts/CToken.sol#816)
Reentrancy in CToken.borrowInternal(uint256) (contracts/CToken.sol#733-741):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#734)
    address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
  - allowed = controller.borrowAllowed(address(this), borrower, borrowAmount) (contracts/CToken.sol#757)
    - allowed = controller.borrowWithinLimits(address(this), vars.accountBorrowNew) (contracts/CToken.sol#792)
  - nonReentrant(false) (contracts/CToken.sol#783)
    - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
    - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  State variables written after the call(s):
  - borrowFresh(msg.sender, borrowAmount) (contracts/CToken.sol#748)
  - totalBorrows = vars.totalBorrowNew (contracts/CToken.sol#817)
Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#955-978):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#956)
    address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
  - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#962)
  - liquidateBorrowFresh(msg.sender, borrower, repayAmount, cTokenCollateral) (contracts/CToken.sol#969)
    - allowed = controller.seizeAllowed(address(this), seizerToken, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1089)
    - allowed = controller.repayBorrowAllowed(address(this), payer, borrower, repayAmount) (contracts/CToken.sol#888)
    - allowed = controller.liquidateBorrowAllowed(address(this), address(cTokenCollateral), liquidator, borrower, repayAmount) (contracts/CToken.sol#983)
    - seizeError = cTokenCollateral.seizeLiquidator, borrower, seizeTokens) (contracts/CToken.sol#1036)
  - nonReentrant(false) (contracts/CToken.sol#955)
    - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
    - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  State variables written after the call(s):
  - liquidateBorrowFresh(msg.sender, borrower, repayAmount, cTokenCollateral) (contracts/CToken.sol#969)
  - totalBorrows = vars.totalBorrowNew (contracts/CToken.sol#935)
  - liquidateBorrowFresh(msg.sender, borrower, repayAmount, cTokenCollateral) (contracts/CToken.sol#969)
  - totalReserves = vars.totalReserveNew (contracts/CToken.sol#1132)
  - totalSupply = vars.totalSupplyNew (contracts/CToken.sol#551)
Reentrancy in CToken.nonReentrant(only) (contracts/CToken.sol#559):
  External calls:
  - _beforeNonReentrant(only) (contracts/CToken.sol#559)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - _afterNonReentrant(only) (contracts/CToken.sol#551)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  State variables written after the call(s):
  - _afterNonReentrant(only) (contracts/CToken.sol#551)
  - _notEntered = true (contracts/CToken.sol#1083)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#632-726):
  External calls:
  - allowed = controller.redeemAllowed(address(this), redeemer, vars.redeemTokens) (contracts/CToken.sol#672)
  State variables written after the call(s):
  - totalSupply = vars.totalSupplyNew (contracts/CToken.sol#715)
Reentrancy in CToken.repayBorrowInternal(address,uint256) (contracts/CToken.sol#850-858):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#851)
    address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
  - repayBorrowFresh(msg.sender, borrower, repayAmount) (contracts/CToken.sol#857)
    - allowed = controller.repayBorrowAllowed(address(this), payer, borrower, repayAmount) (contracts/CToken.sol#888)
  - nonReentrant(false) (contracts/CToken.sol#850)
    - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
    - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  State variables written after the call(s):
  - repayBorrowFresh(msg.sender, borrower, repayAmount) (contracts/CToken.sol#857)
  - totalBorrows = vars.totalBorrowNew (contracts/CToken.sol#935)
Reentrancy in CToken.repayBorrowInternal(uint256) (contracts/CToken.sol#834-842):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#835)
    address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
  - repayBorrowFresh(msg.sender, msg.sender, repayAmount) (contracts/CToken.sol#841)
    - allowed = controller.repayBorrowAllowed(address(this), payer, borrower, repayAmount) (contracts/CToken.sol#888)
  - nonReentrant(false) (contracts/CToken.sol#834)
    - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
    - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  State variables written after the call(s):
  - repayBorrowFresh(msg.sender, msg.sender, repayAmount) (contracts/CToken.sol#841)
  - totalBorrows = vars.totalBorrowNew (contracts/CToken.sol#935)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

CToken.finishInterestAccrual(uint256,uint256,uint256) (contracts/CToken.sol#437-476) ignores return value by address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMan
CToken._setInterestRateModelFresh(interestRateModel) (contracts/CToken.sol#1479-1512) ignores return value by address(oldInterestRateModel).call(abi.encodeWithSignature(setInterestCheckpoint(1))) (contracts/CToken
CToken._setInterestRateModelFresh(interestRateModel) (contracts/CToken.sol#1479-1512) ignores return value by address(newInterestRateModel).call(abi.encodeWithSignature(checkpointInterest(1))) (contracts/CToken.sol#
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#unchecked-low-level-calls

CToken.seizeInternal(address,address,uint256).vars (contracts/CToken.sol#1099) is a local variable never initialized
CToken.mintFresh(address,uint256).vars (contracts/CToken.sol#523) is a local variable never initialized
CToken.borrowFresh(address,uint256,uint256) (contracts/CToken.sol#774) is a local variable never initialized
CToken.redeemFresh(address,uint256,uint256).vars (contracts/CToken.sol#635) is a local variable never initialized
CToken.repayBorrowFresh(address,address,uint256).vars (contracts/CToken.sol#890) is a local variable never initialized
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#uninitialized-local-variables

Exponential.div(ScalarByExpFracCast(uint256,Exponential.NoError.Exp.fraction) (contracts/Exponential.sol#124) shadows:
  Exponential.NoError.fraction(uint256,uint256) (contracts/Exponential.NoError.sol#192-254) (function)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#local-variable-shadowing

Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#755-827):
  External calls:
  - allowed = controller.borrowAllowed(address(this), borrower, borrowAmount) (contracts/CToken.sol#757)
  - allowed = controller.borrowWithinLimits(address(this), vars.accountBorrowNew) (contracts/CToken.sol#792)
  State variables written after the call(s):
  - totalBorrows = vars.totalBorrowNew (contracts/CToken.sol#817)
Reentrancy in CToken.initialize(ControllerInterface,interestRateModel,uint256,string,string,uint8,uint256,uint256) (contracts/CToken.sol#94-75):
  External calls:
  - err = _setInterestRateModelFresh(interestRateModel) (contracts/CToken.sol#58)
    address(oldInterestRateModel).call(abi.encodeWithSignature(checkpointInterest(1))) (contracts/CToken.sol#1586)
    address(newInterestRateModel).call(abi.encodeWithSignature(checkpointInterest(1))) (contracts/CToken.sol#1589)
  State variables written after the call(s):
  - _notEntered = true (contracts/CToken.sol#74)
  - err = _setAdminFeeFresh(adminFeeMantissa) (contracts/CToken.sol#478)
  - _adminFeeMantissa = newAdminFeeMantissa (contracts/CToken.sol#1217)
  - decimals = decimals (contracts/CToken.sol#463)
  - err = _setAdminFeeFresh(adminFeeMantissa) (contracts/CToken.sol#478)
  - _adminFeeMantissa = newAdminFeeMantissa (contracts/CToken.sol#1227)
  - name = name (contracts/CToken.sol#461)
  - err = _setReserveFactorFresh(reserveFactorMantissa) (contracts/CToken.sol#464)
  - _reserveFactorMantissa = newReserveFactorMantissa (contracts/CToken.sol#1273)
  - symbol = symbol (contracts/CToken.sol#462)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#511-588):
  External calls:
  - allowed = controller.mintAllowed(address(this), minter, mintAmount) (contracts/CToken.sol#513)
  State variables written after the call(s):
  - accountTokens(minter) = vars.accountTokensNew (contracts/CToken.sol#578)
  - totalSupply = vars.totalSupplyNew (contracts/CToken.sol#609)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#632-726):
  External calls:
  - allowed = controller.redeemAllowed(address(this), redeemer, vars.redeemTokens) (contracts/CToken.sol#672)
  State variables written after the call(s):
  - accountTokens(redeemer) = vars.accountTokensNew (contracts/CToken.sol#716)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#878-945):
  External calls:
  - allowed = controller.repayBorrowAllowed(address(this), payer, borrower, repayAmount) (contracts/CToken.sol#888)
  State variables written after the call(s):
  - accountBorrows(borrower).principal = vars.accountBorrowNew (contracts/CToken.sol#933)
  - accountBorrows(borrower).interestIndex = borrowIndex (contracts/CToken.sol#934)
  - totalBorrows = vars.totalBorrowNew (contracts/CToken.sol#935)
Reentrancy in CToken.seizeInternal(address,address,uint256) (contracts/CToken.sol#1087-1147):
  External calls:
  - allowed = controller.seizeAllowed(address(this), seizerToken, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1089)
  State variables written after the call(s):
  - accountTokens(borrower) = vars.borrowerTokensNew (contracts/CToken.sol#1134)
  - accountTokens(liquidator) = vars.liquidatorTokensNew (contracts/CToken.sol#1135)
  - totalReserves = vars.totalReserveNew (contracts/CToken.sol#1132)
  - totalSupply = vars.totalSupplyNew (contracts/CToken.sol#1133)
Reentrancy in CToken.transferTokens(address,address,uint256) (contracts/CToken.sol#93-154):
  External calls:
  - allowed = controller.transferAllowed(address(this), src, dst, tokens) (contracts/CToken.sol#98)
  State variables written after the call(s):
  - accountTokens[src] = srcTokensNew (contracts/CToken.sol#138)
  - accountTokens[dst] = dstTokensNew (contracts/CToken.sol#139)
  - transferAllowed[src][dst] = allowanceNew (contracts/CToken.sol#143)

```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2>

```

Reentrancy in CToken_reduceReserves(uint256) (contracts/CToken.sol#1285-1293):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#1366)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#1285)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  Event emitted after the call(s):
  - AccrueInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/CToken.sol#470)
  - error = accrueInterest() (contracts/CToken.sol#1366)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
    - _reduceReservesFresh(reduceAmount) (contracts/CToken.sol#1292)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
    - fail(TokenErrorReporter.Error(error),FailureInfo.REDUCE_RESERVES_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#1289)
  - ReservesReduced(msg.sender,reduceAmount,totalReservesNew) (contracts/CToken.sol#1338)
  - _reduceReservesFresh(reduceAmount) (contracts/CToken.sol#1292)
Reentrancy in CToken_setAdminFee(uint256) (contracts/CToken.sol#1176-1184):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#1177)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#1176)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  Event emitted after the call(s):
  - AccrueInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/CToken.sol#470)
  - error = accrueInterest() (contracts/CToken.sol#1177)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
    - fail(TokenErrorReporter.Error(error),FailureInfo.SET_ADMIN_FEE_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#1180)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
    - _setAdminFeeFresh(newAdminFeeMantissa) (contracts/CToken.sol#1183)
  - NewAdminFee(oldAdminFeeMantissa,newAdminFeeMantissa) (contracts/CToken.sol#1220)
  - _setAdminFeeFresh(newAdminFeeMantissa) (contracts/CToken.sol#1183)
  - NewUserFee(oldUserFeeMantissa,newUserFeeMantissa) (contracts/CToken.sol#1238)
  - _setUserFeeFresh(newUserFeeMantissa) (contracts/CToken.sol#1183)
Reentrancy in CToken_delegate_setImplementationInternal(address,bool,bytes) (contracts/C EtherDelegate.sol#48-66):
  External calls:
  - _functionCall(address(this),abi.encodeWithSignature(implementationInternal(implementationData),implementationData),implementationData) (contracts/C EtherDelegate.sol#42)
  - (success,returnValue) = target.call(data) (contracts/CToken.sol#1597)
  Event emitted after the call(s):
  - NewImplementation(oldImplementation,implementation) (contracts/C EtherDelegate.sol#45)
Reentrancy in CToken_setInterestRateModel(InterestRateModel) (contracts/CToken.sol#1463-1471):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#1464)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  Event emitted after the call(s):
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(TokenErrorReporter.Error(error),FailureInfo.SET_INTEREST_RATE_MODEL_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#1467)
Reentrancy in CToken_getInterestRateModel(InterestRateModel) (contracts/CToken.sol#1463-1471):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#1464)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - _setInterestRateModelFresh(newInterestRateModel) (contracts/CToken.sol#1478)
  - address(oldInterestRateModel).call(abi.encodeWithSignature(resetInterestCheckpoint())) (contracts/CToken.sol#1506)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest())) (contracts/CToken.sol#1509)
  Event emitted after the call(s):
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - _setInterestRateModelFresh(newInterestRateModel) (contracts/CToken.sol#1478)
  - NewMarketInterestRateModel(oldInterestRateModel,newInterestRateModel) (contracts/CToken.sol#1503)
  - _setInterestRateModelFresh(newInterestRateModel) (contracts/CToken.sol#1478)
Reentrancy in CToken_getReserveFactor(uint256) (contracts/CToken.sol#1241-1249):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#1242)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#1241)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  Event emitted after the call(s):
  - AccrueInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/CToken.sol#470)
  - error = accrueInterest() (contracts/CToken.sol#1242)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
    - _setReserveFactorFresh(newReserveFactorMantissa) (contracts/CToken.sol#1248)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
    - fail(TokenErrorReporter.Error(error),FailureInfo.SET_RESERVE_FACTOR_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#1246)
  - NewReserveFactor(oldReserveFactorMantissa,newReserveFactorMantissa) (contracts/CToken.sol#1275)
  - _setReserveFactorFresh(newReserveFactorMantissa) (contracts/CToken.sol#1248)
Reentrancy in CToken_withdrawAdminFees(uint256) (contracts/CToken.sol#1404-1412):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#1405)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#1404)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  Event emitted after the call(s):
  - AccrueInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/CToken.sol#470)
  - error = accrueInterest() (contracts/CToken.sol#1405)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
    - _withdrawAdminFeesFresh(withdrawAmount) (contracts/CToken.sol#1411)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
    - fail(TokenErrorReporter.Error(error),FailureInfo.WITHDRAW_ADMIN_FEES_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#1408)
  - WithdrawAdminFeesFresh(withdrawAmount) (contracts/CToken.sol#1404)
Reentrancy in CToken_withdrawUserFees(uint256) (contracts/CToken.sol#1348-1356):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#1349)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#1348)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  Event emitted after the call(s):
  - AccrueInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/CToken.sol#470)
  - error = accrueInterest() (contracts/CToken.sol#1349)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
    - _withdrawUserFeesFresh(withdrawAmount) (contracts/CToken.sol#1355)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
    - fail(TokenErrorReporter.Error(error),FailureInfo.WITHDRAW_USER_FEES_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#1352)
  - WithdrawUserFeesFresh(withdrawAmount) (contracts/CToken.sol#1348)
Reentrancy in CToken_borrowBalanceCurrent(address) (contracts/CToken.sol#280-291):
  External calls:
  - require(bool,string)(accrueInterest() == uint256(Error.NO_ERROR),accrue interest failed) (contracts/CToken.sol#289)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#288)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  Event emitted after the call(s):
  - AccrueInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowNew) (contracts/CToken.sol#478)
  - require(bool,string)(accrueInterest() == uint256(Error.NO_ERROR),accrue interest failed) (contracts/CToken.sol#289)
Reentrancy in CToken_borrowFresh(address,uint256) (contracts/CToken.sol#755-827):
  External calls:
  - allowed = controller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#757)
  Event emitted after the call(s):
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(Error.NO_BORROWING_CAP,FailureInfo.BORROW_CAP_NOT_AVAILABLE) (contracts/CToken.sol#771)
  - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#226)
  - failOpaqueError(MATH_ERROR,FailureInfo.BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/CToken.sol#788)
  - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#226)
  - failOpaqueError(MATH_ERROR,FailureInfo.BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/CToken.sol#783)
  - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#226)
  - failOpaqueError(COMPTROLLER_REJECTION,FailureInfo.BORROW_COMPTROLLER_REJECTION,allowed) (contracts/CToken.sol#799)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(Error.MARKET_NOT_FRESH,FailureInfo.BORROW_FRESHNESS_CHECK) (contracts/CToken.sol#764)
Reentrancy in CToken_borrowFresh(address,uint256) (contracts/CToken.sol#755-827):
  External calls:
  - allowed = controller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#757)
  - allowed = controller.borrowWithinLimits(address(this),vars.accountBorrowsNew) (contracts/CToken.sol#792)
  Event emitted after the call(s):
  - Borrow(borrower,borrowAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/CToken.sol#820)
  - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#226)
  - failOpaqueError(MATH_ERROR,FailureInfo.BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/CToken.sol#799)
  - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#226)
  - failOpaqueError(COMPTROLLER_REJECTION,FailureInfo.BORROW_COMPTROLLER_REJECTION,allowed) (contracts/CToken.sol#794)
Reentrancy in CToken_borrowInternal(uint256) (contracts/CToken.sol#730-741):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#734)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#733)

```

```
- controller_beforeNonReentrant() (contracts/CToken.sol#1571)
- controller_afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- AccrueInterest(cashPrior, interestAccumulated, borrowIndexNew, totalBorrowsNew) (contracts/CToken.sol#470)
- error = accrueInterest() (contracts/CToken.sol#936)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(TokenErrorReporter.Error(error), FailureInfo.BORROW_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#737)
Reentrancy in CToken.borrowInternal(uint256) (contracts/CToken.sol#733-741):
External calls:
- error = accrueInterest() (contracts/CToken.sol#736)
- address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
- borrowFresh(msg.sender, borrowAmount) (contracts/CToken.sol#740)
- allowed = controller.borrowAllowed(address(this), borrower, borrowAmount) (contracts/CToken.sol#757)
- allowed = controller.borrowWithinLimits(address(this), vars.accountBorrowsNew) (contracts/CToken.sol#792)
- nonReentrant(false) (contracts/CToken.sol#753)
- controller_beforeNonReentrant() (contracts/CToken.sol#1571)
- controller_afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- Borrow(borrower, borrowAmount, vars.accountBorrowsNew, vars.totalBorrowsNew) (contracts/CToken.sol#820)
- borrowFresh(msg.sender, borrowAmount) (contracts/CToken.sol#740)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- borrowFresh(msg.sender, borrowAmount) (contracts/CToken.sol#740)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- borrowFresh(msg.sender, borrowAmount) (contracts/CToken.sol#740)
Reentrancy in CToken.exchangeRateCurrent() (contracts/CToken.sol#345-348):
External calls:
- require(bool, string)(accrueInterest() == uint256(Error.NO_ERROR), accrue interest failed) (contracts/CToken.sol#346)
- address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
- nonReentrant(false) (contracts/CToken.sol#345)
- controller_beforeNonReentrant() (contracts/CToken.sol#1571)
- controller_afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- AccrueInterest(cashPrior, interestAccumulated, borrowIndexNew, totalBorrowsNew) (contracts/CToken.sol#470)
- require(bool, string)(accrueInterest() == uint256(Error.NO_ERROR), accrue interest failed) (contracts/CToken.sol#346)
Reentrancy in CToken.initialize(ControllerInterface, interestRateModel, uint256, string, string, uint8, uint256, uint256) (contracts/CToken.sol#94-75):
External calls:
- err = _setInterestRateModelFresh(interestRateModel) (contracts/CToken.sol#58)
- address(interestRateModel).call(abi.encodeWithSignature(setInterestCheckpointInterest())) (contracts/CToken.sol#1506)
- address(newInterestRateModel).call(abi.encodeWithSignature(checkpointInterest())) (contracts/CToken.sol#1509)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- err = setReserveFactorFresh(reserveFactorMantissa) (contracts/CToken.sol#66)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- err = _setAdminFeeFresh(adminFeeMantissa) (contracts/CToken.sol#70)
- NewAdminFee(oldAdminFeeMantissa, newAdminFeeMantissa) (contracts/CToken.sol#220)
- err = _setAdminFeeFresh(adminFeeMantissa) (contracts/CToken.sol#70)
- NewFeeSet(oldAdminFeeMantissa, newAdminFeeMantissa) (contracts/CToken.sol#220)
- err = _setAdminFeeFresh(adminFeeMantissa) (contracts/CToken.sol#70)
- NewReserveFactor(oldReserveFactorMantissa, newReserveFactorMantissa) (contracts/CToken.sol#1275)
- err = _setReserveFactorFresh(reserveFactorMantissa) (contracts/CToken.sol#66)
Reentrancy in CToken.liquidateBorrowFresh(address, address, uint256, CTokenInterface) (contracts/CToken.sol#981-1050):
External calls:
- allowed = controller.liquidateBorrowAllowed(address(this), address(CTokenCollateral), liquidator, borrower, repayAmount) (contracts/CToken.sol#983)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(Error.INVALID_ACCOUNT_PAIR, FailureInfo.LIQUIDATE_LIQUIDATOR_IS_BORROWER, 0) (contracts/CToken.sol#1008)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(Error.MARKET_NOT_FRESH, FailureInfo.LIQUIDATE_FRESHNESS_CHECK, 0) (contracts/CToken.sol#998)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED, FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_ZERO, 0) (contracts/CToken.sol#1005)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(Error.MARKET_NOT_FRESH, FailureInfo.LIQUIDATE_COLLATERAL_FRESHNESS_CHECK, 0) (contracts/CToken.sol#995)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED, FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_UINT_MAX, 0) (contracts/CToken.sol#1010)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(DaemonError.COMPTROLLER_REJECTION, FailureInfo.LIQUIDATE_COMPTROLLER_REJECTION, allowed, 0) (contracts/CToken.sol#986)
Reentrancy in CToken.liquidateBorrowFresh(address, address, uint256, CTokenInterface) (contracts/CToken.sol#981-1050):
External calls:
- allowed = controller.liquidateBorrowAllowed(address(this), address(CTokenCollateral), liquidator, borrower, repayAmount) (contracts/CToken.sol#983)
- repayBorrowError.actualRepayAmount = repayBorrowFresh(liquidator, borrower, repayAmount) (contracts/CToken.sol#1015)
- allowed = controller.liquidateBorrowAllowed(address(this), payer, borrower, repayAmount) (contracts/CToken.sol#888)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(TokenErrorReporter.Error(repayBorrowError), FailureInfo.LIQUIDATE_REPAY_BORROW_FAILED, 0) (contracts/CToken.sol#1017)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- (repayBorrowError, actualRepayAmount) = repayBorrowFresh(liquidator, borrower, repayAmount) (contracts/CToken.sol#1015)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- (repayBorrowError, actualRepayAmount) = repayBorrowFresh(liquidator, borrower, repayAmount) (contracts/CToken.sol#1015)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- (repayBorrowError, actualRepayAmount) = repayBorrowFresh(liquidator, borrower, repayAmount) (contracts/CToken.sol#1015)
Reentrancy in CToken.liquidateBorrowFresh(address, address, uint256, CTokenInterface) (contracts/CToken.sol#981-1050):
External calls:
- allowed = controller.liquidateBorrowAllowed(address(this), address(CTokenCollateral), liquidator, borrower, repayAmount) (contracts/CToken.sol#983)
- (repayBorrowError, actualRepayAmount) = repayBorrowFresh(liquidator, borrower, repayAmount) (contracts/CToken.sol#1015)
- allowed = controller.liquidateBorrowAllowed(address(this), payer, borrower, repayAmount) (contracts/CToken.sol#888)
- seizeError = seizeInternal(address(this), liquidator, borrower, seizeTokens) (contracts/CToken.sol#1034)
- allowed = controller.seizeAllowed(address(this), seizeToken, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1089)
- seizeError = seizeInternal(address(this), liquidator, borrower, seizeTokens) (contracts/CToken.sol#1034)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- seizeError = seizeInternal(address(this), liquidator, borrower, seizeTokens) (contracts/CToken.sol#1034)
- ReserveAdded(address(this), vars.protocolSeizeAmount, vars.totalReservesNew) (contracts/CToken.sol#1148)
- seizeError = seizeInternal(address(this), liquidator, borrower, seizeTokens) (contracts/CToken.sol#1034)
- Transfer(borrower, liquidator, vars.liquidatorSeizeTokens) (contracts/CToken.sol#1138)
- seizeError = seizeInternal(address(this), liquidator, borrower, seizeTokens) (contracts/CToken.sol#1034)
- Transfer(borrower, address(this), vars.protocolSeizeTokens) (contracts/CToken.sol#1139)
- seizeError = seizeInternal(address(this), liquidator, borrower, seizeTokens) (contracts/CToken.sol#1034)
Reentrancy in CToken.liquidateBorrowFresh(address, address, uint256, CTokenInterface) (contracts/CToken.sol#981-1050):
External calls:
- allowed = controller.liquidateBorrowAllowed(address(this), address(CTokenCollateral), liquidator, borrower, repayAmount) (contracts/CToken.sol#983)
- (repayBorrowError, actualRepayAmount) = repayBorrowFresh(liquidator, borrower, repayAmount) (contracts/CToken.sol#1015)
- allowed = controller.liquidateBorrowAllowed(address(this), payer, borrower, repayAmount) (contracts/CToken.sol#888)
- seizeError = seizeInternal(address(this), liquidator, borrower, seizeTokens) (contracts/CToken.sol#1034)
- allowed = controller.seizeAllowed(address(this), seizeToken, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1089)
- seizeError = CTokenCollateral.seize(liquidator, borrower, seizeTokens) (contracts/CToken.sol#1036)
Event emitted after the call(s):
- LiquidateBorrow(liquidator, borrower, actualRepayAmount, address(CTokenCollateral), seizeTokens) (contracts/CToken.sol#1043)
Reentrancy in CToken.liquidateBorrowInternal(address, uint256, CTokenInterface) (contracts/CToken.sol#956-970):
External calls:
- error = accrueInterest() (contracts/CToken.sol#956)
- address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
- nonReentrant(false) (contracts/CToken.sol#956)
- controller_beforeNonReentrant() (contracts/CToken.sol#1571)
- controller_afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- AccrueInterest(cashPrior, interestAccumulated, borrowIndexNew, totalBorrowsNew) (contracts/CToken.sol#470)
- error = accrueInterest() (contracts/CToken.sol#956)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(TokenErrorReporter.Error(error), FailureInfo.LIQUIDATE_ACCRUE_BORROW_INTEREST_FAILED, 0) (contracts/CToken.sol#959)
Reentrancy in CToken.liquidateBorrowInternal(address, uint256, CTokenInterface) (contracts/CToken.sol#956-970):
External calls:
- error = accrueInterest() (contracts/CToken.sol#956)
- address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
- error = CTokenCollateral.seizeInternal(address(this), liquidator, borrower, seizeTokens) (contracts/CToken.sol#1034)
- nonReentrant(false) (contracts/CToken.sol#956)
- controller_beforeNonReentrant() (contracts/CToken.sol#1571)
- controller_afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(TokenErrorReporter.Error(error), FailureInfo.LIQUIDATE_COLLATERAL_INTEREST_FAILED, 0) (contracts/CToken.sol#965)
Reentrancy in CToken.liquidateBorrowInternal(address, uint256, CTokenInterface) (contracts/CToken.sol#956-970):
External calls:
- error = accrueInterest() (contracts/CToken.sol#956)
- address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
- error = CTokenCollateral.seizeAllowed(address(this), liquidator, borrower, repayAmount, CTokenCollateral) (contracts/CToken.sol#969)
- allowed = controller.seizeAllowed(address(this), seizeToken, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1089)
- allowed = controller.liquidateBorrowAllowed(address(this), payer, borrower, repayAmount) (contracts/CToken.sol#888)
- allowed = controller.liquidateBorrowAllowed(address(this), address(CTokenCollateral), liquidator, borrower, repayAmount) (contracts/CToken.sol#983)
- seizeError = CTokenCollateral.seize(liquidator, borrower, seizeTokens) (contracts/CToken.sol#1036)
- nonReentrant(false) (contracts/CToken.sol#956)
- controller_beforeNonReentrant() (contracts/CToken.sol#1571)
- controller_afterNonReentrant() (contracts/CToken.sol#1582)
```

```

Event emitted after the call(s):
- Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
  - LiquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#969)
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - LiquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#969)
- LiquidateBorrow(msg.sender,borrower,actualBorrowAmount,address(CTokenCollateral),assetTokens) (contracts/CToken.sol#1843)
  - LiquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#969)
- RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowNew,vars.totalBorrowNew) (contracts/CToken.sol#938)
  - LiquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#969)
- ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReserveNew) (contracts/CToken.sol#1140)
  - LiquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#969)
- Transfer(borrower,vars.liquidator,vars.liquidatorSeizeTokens) (contracts/CToken.sol#1139)
  - LiquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#969)
- Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/CToken.sol#1139)
  - LiquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#969)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#511-588):
  External calls:
  - allowed = controller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#513)
Event emitted after the call(s):
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(Error.MARKET_NOT_FRESH,FailureInfo.MINT_FRESHNESS_CHECK,0) (contracts/CToken.sol#520)
- Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
  - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.MINT_COMPTROLLER_REJECTION,allowed,0) (contracts/CToken.sol#616)
- Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
  - failOpaque(Error.MATH_ERROR,FailureInfo.MINT_EXCHANGE_RATE_READ_FAILED,uint256(vars.mathErr),0) (contracts/CToken.sol#527)
- Mint(minter,vars.actualMintAmount,vars.mintTokens) (contracts/CToken.sol#473)
- Transfer(address(this),minter,vars.mintTokens) (contracts/CToken.sol#474)
Reentrancy in CToken.mintInternal(uint256) (contracts/CToken.sol#486-492):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#486)
    - address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#486)
  - controller.beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller.afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- AccrueInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/CToken.sol#470)
  - error = accrueInterest() (contracts/CToken.sol#485)
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(TokenErrorReporter.Error(error),FailureInfo.MINT_ACCRUE_INTEREST_FAILED,0) (contracts/CToken.sol#488)
Reentrancy in CToken.mintInternal(uint256) (contracts/CToken.sol#486-492):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#486)
    - address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - mintFresh(msg.sender,mintAmount) (contracts/CToken.sol#491)
  - allowed = controller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#513)
  - controller.mintVerify(address(this),minter,vars.actualMintAmount,vars.mintTokens) (contracts/CToken.sol#577)
  - nonReentrant(false) (contracts/CToken.sol#486)
  - controller.beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller.afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
  - mintFresh(msg.sender,mintAmount) (contracts/CToken.sol#491)
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - mintFresh(msg.sender,mintAmount) (contracts/CToken.sol#491)
- Mint(minter,vars.actualMintAmount,vars.mintTokens) (contracts/CToken.sol#473)
  - mintFresh(msg.sender,mintAmount) (contracts/CToken.sol#491)
- Transfer(address(this),minter,vars.mintTokens) (contracts/CToken.sol#474)
  - mintFresh(msg.sender,mintAmount) (contracts/CToken.sol#491)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#632-726):
  External calls:
  - allowed = controller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/CToken.sol#672)
Event emitted after the call(s):
- Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
  - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.REDEEM_COMPTROLLER_REJECTION,allowed) (contracts/CToken.sol#674)
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.REDEEM_TRANSFER_OUT_NOT_POSSIBLE) (contracts/CToken.sol#699)
- Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
  - failOpaque(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.REDEEM_TRANSFER_OUT_NOT_POSSIBLE) (contracts/CToken.sol#699)
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(Error.MARKET_NOT_FRESH,FailureInfo.REDEEM_FRESHNESS_CHECK) (contracts/CToken.sol#679)
- Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
  - failOpaque(Error.MATH_ERROR,FailureInfo.REDEEM_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/CToken.sol#694)
- Redeem(redeemer,vars.redeemAmount,vars.redeemTokens) (contracts/CToken.sol#720)
  - Transfer(redeemer,address(this),vars.redeemTokens) (contracts/CToken.sol#719)
Reentrancy in CToken.redeemInternal(uint256) (contracts/CToken.sol#688-694):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#688)
    - address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#688)
  - controller.beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller.afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- AccrueInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/CToken.sol#470)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(TokenErrorReporter.Error(error),FailureInfo.REDEEM_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#692)
Reentrancy in CToken.redeemInternal(uint256) (contracts/CToken.sol#688-694):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#688)
    - address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - redeemFresh(msg.sender,vars.redeemAmount,vars.redeemTokens) (contracts/CToken.sol#695)
  - allowed = controller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/CToken.sol#672)
  - controller.redeemVerify(address(this),redeemer,vars.redeemAmount,vars.redeemTokens) (contracts/CToken.sol#723)
  - nonReentrant(false) (contracts/CToken.sol#688)
  - controller.beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller.afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
  - redeemFresh(msg.sender,vars.redeemAmount,vars.redeemTokens) (contracts/CToken.sol#695)
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - redeemFresh(msg.sender,vars.redeemAmount,vars.redeemTokens) (contracts/CToken.sol#695)
- Redeem(redeemer,vars.redeemAmount,vars.redeemTokens) (contracts/CToken.sol#720)
  - redeemFresh(msg.sender,vars.redeemAmount,vars.redeemTokens) (contracts/CToken.sol#695)
- Transfer(redeemer,address(this),vars.redeemTokens) (contracts/CToken.sol#719)
  - redeemFresh(msg.sender,vars.redeemAmount,vars.redeemTokens) (contracts/CToken.sol#695)
Reentrancy in CToken.redeemUnderlyingInternal(uint256) (contracts/CToken.sol#604-612):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#606)
    - address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#606)
  - controller.beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller.afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- AccrueInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/CToken.sol#470)
  - error = accrueInterest() (contracts/CToken.sol#605)
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(TokenErrorReporter.Error(error),FailureInfo.REDEEM_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#608)
Reentrancy in CToken.redeemUnderlyingInternal(uint256) (contracts/CToken.sol#604-612):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#606)
    - address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - redeemFresh(msg.sender,0,vars.redeemAmount) (contracts/CToken.sol#611)
  - allowed = controller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/CToken.sol#672)
  - controller.redeemVerify(address(this),redeemer,vars.redeemAmount,vars.redeemTokens) (contracts/CToken.sol#723)
  - nonReentrant(false) (contracts/CToken.sol#606)
  - controller.beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller.afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - redeemFresh(msg.sender,0,vars.redeemAmount) (contracts/CToken.sol#611)
- Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
  - redeemFresh(msg.sender,0,vars.redeemAmount) (contracts/CToken.sol#611)
- Redeem(redeemer,vars.redeemAmount,vars.redeemTokens) (contracts/CToken.sol#720)
  - redeemFresh(msg.sender,0,vars.redeemAmount) (contracts/CToken.sol#611)
- Transfer(redeemer,address(this),vars.redeemTokens) (contracts/CToken.sol#719)
  - redeemFresh(msg.sender,0,vars.redeemAmount) (contracts/CToken.sol#611)
Reentrancy in CToken.repayBorrow(msg.sender,address,uint256) (contracts/CToken.sol#850-856):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#855)
    - address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#856)
  - controller.beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller.afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):

```

```

- AccrueInterest(cashPrior, interestAccumulated, borrowIndexNew, totalBorrowsNew) (contracts/CToken.sol#470)
  error = accrueInterest() (contracts/CToken.sol#485)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - fail(TokenErrorReporter.Error(error), FailureInfo.REPAY_BEHALF_ACCRUE_INTEREST_FAILED, 0) (contracts/CToken.sol#854)
Reentrancy in CToken.repayBorrowBehalfInternal(address, uint256) (contracts/CToken.sol#858-859):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#851)
    - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
  - repayBorrowFresh(msg.sender, borrower, repayAmount) (contracts/CToken.sol#857)
    - allowed = controller.repayBorrowAllowed(address(this), payer, borrower, repayAmount) (contracts/CToken.sol#888)
  - nonReentrant(false) (contracts/CToken.sol#860)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  Event emitted after the call(s):
  - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - repayBorrowFresh(msg.sender, borrower, repayAmount) (contracts/CToken.sol#857)
  - Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
  - repayBorrowFresh(msg.sender, borrower, repayAmount) (contracts/CToken.sol#857)
  - RepayBorrow(payer, borrower, vars.actualRepayAmount, vars.accountBorrowsNew, vars.totalBorrowsNew) (contracts/CToken.sol#938)
  - repayBorrowFresh(msg.sender, borrower, repayAmount) (contracts/CToken.sol#857)
Reentrancy in CToken.repayBorrowFresh(msg.sender, address, address, uint256) (contracts/CToken.sol#838-943):
  External calls:
  - allowed = controller.repayBorrowAllowed(address(this), payer, borrower, repayAmount) (contracts/CToken.sol#888)
  Event emitted after the call(s):
  - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - fail(Error.MARKET_NOT_FRESH, FailureInfo.REPAY_BORROW_FRESHNESS_CHECK, 0) (contracts/CToken.sol#887)
  - Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
  - failOpaque(Error.COMPTROLLER_REJECTION, FailureInfo.REPAY_BORROW_COMPROLLER_REJECTION, allowed, 0) (contracts/CToken.sol#882)
  - Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
  - failOpaque(Error.MATH_ERROR, FailureInfo.REPAY_BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED, uint256(vars.mathErr), 0) (contracts/CToken.sol#898)
  - RepayBorrow(payer, borrower, vars.actualRepayAmount, vars.accountBorrowsNew, vars.totalBorrowsNew) (contracts/CToken.sol#938)
Reentrancy in CToken.repayBorrowInternal(uint256) (contracts/CToken.sol#834-842):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#835)
    - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#836)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  Event emitted after the call(s):
  - AccrueInterest(cashPrior, interestAccumulated, borrowIndexNew, totalBorrowsNew) (contracts/CToken.sol#470)
  - error = accrueInterest() (contracts/CToken.sol#835)
  - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - fail(TokenErrorReporter.Error(error), FailureInfo.REPAY_BORROW_ACCRUE_INTEREST_FAILED, 0) (contracts/CToken.sol#838)
Reentrancy in CToken.repayBorrowInternal(uint256) (contracts/CToken.sol#834-842):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#835)
    - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
  - repayBorrowFresh(msg.sender, msg.sender, repayAmount) (contracts/CToken.sol#841)
  - allowed = controller.repayBorrowAllowed(address(this), payer, borrower, repayAmount) (contracts/CToken.sol#888)
  - nonReentrant(false) (contracts/CToken.sol#860)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  Event emitted after the call(s):
  - Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
  - repayBorrowFresh(msg.sender, msg.sender, repayAmount) (contracts/CToken.sol#841)
  - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - repayBorrowFresh(msg.sender, msg.sender, repayAmount) (contracts/CToken.sol#841)
  - RepayBorrow(payer, borrower, vars.actualRepayAmount, vars.accountBorrowsNew, vars.totalBorrowsNew) (contracts/CToken.sol#938)
  - repayBorrowFresh(msg.sender, msg.sender, repayAmount) (contracts/CToken.sol#841)
Reentrancy in CToken.seize(address, address, address, uint256) (contracts/CToken.sol#1061-1061):
  External calls:
  - seizeInternal(msg.sender, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1062)
  - allowed = controller.seizeAllowed(address(this), seizerToken, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1089)
  - nonReentrant(true) (contracts/CToken.sol#1063)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  Event emitted after the call(s):
  - Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
  - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - seizeInternal(msg.sender, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1062)
  - ReserveAdded(address(this), vars.protocolSeizeAmount, vars.totalReservesNew) (contracts/CToken.sol#1140)
  - seizeInternal(msg.sender, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1062)
  - Transfer(borrower, liquidator, vars.liquidatorSeizeTokens) (contracts/CToken.sol#1139)
  - seizeInternal(msg.sender, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1062)
  - Transfer(borrower, address(this), vars.protocolSeizeTokens) (contracts/CToken.sol#1139)
  - seizeInternal(msg.sender, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1062)
Reentrancy in CToken.seizeInternal(address, address, address, uint256) (contracts/CToken.sol#1067-1147):
  External calls:
  - allowed = controller.seizeAllowed(address(this), seizerToken, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1089)
  Event emitted after the call(s):
  - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - fail(Error.INVALID_ACCOUNT_PAIR, FailureInfo.LIQUIDATOR_SEIZE_LIQUIDATOR_IS_BORROWER) (contracts/CToken.sol#1096)
  - Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
  - failOpaque(Error.COMPTROLLER_REJECTION, FailureInfo.LIQUIDATE_SEIZE_COMPROLLER_REJECTION, allowed) (contracts/CToken.sol#1091)
  - Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
  - failOpaque(Error.MATH_ERROR, FailureInfo.LIQUIDATE_SEIZE_BALANCE_INCREMENT_FAILED, uint256(vars.mathErr)) (contracts/CToken.sol#1124)
  - Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
  - failOpaque(Error.MATH_ERROR, FailureInfo.LIQUIDATE_SEIZE_BALANCE_INCREMENT_FAILED, uint256(vars.mathErr)) (contracts/CToken.sol#1180)
  - ReserveAdded(address(this), vars.protocolSeizeAmount, vars.totalReservesNew) (contracts/CToken.sol#1138)
  - Transfer(borrower, liquidator, vars.liquidatorSeizeTokens) (contracts/CToken.sol#1139)
  - Transfer(borrower, address(this), vars.protocolSeizeTokens) (contracts/CToken.sol#1139)
Reentrancy in CToken.seizeInternal(address, address, address, uint256) (contracts/CToken.sol#1067-1147):
  External calls:
  - require(bool, string)(accrueInterest() == uint256(Error.NO_ERROR), accrue interest failed) (contracts/CToken.sol#279)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#278)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  Event emitted after the call(s):
  - AccrueInterest(cashPrior, interestAccumulated, borrowIndexNew, totalBorrowsNew) (contracts/CToken.sol#470)
  - require(bool, string)(accrueInterest() == uint256(Error.NO_ERROR), accrue interest failed) (contracts/CToken.sol#279)
Reentrancy in CToken.transfer(address, uint256) (contracts/CToken.sol#162-164):
  External calls:
  - transferTokens(msg.sender, msg.sender, dst, amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#163)
  - allowed = controller.transferAllowed(address(this), src, dst, tokens) (contracts/CToken.sol#495)
  - nonReentrant(false) (contracts/CToken.sol#162)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  Event emitted after the call(s):
  - Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
  - transferTokens(msg.sender, msg.sender, dst, amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#163)
  - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - transferTokens(msg.sender, msg.sender, dst, amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#163)
  - Transfer(src, dst, tokens) (contracts/CToken.sol#147)
  - transferTokens(msg.sender, msg.sender, dst, amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#163)
Reentrancy in CToken.transferFrom(address, address, uint256) (contracts/CToken.sol#173-176):
  External calls:
  - transferTokens(msg.sender, src, dst, amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#174)
  - allowed = controller.transferAllowed(address(this), src, dst, tokens) (contracts/CToken.sol#495)
  - nonReentrant(false) (contracts/CToken.sol#173)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  Event emitted after the call(s):
  - Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
  - transferTokens(msg.sender, src, dst, amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#174)
  - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - transferTokens(msg.sender, src, dst, amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#174)
  - Transfer(src, dst, tokens) (contracts/CToken.sol#147)
  - transferTokens(msg.sender, src, dst, amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#174)
Reentrancy in CToken.transferTokens(address, address, address, uint256) (contracts/CToken.sol#193-194):
  External calls:
  - allowed = controller.transferAllowed(address(this), src, dst, tokens) (contracts/CToken.sol#495)
  Event emitted after the call(s):
  - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - fail(Error.BAD_INPUT, FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/CToken.sol#192)
  - Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#226)
  - failOpaque(Error.COMPTROLLER_REJECTION, FailureInfo.TRANSFER_COMPROLLER_REJECTION, allowed) (contracts/CToken.sol#197)
  - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - fail(Error.MATH_ERROR, FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/CToken.sol#221)
  - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - fail(Error.MATH_ERROR, FailureInfo.TRANSFER_NOT_TOO_MUCH) (contracts/CToken.sol#226)
  - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - fail(Error.MATH_ERROR, FailureInfo.TRANSFER_TOO_MUCH) (contracts/CToken.sol#231)

```

```

- Transfer(src, dst, tokens) (contracts/CToken.sol#147)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
CToken._functionCall(address, bytes, string) (contracts/CToken.sol#159-161) uses assembly
  INLINE ASM (contracts/CToken.sol#160-160)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#assembly-usage
CToken.doTransferIn(address, uint256) (contracts/CToken.sol#164) is never used and should be removed
CToken.doTransferOut(address, uint256) (contracts/CToken.sol#165) is never used and should be removed
CToken.getCashPrior() (contracts/CToken.sol#157) is never used and should be removed
ComptrollerErrorReporter.Fall(ComptrollerErrorReporter, Error, ComptrollerErrorReporter, FailureInfo) (contracts/ErrorReporter.sol#69-73) is never used and should be removed
ComptrollerErrorReporter.Fall(ComptrollerErrorReporter, Error, ComptrollerErrorReporter, FailureInfo, uint256) (contracts/ErrorReporter.sol#74-82) 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#140-142) is never used and should be removed
Exponential.mulExp(ExponentialNoError.Exp, ExponentialNoError.Exp) (contracts/Exponential.sol#95-99) is never used and should be removed
Exponential.mulExp(ExponentialNoError.Exp, ExponentialNoError.Exp) (contracts/Exponential.sol#135-135) is never used and should be removed
Exponential.mulExp(uint256, uint256) (contracts/Exponential.sol#146-142) is never used and should be removed
Exponential.mulExp(ExponentialNoError.Exp, ExponentialNoError.Exp) (contracts/Exponential.sol#167-173) is never used and should be removed
Exponential.mulExp(ExponentialNoError.Exp, ExponentialNoError.Exp) (contracts/Exponential.sol#179-186) is never used and should be removed
Exponential.subExp(ExponentialNoError.Exp, ExponentialNoError.Exp) (contracts/Exponential.sol#146-142) is never used and should be removed
ExponentialNoError.add(ExponentialNoError, Double, ExponentialNoError, Double) (contracts/ExponentialNoError.sol#95-99) is never used and should be removed
ExponentialNoError.add(ExponentialNoError, Exp, ExponentialNoError, Exp) (contracts/ExponentialNoError.sol#87-89) is never used and should be removed
ExponentialNoError.div(ExponentialNoError, Double, ExponentialNoError, Double) (contracts/ExponentialNoError.sol#171-173) is never used and should be removed
ExponentialNoError.div(ExponentialNoError, Exp, ExponentialNoError, Exp) (contracts/ExponentialNoError.sol#159-161) is never used and should be removed
ExponentialNoError.div(ExponentialNoError, Exp, uint256) (contracts/ExponentialNoError.sol#160-160) is never used and should be removed
ExponentialNoError.div(ExponentialNoError, Double, ExponentialNoError, Double) (contracts/ExponentialNoError.sol#179-181) is never used and should be removed
ExponentialNoError.div(ExponentialNoError, Exp) (contracts/ExponentialNoError.sol#167-169) is never used and should be removed
ExponentialNoError.div(ExponentialNoError, Exp, ExponentialNoError, Exp) (contracts/ExponentialNoError.sol#181-185) is never used and should be removed
ExponentialNoError.div(ExponentialNoError, string) (contracts/ExponentialNoError.sol#187-190) is never used and should be removed
ExponentialNoError.fraction(uint256, uint256) (contracts/ExponentialNoError.sol#192-194) is never used and should be removed
ExponentialNoError.greaterThanExp(ExponentialNoError, Exp, ExponentialNoError, Exp) (contracts/ExponentialNoError.sol#166-168) is never used and should be removed
ExponentialNoError.greaterThanExp(ExponentialNoError, Exp, ExponentialNoError, Exp) (contracts/ExponentialNoError.sol#175-175) is never used and should be removed
ExponentialNoError.lessThanExp(ExponentialNoError, Exp, ExponentialNoError, Exp) (contracts/ExponentialNoError.sol#152-154) is never used and should be removed
ExponentialNoError.lessThanOrEqualExp(ExponentialNoError, Exp, ExponentialNoError, Exp) (contracts/ExponentialNoError.sol#159-161) is never used and should be removed
ExponentialNoError.mul(ExponentialNoError, Double, ExponentialNoError, Double) (contracts/ExponentialNoError.sol#135-135) is never used and should be removed
ExponentialNoError.mul(ExponentialNoError, Double, uint256) (contracts/ExponentialNoError.sol#138-140) 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, string) (contracts/ExponentialNoError.sol#145-146) is never used and should be removed
ExponentialNoError.safe2x2(uint256, string) (contracts/ExponentialNoError.sol#77-80) is never used and should be removed
ExponentialNoError.safe2x2(uint256, string) (contracts/ExponentialNoError.sol#82-85) is never used and should be removed
ExponentialNoError.sub(ExponentialNoError, Double, ExponentialNoError, Double) (contracts/ExponentialNoError.sol#159-161) is never used and should be removed
ExponentialNoError.sub(ExponentialNoError, Exp, ExponentialNoError, Exp) (contracts/ExponentialNoError.sol#105-107) is never used and should be removed
UnitrollerAdminStorage.hasAdminRights() (contracts/ComptrollerStorage.sol#186-188) is never used and should be removed
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#low-level-calls
Low level call in CToken.finishInterestAccrual(uint256, uint256, uint256, uint256) (contracts/CToken.sol#437-493)
  address interestRateModel.call(abi.encodeWithSignature(checkpointInterest(uint256, bytes32,uint256, bytes32,uint256))) (contracts/CToken.sol#473)
Low level call in CToken._setInterestRateModelFresh(InterestRateModel) (contracts/CToken.sol#1479-1512)
  address oldInterestRateModel.call(abi.encodeWithSignature(resetInterestCheckpoint(1))) (contracts/CToken.sol#1506)
  address newInterestRateModel.call(abi.encodeWithSignature(checkpointInterest(1))) (contracts/CToken.sol#1509)
Low level call in CToken._functionCall(address, bytes, string) (contracts/CToken.sol#159-161):
  (success, returnedData) = target.call(data) (contracts/CToken.sol#159)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#low-level-calls
Function CTokenDelegate._becomeImplementation(bytes) (contracts/GTokenDelegate.sol#28-30) is not in mixedCase
Function CTokenDelegate._setImplementationSafe(address, bool, bytes) (contracts/GTokenDelegate.sol#74-80) is not in mixedCase
Function CTokenDelegate._prepare() (contracts/GTokenDelegate.sol#86-91) is not in mixedCase
Function CToken._setAdminFee(uint256) (contracts/CToken.sol#1176-1184) is not in mixedCase
Function CToken._setWithdrawalFee(uint256) (contracts/CToken.sol#1214-1219) is not in mixedCase
Function CToken._reduceReserves(uint256) (contracts/CToken.sol#1285-1293) is not in mixedCase
Function CToken._withdrawReserves(uint256) (contracts/CToken.sol#1345-1350) is not in mixedCase
Function CToken._setWithdrawalFee(InterestRateModel) (contracts/CToken.sol#1408-1412) is not in mixedCase
Function CToken._setInterestRateModel(InterestRateModel) (contracts/CToken.sol#1463-1471) is not in mixedCase
Function CToken._setNameAndSymbol(string, string) (contracts/CToken.sol#1520-1529) is not in mixedCase
Parameter CToken._setNameAndSymbol(string, string), name (contracts/CToken.sol#1521) is not in mixedCase
Parameter CToken._setNameAndSymbol(string, string), _symbol (contracts/CToken.sol#1521) is not in mixedCase
Constant CTokenAdminStorage.fuseAdmin (contracts/CTokenInterfaces.sol#112) is not in UPPER_CASE_WITH_UNDERSCORES
Variable CTokenStorage._notEntered (contracts/CTokenInterfaces.sol#112) is not in mixedCase
Constant CTokenStorage.borrowRateMaxMantissa (contracts/CTokenInterfaces.sol#139) is not in UPPER_CASE_WITH_UNDERSCORES
Constant CTokenStorage.reserveFactorPlusFeeMaxMantissa (contracts/CTokenInterfaces.sol#140) is not in UPPER_CASE_WITH_UNDERSCORES
Variable CTokenStorage._pendingAdmin (contracts/CTokenInterfaces.sol#140) is not in mixedCase
Constant CTokenInterface.isCToken (contracts/CTokenInterfaces.sol#151) is not in UPPER_CASE_WITH_UNDERSCORES
Constant CTokenInterface.isCether (contracts/CTokenInterfaces.sol#150) is not in UPPER_CASE_WITH_UNDERSCORES
Constant CTokenInterface.isCethex (contracts/CTokenInterfaces.sol#149) is not in UPPER_CASE_WITH_UNDERSCORES
Function CTokenInterface._prepare() (contracts/CTokenInterfaces.sol#132) is not in mixedCase
Function CTokenInterface._setImplementationSafe(address, bool, bytes) (contracts/CTokenInterfaces.sol#110) is not in mixedCase
Function CTokenInterface._becomeImplementation(bytes) (contracts/CTokenInterfaces.sol#126) is not in mixedCase
Function CTokenInterface._prepare() (contracts/CTokenInterfaces.sol#132) is not in mixedCase
Function ComptrollerInterface._beforeNonReentrant() (contracts/ComptrollerInterface.sol#67) is not in mixedCase
Function ComptrollerInterface._afterNonReentrant() (contracts/ComptrollerInterface.sol#67) is not in mixedCase
Constant ComptrollerInterface.isComptroller (contracts/ComptrollerInterface.sol#65) is not in UPPER_CASE_WITH_UNDERSCORES
Constant UnitrollerAdminStorage.fuseAdmin (contracts/ComptrollerStorage.sol#111) is not in UPPER_CASE_WITH_UNDERSCORES
Variable ComptrollerV3Storage._mintGuardianPaused (contracts/ComptrollerStorage.sol#145) is not in mixedCase
Variable ComptrollerV3Storage._borrowGuardianPaused (contracts/ComptrollerStorage.sol#146) is not in mixedCase
Variable ComptrollerV3Storage._notEntered (contracts/ComptrollerStorage.sol#172) is not in mixedCase
Variable ComptrollerV3Storage._notEnteredInitialized (contracts/ComptrollerStorage.sol#170) is not in mixedCase
Function ExponentialNoError.mul_ScalarFuncateAddUnit(ExponentialNoError.Exp, uint256) (contracts/ExponentialNoError.sol#44-47) is not in mixedCase
Constant ExponentialNoError.expScale (contracts/ExponentialNoError.sol#11) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.doubleScale (contracts/ExponentialNoError.sol#12) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.halfExpScale (contracts/ExponentialNoError.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError.mantissaOne (contracts/ExponentialNoError.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant InterestRateModel.isInterestRateModel (contracts/InterestRateModel.sol#18) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PriceOracle.isPriceOracle (contracts/PriceOracle.sol#17) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
Redundant expression "data" (contracts/GTokenDelegate.sol#22) in CTokenDelegate (contracts/GTokenDelegate.sol#18-92)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#redundant-statements
Variable CToken.seize(address, address, uint256).seizeTokens (contracts/CToken.sol#104) is too similar to CToken.seizeInternal(address, address, address, uint256).seizeToken (contracts/CToken.sol#107)
Variable CToken.liquidateBorrowFresh(address, address, uint256, CTokenInterface).seizeTokens (contracts/CToken.sol#105) is too similar to CToken.seizeInternal(address, address, address, uint256).seizeToken (contracts/CToken.sol#107)
Variable CToken.seizeInternal(address, address, address, uint256).seizeTokens (contracts/CToken.sol#107) is too similar to CToken.seizeInternal(address, address, address, uint256).seizeToken (contracts/CToken.sol#107)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#variable-names-are-too-similar
CTokenStorage._pendingAdmin (contracts/CTokenInterfaces.sol#149) is never used in CTokenDelegate (contracts/GTokenDelegate.sol#18-92)
ComptrollerV3Storage.maxAssets (contracts/ComptrollerStorage.sol#170) is never used in ComptrollerV3Storage (contracts/ComptrollerStorage.sol#153-176)
ComptrollerV3Storage.borrowerIndexes (contracts/ComptrollerStorage.sol#111) is never used in ComptrollerV3Storage (contracts/ComptrollerStorage.sol#153-176)
ComptrollerV3Storage.whitelistIndexes (contracts/ComptrollerStorage.sol#137) is never used in ComptrollerV3Storage (contracts/ComptrollerStorage.sol#153-176)
ComptrollerV3Storage._notEntered (contracts/ComptrollerStorage.sol#172) is never used in ComptrollerV3Storage (contracts/ComptrollerStorage.sol#153-176)
ComptrollerV3Storage._notEnteredInitialized (contracts/ComptrollerStorage.sol#170) is never used in ComptrollerV3Storage (contracts/ComptrollerStorage.sol#153-176)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#unused-state-variable
Cerc28Storage.underlying (contracts/CTokenInterfaces.sol#276) should be constant
CTokenStorage._pendingAdmin (contracts/CTokenInterfaces.sol#149) should be constant
ComptrollerV3Storage.closeFeesMantissa (contracts/ComptrollerStorage.sol#168) should be constant
ComptrollerV3Storage.liquidationIncentiveMantissa (contracts/ComptrollerStorage.sol#165) should be constant
ComptrollerV3Storage.maxAssets (contracts/ComptrollerStorage.sol#170) should be constant
ComptrollerV3Storage._borrowGuardianPaused (contracts/ComptrollerStorage.sol#146) should be constant
ComptrollerV3Storage._mintGuardianPaused (contracts/ComptrollerStorage.sol#145) should be constant
ComptrollerV3Storage.enforceMintLimit (contracts/ComptrollerStorage.sol#128) should be constant
ComptrollerV3Storage.pauseGuardian (contracts/ComptrollerStorage.sol#144) should be constant
ComptrollerV3Storage.seizeGuardianPaused (contracts/ComptrollerStorage.sol#148) should be constant
ComptrollerV3Storage.transferGuardianPaused (contracts/ComptrollerStorage.sol#147) should be constant
ComptrollerV3Storage._notEntered (contracts/ComptrollerStorage.sol#172) should be constant
ComptrollerV3Storage._notEnteredInitialized (contracts/ComptrollerStorage.sol#170) should be constant
ComptrollerV3Storage.autoImplementation (contracts/ComptrollerStorage.sol#187) should be constant
ComptrollerV3Storage.borrowGuardian (contracts/ComptrollerStorage.sol#168) should be constant
UnitrollerAdminStorage.admin (contracts/ComptrollerStorage.sol#146) should be constant
UnitrollerAdminStorage.adminHasRights (contracts/ComptrollerStorage.sol#143) should be constant
UnitrollerAdminStorage.comptrollerImplementation (contracts/ComptrollerStorage.sol#143) should be constant
UnitrollerAdminStorage.fuseAdminRights (contracts/ComptrollerStorage.sol#146) should be constant
UnitrollerAdminStorage.pendingAdmin (contracts/ComptrollerStorage.sol#121) should be constant
UnitrollerAdminStorage._pendingAdminImplementation (contracts/ComptrollerStorage.sol#148) should be constant
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
initialize(ComptrollerInterface, InterestRateModel, string, string, uint256, uint256) should be declared external:
  - Cether.initialize(ComptrollerInterface, InterestRateModel, string, string, uint256, uint256) (contracts/Cether.sol#189-29)
  - setInterestRateModel(InterestRateModel) should be declared external:
    - CToken._setInterestRateModel(InterestRateModel) (contracts/CToken.sol#1463-1471)
    - CTokenInterface._setInterestRateModel(InterestRateModel) (contracts/CTokenInterfaces.sol#269)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#public-function-that-could-be-declared-external

```


CTokenInterfaces.sol

```

UnitrollerAdminStorage admin (contracts/ComptrollerStorage.sol#16) is never initialized. It is used in:
- UnitrollerAdminStorage.hasAdminRights() (contracts/ComptrollerStorage.sol#36-38)
Reference: https://github.com/crytic/Slither/wiki/Detector-Documentation#uninitialized-state-variables

EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#8-70) has incorrect ERC20 function interface:EIP20NonStandardInterface.transfer(address,uint256) (contracts/EIP20NonStandardInterface.sol#34)
EIP20NonStandardInterface (contracts/EIP20NonStandardInterface.sol#8-70) has incorrect ERC20 function interface:EIP20NonStandardInterface.transferFrom(address,address,uint256) (contracts/EIP20NonStandardInterface.sol#34)
Reference: https://github.com/crytic/Slither/wiki/Detector-Documentation#incorrect-erc20-interface

CToken accrueInterest() (contracts/CToken.sol#411-432) uses a dangerous strict equality:
- accrualBlockNumber == currentBlockNumber (contracts/CToken.sol#412)
CToken accrueInterest() (contracts/CToken.sol#411-432) uses a dangerous strict equality:
- require(Bool, string)(mathErr == MathError.NO_ERROR, could not calculate block delta) (contracts/CToken.sol#429)
CToken balancesOfUnderlyingAddress() (contracts/CToken.sol#217-222) uses a dangerous strict equality:
- require(Bool, string)(mErr == MathError.NO_ERROR, balance could not be calculated) (contracts/CToken.sol#220)
CToken borrowBalanceStored(address) (contracts/CToken.sol#298-302) uses a dangerous strict equality:
- require(Bool, string)(err == MathError.NO_ERROR, borrowBalanceStored: borrowBalanceStoredInternal failed) (contracts/CToken.sol#300)
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) uses a dangerous strict equality:
- b == 0 (contracts/CarefulMath.sol#42)
CToken.exchangeRateStored() (contracts/CToken.sol#355-359) uses a dangerous strict equality:
- require(Bool, string)(err == MathError.NO_ERROR, exchangeRateStoredInternal failed) (contracts/CToken.sol#367)
CToken.exchangeRateStoredInternal() (contracts/CToken.sol#366-396) uses a dangerous strict equality:
- totalSupply == 0 (contracts/CToken.sol#366)
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,uint8,uint256,uint256) (contracts/CToken.sol#34-75) uses a dangerous strict equality:
- require(Bool, string)(accrualBlockNumber == 0 && borrowIndex == 0, initialized-only-once) (contracts/CToken.sol#43)
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,uint8,uint256,uint256) (contracts/CToken.sol#34-75) uses a dangerous strict equality:
- require(Bool, string)(err == uint256(Error.NO_ERROR), setting lim failed) (contracts/CToken.sol#69)
CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,uint8,uint256,uint256) (contracts/CToken.sol#34-75) uses a dangerous strict equality:
- require(Bool, string)(err == uint256(Error.NO_ERROR), setting admin fee failed) (contracts/CToken.sol#71)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#981-1009) uses a dangerous strict equality:
- require(Bool, string)(amountSeizeError == uint256(Error.NO_ERROR), LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/CToken.sol#1026)
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#981-1009) uses a dangerous strict equality:
- require(Bool, string)(tokenSeizeError == uint256(Error.NO_ERROR), token seizure failed) (contracts/CToken.sol#1048)
CToken.mintFresh(address,uint256) (contracts/CToken.sol#811-888) uses a dangerous strict equality:
- require(Bool, string)(vars.mathErr == MathError.NO_ERROR, MINT_EXCHANGE_CALCULATION_FAILED) (contracts/CToken.sol#857)
Exponential.mulExp(ExponentialNoError,ExponentialNoError,Exp) (contracts/Exponential.sol#130-158) uses a dangerous strict equality:
- assert(Bool)(err2 == MathError.NO_ERROR) (contracts/Exponential.sol#152)
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) uses a dangerous strict equality:
- a == 0 (contracts/CarefulMath.sol#24)
ExponentialNoError.mulUInt(uint256,uint256,string) (contracts/ExponentialNoError.sol#150-157) uses a dangerous strict equality:
- a == 0 || b == 0 (contracts/ExponentialNoError.sol#151)
ExponentialNoError.mulUInt(uint256,uint256,string) (contracts/ExponentialNoError.sol#150-157) uses a dangerous strict equality:
- require(Bool, string)(c / a == b, errorMessage) (contracts/ExponentialNoError.sol#155)
CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#878-945) uses a dangerous strict equality:
- require(Bool, string)(vars.mathErr == MathError.NO_ERROR, REPAY_BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED) (contracts/CToken.sol#927)
CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#878-945) uses a dangerous strict equality:
- require(Bool, string)(vars.mathErr == MathError.NO_ERROR, REPAY_BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED) (contracts/CToken.sol#930)
CToken.seizeInternal(address,address,uint256) (contracts/CToken.sol#1167-1147) uses a dangerous strict equality:
- require(Bool, string)(vars.mathErr == MathError.NO_ERROR, exchange rate math error) (contracts/CToken.sol#1116)
CToken.transfer(address,uint256) (contracts/CToken.sol#162-164) uses a dangerous strict equality:
- transferTokens(msg.sender,msg.sender,dst,amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#163)
CToken.transferFrom(address,address,uint256) (contracts/CToken.sol#173-175) uses a dangerous strict equality:
- transferTokens(msg.sender,src,dst,amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#174)
Reference: https://github.com/crytic/Slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in CToken_beforeNonReentrant(Bool) (contracts/CToken.sol#1569-1573):
External calls:
- comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
State variables written after the call(s):
- _notEntered == false (contracts/CToken.sol#1572)
Reentrancy in CToken_reduceReserves(uint256) (contracts/CToken.sol#1285-1293):
External calls:
- error = accrueInterest() (contracts/CToken.sol#1286)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#1473)
- nonReentrant(false) (contracts/CToken.sol#1288)
- comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
- comptroller_afterNonReentrant() (contracts/CToken.sol#1582)
State variables written after the call(s):
- _reduceReserveFresh(reduceAmount) (contracts/CToken.sol#1292)
- totalReserve == totalReserveNew (contracts/CToken.sol#1333)
Reentrancy in CToken_setAdminFee(uint256) (contracts/CToken.sol#1174-1184):
External calls:
- error = accrueInterest() (contracts/CToken.sol#1177)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#1473)
- nonReentrant(false) (contracts/CToken.sol#1176)
- comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
- comptroller_afterNonReentrant() (contracts/CToken.sol#1582)
State variables written after the call(s):
- _setAdminFeeFresh(newAdminFeeMantissa) (contracts/CToken.sol#1183)
- adminFeeMantissa = newAdminFeeMantissa (contracts/CToken.sol#1217)
- _setAdminFeeFresh(newAdminFeeMantissa) (contracts/CToken.sol#1218)
- fuseFeeMantissa = newFuseFeeMantissa (contracts/CToken.sol#1227)
Reentrancy in CToken_setInterestRateModel(InterestRateModel) (contracts/CToken.sol#1463-1471):
External calls:
- error = accrueInterest() (contracts/CToken.sol#1464)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#1473)
- _setInterestRateModelFresh(newInterestRateModel) (contracts/CToken.sol#1470)
- address(oldInterestRateModel).call(abi.encodeWithSignature(resetInterestCheckpoints())) (contracts/CToken.sol#1506)
- address(newInterestRateModel).call(abi.encodeWithSignature(checkpointInterest())) (contracts/CToken.sol#1509)
State variables written after the call(s):
- _setInterestRateModelFresh(newInterestRateModel) (contracts/CToken.sol#1478)
- interestRateModel = newInterestRateModel (contracts/CToken.sol#1508)
Reentrancy in CToken_setReserveFactor(uint256) (contracts/CToken.sol#1241-1249):
External calls:
- error = accrueInterest() (contracts/CToken.sol#1242)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#1473)
- nonReentrant(false) (contracts/CToken.sol#1241)
- comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
- comptroller_afterNonReentrant() (contracts/CToken.sol#1582)
State variables written after the call(s):
- _setReserveFactorFresh(newReserveFactorMantissa) (contracts/CToken.sol#1248)
- reserveFactorMantissa = newReserveFactorMantissa (contracts/CToken.sol#1279)
Reentrancy in CToken_withdrawAdminFees(uint256) (contracts/CToken.sol#1404-1412):
External calls:
- error = accrueInterest() (contracts/CToken.sol#1406)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#1473)
- nonReentrant(false) (contracts/CToken.sol#1404)
- comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
- comptroller_afterNonReentrant() (contracts/CToken.sol#1582)
State variables written after the call(s):
- _withdrawAdminFeeFresh(withdrawAmount) (contracts/CToken.sol#1411)
- totalAdminFees == totalAdminFeesNew (contracts/CToken.sol#1449)
Reentrancy in CToken_withdrawFuseFees(uint256) (contracts/CToken.sol#1348-1356):
External calls:
- error = accrueInterest() (contracts/CToken.sol#1349)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#1473)
- nonReentrant(false) (contracts/CToken.sol#1348)
- comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
- comptroller_afterNonReentrant() (contracts/CToken.sol#1582)
State variables written after the call(s):
- _withdrawFuseFeesFresh(withdrawAmount) (contracts/CToken.sol#1355)
- totalFuseFees == totalFuseFeesNew (contracts/CToken.sol#1391)
Reentrancy in CToken_borrowFresh(address,uint256) (contracts/CToken.sol#755-827):
External calls:
- allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#759)
- allowed = comptroller.borrowWithinLimits(address(this),vars.accountBorrowNew) (contracts/CToken.sol#792)
State variables written after the call(s):
- accountBorrow[borrower].principal = vars.accountBorrowNew (contracts/CToken.sol#815)
- accountBorrow[borrower].interestIndex = borrowIndex (contracts/CToken.sol#816)
Reentrancy in CToken_borrowInternal(uint256) (contracts/CToken.sol#733-741):
External calls:
- error = accrueInterest() (contracts/CToken.sol#734)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#1473)
- borrowFresh(msg.sender,borrowAmount) (contracts/CToken.sol#748)
- allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#767)
- allowed = comptroller.borrowWithinLimits(address(this),vars.accountBorrowNew) (contracts/CToken.sol#792)
- nonReentrant(false) (contracts/CToken.sol#733)
- comptroller_beforeNonReentrant() (contracts/CToken.sol#1571)
- comptroller_afterNonReentrant() (contracts/CToken.sol#1582)
State variables written after the call(s):
- borrowFresh(msg.sender,borrowAmount) (contracts/CToken.sol#748)
- totalBorrow == vars.totalBorrowNew (contracts/CToken.sol#817)

```

```

Reentrancy in CToken.liquidateBorrowInternal(address,uint256,CTokenInterface) (contracts/CToken.sol#955-978):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#956)
    - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - error = cTokenCollateral.accrueInterest() (contracts/CToken.sol#962)
  - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#969)
    - allowed = controller.liquidateBorrowAllowed(address(this),address(cTokenCollateral),liquidator,borrower,repayAmount) (contracts/CToken.sol#983)
    - allowed = controller.seizeAllowed(address(this),seizeToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#1089)
    - allowed = controller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#888)
    - seizeError = cTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/CToken.sol#1036)
  - nonReentrant(false) (contracts/CToken.sol#958)
  - controller._beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller._afterNonReentrant() (contracts/CToken.sol#1582)
  State variables written after the call(s):
  - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#969)
  - totalBorrows = vars.totalBorrowsNew (contracts/CToken.sol#935)
  - liquidateBorrowFresh(msg.sender,borrower,repayAmount,cTokenCollateral) (contracts/CToken.sol#969)
  - totalReserves = vars.totalReservesNew (contracts/CToken.sol#1132)
Reentrancy in CToken.nonReentrant(bool) (contracts/CToken.sol#1558-1562):
  External calls:
  - _beforeNonReentrant(localOnly) (contracts/CToken.sol#1559)
  - controller._beforeNonReentrant() (contracts/CToken.sol#1571)
  - _afterNonReentrant(localOnly) (contracts/CToken.sol#1564)
  - controller._afterNonReentrant() (contracts/CToken.sol#1582)
  State variables written after the call(s):
  - _afterNonReentrant(localOnly) (contracts/CToken.sol#1564)
  - notEntered = true (contracts/CToken.sol#1581)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#632-726):
  External calls:
  - allowed = controller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/CToken.sol#672)
  State variables written after the call(s):
  - totalSupply = vars.totalSupplyNew (contracts/CToken.sol#715)
Reentrancy in CToken.repayBorrowHalfInternal(address,uint256) (contracts/CToken.sol#850-858):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#855)
    - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - repayBorrowFresh(msg.sender,borrower,repayAmount) (contracts/CToken.sol#857)
  - allowed = controller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#888)
  - nonReentrant(false) (contracts/CToken.sol#858)
  - controller._beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller._afterNonReentrant() (contracts/CToken.sol#1582)
  State variables written after the call(s):
  - repayBorrowFresh(msg.sender,borrower,repayAmount) (contracts/CToken.sol#857)
  - totalBorrow = vars.totalBorrowNew (contracts/CToken.sol#935)
Reentrancy in CToken.repayBorrowInternal(uint256) (contracts/CToken.sol#834-842):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#835)
    - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - repayBorrowFresh(msg.sender,msg.sender,repayAmount) (contracts/CToken.sol#841)
  - allowed = controller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#888)
  - nonReentrant(false) (contracts/CToken.sol#834)
  - controller._beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller._afterNonReentrant() (contracts/CToken.sol#1582)
  State variables written after the call(s):
  - repayBorrowFresh(msg.sender,msg.sender,repayAmount) (contracts/CToken.sol#841)
  - totalBorrow = vars.totalBorrowNew (contracts/CToken.sol#935)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
CToken.finishInterestAccrued(uint256,uint256,uint256) (contracts/CToken.sol#437-476) ignores return value by address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
CToken.setInterestRateModelFresh(InterestRateModel) (contracts/CToken.sol#1479-1512) ignores return value by address(oldInterestRateModel).call(abi.encodeWithSignature(resetInterestCheckpoint())) (contracts/CToken.sol#1479-1512)
CToken.setInterestRateModelFresh(InterestRateModel) (contracts/CToken.sol#1479-1512) ignores return value by address(newInterestRateModel).call(abi.encodeWithSignature(checkpointInterest())) (contracts/CToken.sol#1479-1512)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-calls
CToken.repayBorrowFresh(address,uint256,uint256,vars) (contracts/CToken.sol#890) is a local variable never initialized
CToken.mintFresh(address,uint256,vars) (contracts/CToken.sol#52) is a local variable never initialized
CToken.seizeInternal(address,address,uint256,vars) (contracts/CToken.sol#1099) is a local variable never initialized
CToken.borrowFresh(address,uint256,vars) (contracts/CToken.sol#774) is a local variable never initialized
CToken.redeemFresh(address,uint256,uint256,vars) (contracts/CToken.sol#655) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
Exponential.div(ScalarByExpFuncate(uint256,ExponentialNoError.Exp.fraction) (contracts/Exponential.sol#126) shadows:
  - ExponentialNoError.fraction(uint256,uint256) (contracts/ExponentialNoError.sol#192-194) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
CToken.initialize(ControllerInterface,InterestRateModel,uint256,string,string,uint8,uint256,uint256) (contracts/CToken.sol#934-945) should emit an event for:
  - initialExchangeRateMantissa = initialExchangeRateMantissa (contracts/CToken.sol#846)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
Reentrancy in CToken.borrowFresh(address,uint256) (contracts/CToken.sol#755-827):
  External calls:
  - allowed = controller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#757)
  - allowed = controller.borrowWithinLimits(address(this),vars.accountBorrowNew) (contracts/CToken.sol#792)
  State variables written after the call(s):
  - totalBorrows = vars.totalBorrowNew (contracts/CToken.sol#817)
Reentrancy in CToken.initialize(ControllerInterface,InterestRateModel,uint256,string,string,uint8,uint256,uint256) (contracts/CToken.sol#934-945):
  External calls:
  - err = _setInterestRateModelFresh(interestRateModel) (contracts/CToken.sol#98)
  - address(oldInterestRateModel).call(abi.encodeWithSignature(resetInterestCheckpoints())) (contracts/CToken.sol#1586)
  - address(newInterestRateModel).call(abi.encodeWithSignature(checkpointInterest())) (contracts/CToken.sol#1589)
  State variables written after the call(s):
  - _notEntered = true (contracts/CToken.sol#874)
  - err = _setInterestRateModelFresh(interestRateModel) (contracts/CToken.sol#98)
  - adminMantissa = newAdminMantissa (contracts/CToken.sol#1217)
  - decimals = decimals (contracts/CToken.sol#643)
  - err = _setInterestRateModelFresh(interestRateModel) (contracts/CToken.sol#98)
  - fuseFasMantissa = newFuseFasMantissa (contracts/CToken.sol#1227)
  - name = name (contracts/CToken.sol#651)
  - err = _setReserveFactorFresh(reserveFactorMantissa) (contracts/CToken.sol#656)
  - reserveFactorMantissa = newReserveFactorMantissa (contracts/CToken.sol#1273)
  - symbol = symbol (contracts/CToken.sol#62)
Reentrancy in CToken.mintFresh(address,uint256) (contracts/CToken.sol#511-588):
  External calls:
  - allowed = controller.mintAllowed(address(this),minter,mintAmount) (contracts/CToken.sol#513)
  State variables written after the call(s):
  - accountTokens[minter] = vars.accountTokensNew (contracts/CToken.sol#578)
  - totalSupply = vars.totalSupplyNew (contracts/CToken.sol#569)
Reentrancy in CToken.redeemFresh(address,uint256,uint256) (contracts/CToken.sol#632-726):
  External calls:
  - allowed = controller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/CToken.sol#672)
  State variables written after the call(s):
  - accountTokens[redeemer] = vars.accountTokensNew (contracts/CToken.sol#716)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#878-945):
  External calls:
  - allowed = controller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/CToken.sol#888)
  State variables written after the call(s):
  - accountBorrow[borrower].principal = vars.accountBorrowNew (contracts/CToken.sol#933)
  - accountBorrow[borrower].interestIndex = borrowIndex (contracts/CToken.sol#934)
  - totalBorrow = vars.totalBorrowNew (contracts/CToken.sol#935)
Reentrancy in CToken.seizeInternal(address,address,uint256) (contracts/CToken.sol#1087-1147):
  External calls:
  - allowed = controller.seizeAllowed(address(this),seizeToken,liquidator,borrower,seizeTokens) (contracts/CToken.sol#1089)
  State variables written after the call(s):
  - accountTokens[borrower] = vars.borrowerTokensNew (contracts/CToken.sol#1134)
  - accountTokens[liquidator] = vars.liquidatorTokensNew (contracts/CToken.sol#1135)
  - totalReserves = vars.totalReservesNew (contracts/CToken.sol#1132)
  - totalSupply = vars.totalSupplyNew (contracts/CToken.sol#1133)
Reentrancy in CToken.transferTokens(address,address,address,uint256) (contracts/CToken.sol#93-154):
  External calls:
  - allowed = controller.transferAllowed(address(this),src,dst,tokens) (contracts/CToken.sol#96)
  State variables written after the call(s):
  - accountTokens[src] = srcTokensNew (contracts/CToken.sol#138)
  - accountTokens[dst] = dstTokensNew (contracts/CToken.sol#139)
  - transferAllowances[src][spender] = allowanceNew (contracts/CToken.sol#143)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
Reentrancy in CToken._reduceReserves(uint256) (contracts/CToken.sol#1285-1293):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#1286)
    - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#1285)
  - controller._beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller._afterNonReentrant() (contracts/CToken.sol#1582)
  Event emitted after the call(s):
  - AccrueInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowNew) (contracts/CToken.sol#478)

```

```
- error = accrueInterest() (contracts/CToken.sol#1286)
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
- fail(TokenErrorReporter.Error(error),FailureInfo.REVOKE_RESERVE_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#1289)
- Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
- _reduceReserveFresh(reduceAmount) (contracts/CToken.sol#1292)
- ReserveSuccess(msg.sender,reduceAmount,totalReserveNew) (contracts/CToken.sol#1338)
- _reduceReserveFresh(reduceAmount) (contracts/CToken.sol#1292)
Reentrancy in CToken_setAdminFee(uint256) (contracts/CToken.sol#1176-1184):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#1177)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#1176)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  Event emitted after the call(s):
  - AccruedInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/CToken.sol#478)
  - error = accrueInterest() (contracts/CToken.sol#1177)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(TokenErrorReporter.Error(error),FailureInfo.SET_ADMIN_FEE_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#1180)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - _setAdminFeeFresh(newAdminFeeMantissa) (contracts/CToken.sol#1183)
  - NewAdminFee(oldAdminFeeMantissa,newAdminFeeMantissa) (contracts/CToken.sol#1228)
  - _setAdminFeeFresh(newAdminFeeMantissa) (contracts/CToken.sol#1183)
  - NewAdminFee(oldAdminFeeMantissa,newAdminFeeMantissa) (contracts/CToken.sol#1238)
  - _setAdminFeeFresh(newAdminFeeMantissa) (contracts/CToken.sol#1183)
Reentrancy in CToken_setInterestRateModel(InterestRateModel) (contracts/CToken.sol#1463-1471):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#1464)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  Event emitted after the call(s):
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(TokenErrorReporter.Error(error),FailureInfo.SET_INTEREST_RATE_MODEL_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#1467)
  Reentrancy in CToken_setInterestRateModel(InterestRateModel) (contracts/CToken.sol#1463-1471):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#1464)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - _setInterestRateModelFresh(newInterestRateModel) (contracts/CToken.sol#1478)
  - address(oldInterestRateModel).call(abi.encodeWithSignature(reserveInterestCheckpoint())) (contracts/CToken.sol#1586)
  - address(newInterestRateModel).call(abi.encodeWithSignature(checkpointInterest())) (contracts/CToken.sol#1589)
  Event emitted after the call(s):
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - _setInterestRateModelFresh(newInterestRateModel) (contracts/CToken.sol#1478)
  - NewMarketInterestRateModel(oldInterestRateModel,newInterestRateModel) (contracts/CToken.sol#1583)
  - _setInterestRateModelFresh(newInterestRateModel) (contracts/CToken.sol#1478)
Reentrancy in CToken_setReserveFactor(uint256) (contracts/CToken.sol#1245-1249):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#1246)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#1245)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  Event emitted after the call(s):
  - AccruedInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/CToken.sol#478)
  - error = accrueInterest() (contracts/CToken.sol#1246)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - _setReserveFactorFresh(newReserveFactorMantissa) (contracts/CToken.sol#1248)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(TokenErrorReporter.Error(error),FailureInfo.SET_RESERVE_FACTOR_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#1246)
  - NewReserveFactor(oldReserveFactorMantissa,newReserveFactorMantissa) (contracts/CToken.sol#1275)
  - _setReserveFactorFresh(newReserveFactorMantissa) (contracts/CToken.sol#1248)
  Reentrancy in CToken_withdrawAdminFees(uint256) (contracts/CToken.sol#1404-1412):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#1405)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#1404)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  Event emitted after the call(s):
  - AccruedInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/CToken.sol#478)
  - error = accrueInterest() (contracts/CToken.sol#1405)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(TokenErrorReporter.Error(error),FailureInfo.WITHDRAW_ADMIN_FEES_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#1408)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - _withdrawAdminFeesFresh(withdrawAmount) (contracts/CToken.sol#1411)
  Reentrancy in CToken_withdrawAdminFees(uint256) (contracts/CToken.sol#1404-1412):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#1405)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#1404)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  Event emitted after the call(s):
  - AccruedInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/CToken.sol#478)
  - error = accrueInterest() (contracts/CToken.sol#1405)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(TokenErrorReporter.Error(error),FailureInfo.WITHDRAW_ADMIN_FEES_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#1408)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - _withdrawAdminFeesFresh(withdrawAmount) (contracts/CToken.sol#1411)
  Reentrancy in CToken_borrowBalanceCurrent(address) (contracts/CToken.sol#288-291):
  External calls:
  - require(bool,string)(accrueInterest() == uint256(Error.NO_ERROR),accrue interest failed) (contracts/CToken.sol#289)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#288)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  Event emitted after the call(s):
  - AccruedInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/CToken.sol#478)
  - require(bool,string)(accrueInterest() == uint256(Error.NO_ERROR),accrue interest failed) (contracts/CToken.sol#289)
  Reentrancy in CToken_borrowFresh(address,uint256) (contracts/CToken.sol#755-827):
  External calls:
  - allowed = controller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#757)
  Event emitted after the call(s):
  - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
  - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.BORROW_COMPTROLLER_REJECTION,allowed) (contracts/CToken.sol#759)
  - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
  - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/CToken.sol#788)
  - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
  - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/CToken.sol#783)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(Error.MONEY_NOT_FRESH,FailureInfo.BORROW_FRESHNESS_CHECK) (contracts/CToken.sol#764)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.BORROW_CASH_NOT_AVAILABLE) (contracts/CToken.sol#771)
  Reentrancy in CToken_borrowFresh(address,uint256) (contracts/CToken.sol#755-827):
  External calls:
  - allowed = controller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/CToken.sol#757)
  - allowed = controller.borrowWithinLimits(address(this),vars.accountBorrowsNew) (contracts/CToken.sol#792)
  Event emitted after the call(s):
  - Borrow(borrower,borrowAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/CToken.sol#820)
  - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
  - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.BORROW_COMPTROLLER_REJECTION,allowed) (contracts/CToken.sol#794)
  - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#235)
  - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/CToken.sol#799)
  Reentrancy in CToken_borrowInternal(uint256) (contracts/CToken.sol#733-741):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#734)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#733)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
  Event emitted after the call(s):
  - AccruedInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/CToken.sol#478)
  - error = accrueInterest() (contracts/CToken.sol#734)
  - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#226)
  - fail(TokenErrorReporter.Error(error),FailureInfo.BORROW_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#737)
  Reentrancy in CToken_borrowInternal(uint256) (contracts/CToken.sol#733-741):
  External calls:
  - error = accrueInterest() (contracts/CToken.sol#734)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256),borrowRateMantissa)) (contracts/CToken.sol#473)
  - borrowFresh(msg.sender,borrowAmount) (contracts/CToken.sol#740)
  - allowed = controller.borrowWithinLimits(address(this),borrower,borrowAmount) (contracts/CToken.sol#767)
  - allowed = controller.borrowWithinLimits(address(this),vars.accountBorrowsNew) (contracts/CToken.sol#792)
  - nonReentrant(false) (contracts/CToken.sol#733)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
```

```
Event emitted after the call(s):
- Borrow(borrower, borrowAmount, vars.accountBorrowNew, vars.totalBorrowNew) (contracts/CToken.sol#420)
- borrowFresh(msg.sender, borrowAmount) (contracts/CToken.sol#424)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - borrowFresh(msg.sender, borrowAmount) (contracts/CToken.sol#424)
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
  - borrowFresh(msg.sender, borrowAmount) (contracts/CToken.sol#424)
Reentrancy in CToken.exchangeRateCurrent() (contracts/CToken.sol#345-348):
  External calls:
  - require(bool, string)(accruedInterest() == uint256(Error_NO_ERROR), accrued interest failed) (contracts/CToken.sol#346)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#348)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- AccruedInterest(cashPrior, interestAccumulated, borrowIndexNew, totalBorrowNew) (contracts/CToken.sol#470)
  - require(bool, string)(accruedInterest() == uint256(Error_NO_ERROR), accrued interest failed) (contracts/CToken.sol#346)
Reentrancy in CToken.initialize(ControllerInterface, interestRateModel, uint256, string, string, uint8, uint256, uint256) (contracts/CToken.sol#94-75):
  External calls:
  - err = setInterestRateModelFresh(interestRateModel) (contracts/CToken.sol#95)
  - address(newInterestRateModel).call(abi.encodeWithSignature(checkpointInterest())) (contracts/CToken.sol#1586)
  - address(newInterestRateModel).call(abi.encodeWithSignature(checkpointInterest())) (contracts/CToken.sol#1589)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - err = setReserveFactorFresh(reserveFactorMantissa) (contracts/CToken.sol#66)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - err = setAdminFeeFresh(adminFeeMantissa) (contracts/CToken.sol#78)
- NewAdminFee(oldAdminFeeMantissa, newAdminFeeMantissa) (contracts/CToken.sol#1220)
  - err = setAdminFeeFresh(adminFeeMantissa) (contracts/CToken.sol#78)
- NewFeeForOldUserMantissa, newFeeMantissa) (contracts/CToken.sol#1228)
  - err = setAdminFeeFresh(adminFeeMantissa) (contracts/CToken.sol#78)
- NewReserveFactor(oldReserveFactorMantissa, newReserveFactorMantissa) (contracts/CToken.sol#1275)
  - err = setReserveFactorFresh(reserveFactorMantissa) (contracts/CToken.sol#66)
Reentrancy in CToken.liquidateBorrowFresh(address, address, uint256, CTokenInterface) (contracts/CToken.sol#981-1050):
  External calls:
  - allowed = controller.liquidateBorrowAllowed(address(this), address(cTokenCollateral), liquidator, borrower, repayAmount) (contracts/CToken.sol#983)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - fail(Error_MARKET_NOT_FRESH, FailureInfo.LIQUIDATE_FRESHNESS_CHECK, 0) (contracts/CToken.sol#998)
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
  - fail(OpacityError.COMPTROLLER_REJECTION, FailureInfo.LIQUIDATE_COMPTROLLER_REJECTION, allowed, 0) (contracts/CToken.sol#988)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - fail(Error_MARKET_NOT_FRESH, FailureInfo.LIQUIDATE_COLLATERAL_FRESHNESS_CHECK, 0) (contracts/CToken.sol#995)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - fail(Error_INVALID_CLOSE_AMOUNT_REQUESTED, FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_UINT_MAX, 0) (contracts/CToken.sol#1010)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - fail(Error_INVALID_CLOSE_AMOUNT_REQUESTED, FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_ZERO, 0) (contracts/CToken.sol#1005)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - fail(Error_INVALID_ACCOUNT_PAIR, FailureInfo.LIQUIDATE_LIQUIDATOR_IS_BORROWER, 0) (contracts/CToken.sol#1008)
Reentrancy in CToken.liquidateBorrowFresh(address, address, uint256, CTokenInterface) (contracts/CToken.sol#981-1050):
  External calls:
  - allowed = controller.liquidateBorrowAllowed(address(this), address(cTokenCollateral), liquidator, borrower, repayAmount) (contracts/CToken.sol#983)
  - (repayBorrowError.actualRepayAmount) = repayBorrowFresh(liquidator, borrower, repayAmount) (contracts/CToken.sol#1015)
  - allowed = controller.repayBorrowAllowed(address(this), payer, borrower, repayAmount) (contracts/CToken.sol#888)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - fail(OpacityError.COMPTROLLER_REJECTION, FailureInfo.LIQUIDATE_REPAY_BORROW_FRESH_FAILED, 0) (contracts/CToken.sol#1017)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - (repayBorrowError.actualRepayAmount) = repayBorrowFresh(liquidator, borrower, repayAmount) (contracts/CToken.sol#1015)
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
  - (repayBorrowError.actualRepayAmount) = repayBorrowFresh(liquidator, borrower, repayAmount) (contracts/CToken.sol#981)
Reentrancy in CToken.liquidateBorrowFresh(address, address, uint256, CTokenInterface) (contracts/CToken.sol#981-1050):
  External calls:
  - allowed = controller.liquidateBorrowAllowed(address(this), address(cTokenCollateral), liquidator, borrower, repayAmount) (contracts/CToken.sol#983)
  - (repayBorrowError.actualRepayAmount) = repayBorrowFresh(liquidator, borrower, repayAmount) (contracts/CToken.sol#1015)
  - allowed = controller.repayBorrowAllowed(address(this), payer, borrower, repayAmount) (contracts/CToken.sol#888)
  - seizeError = seizeInternal(address(this), liquidator, borrower, seizeTokens) (contracts/CToken.sol#1034)
  - allowed = controller.seizeAllowed(address(this), seizerToken, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1089)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - seizeError = seizeInternal(address(this), liquidator, borrower, seizeTokens) (contracts/CToken.sol#1034)
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
  - seizeError = seizeInternal(address(this), liquidator, borrower, seizeTokens) (contracts/CToken.sol#1034)
- ReserveesAdded(address(this), vars.protocolSeizeAmount, vars.totalReserveesNew) (contracts/CToken.sol#1140)
  - seizeError = seizeInternal(address(this), liquidator, borrower, seizeTokens) (contracts/CToken.sol#1034)
- Transfer(borrower, liquidator, vars.liquidatorSeizeTokens) (contracts/CToken.sol#1130)
  - seizeError = seizeInternal(address(this), liquidator, borrower, seizeTokens) (contracts/CToken.sol#1034)
- Transfer(borrower, address(this), vars.protocolSeizeTokens) (contracts/CToken.sol#1130)
  - seizeError = seizeInternal(address(this), liquidator, borrower, seizeTokens) (contracts/CToken.sol#1034)
Reentrancy in CToken.liquidateBorrowFresh(address, address, uint256, CTokenInterface) (contracts/CToken.sol#981-1050):
  External calls:
  - allowed = controller.liquidateBorrowAllowed(address(this), address(cTokenCollateral), liquidator, borrower, repayAmount) (contracts/CToken.sol#983)
  - (repayBorrowError.actualRepayAmount) = repayBorrowFresh(liquidator, borrower, repayAmount) (contracts/CToken.sol#1015)
  - allowed = controller.repayBorrowAllowed(address(this), payer, borrower, repayAmount) (contracts/CToken.sol#888)
  - seizeError = seizeInternal(address(this), liquidator, borrower, seizeTokens) (contracts/CToken.sol#1034)
  - allowed = controller.seizeAllowed(address(this), seizerToken, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1089)
  - seizeError = seizeInternal(address(this), seizerToken, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1034)
Event emitted after the call(s):
- LiquidateBorrow(liquidator, borrower, actualRepayAmount, address(cTokenCollateral), seizeTokens) (contracts/CToken.sol#1043)
Reentrancy in CToken.liquidateBorrowInternal(address, uint256, CTokenInterface) (contracts/CToken.sol#956-978):
  External calls:
  - error = accruedInterest() (contracts/CToken.sol#956)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
  - nonReentrant(false) (contracts/CToken.sol#958)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- AccruedInterest(cashPrior, interestAccumulated, borrowIndexNew, totalBorrowNew) (contracts/CToken.sol#470)
  - err = accruedInterest() (contracts/CToken.sol#956)
  - Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - fail(Error_ERROR_REPORTER_ERROR, FailureInfo.LIQUIDATE_ACCRUE_BORROW_INTEREST_FAILED, 0) (contracts/CToken.sol#959)
Reentrancy in CToken.liquidateBorrowInternal(address, uint256, CTokenInterface) (contracts/CToken.sol#956-978):
  External calls:
  - error = accruedInterest() (contracts/CToken.sol#956)
  - error = cTokenCollateral.accruedInterest() (contracts/CToken.sol#962)
  - error = nonReentrant(false) (contracts/CToken.sol#958)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - fail(Error_ERROR_REPORTER_ERROR, FailureInfo.LIQUIDATE_ACCRUE_COLLATERAL_INTEREST_FAILED, 0) (contracts/CToken.sol#965)
Reentrancy in CToken.liquidateBorrowInternal(address, uint256, CTokenInterface) (contracts/CToken.sol#956-978):
  External calls:
  - error = accruedInterest() (contracts/CToken.sol#956)
  - address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
  - liquidateBorrowFresh(msg.sender, borrower, repayAmount, cTokenCollateral) (contracts/CToken.sol#969)
  - allowed = controller.liquidateBorrowAllowed(address(this), address(cTokenCollateral), liquidator, borrower, repayAmount) (contracts/CToken.sol#983)
  - allowed = controller.seizeAllowed(address(this), seizerToken, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1089)
  - allowed = controller.repayBorrowAllowed(address(this), payer, borrower, repayAmount) (contracts/CToken.sol#888)
  - seizeError = cTokenCollateral.seize(liquidator, borrower, seizeTokens) (contracts/CToken.sol#1036)
  - nonReentrant(false) (contracts/CToken.sol#958)
  - controller_beforeNonReentrant() (contracts/CToken.sol#1571)
  - controller_afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
  - liquidateBorrowFresh(msg.sender, borrower, repayAmount, cTokenCollateral) (contracts/CToken.sol#969)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
  - liquidateBorrowFresh(msg.sender, borrower, repayAmount, cTokenCollateral) (contracts/CToken.sol#969)
- LiquidateBorrow(liquidator, borrower, actualRepayAmount, address(cTokenCollateral), seizeTokens) (contracts/CToken.sol#1043)
  - liquidateBorrowFresh(msg.sender, borrower, repayAmount, cTokenCollateral) (contracts/CToken.sol#969)
- RepayBorrow(payer, borrower, vars.actualRepayAmount, vars.accountBorrowNew, vars.totalBorrowNew) (contracts/CToken.sol#938)
  - liquidateBorrowFresh(msg.sender, borrower, repayAmount, cTokenCollateral) (contracts/CToken.sol#969)
- ReserveesAdded(address(this), vars.protocolSeizeAmount, vars.totalReserveesNew) (contracts/CToken.sol#1140)
  - liquidateBorrowFresh(msg.sender, borrower, repayAmount, cTokenCollateral) (contracts/CToken.sol#969)
- Transfer(borrower, liquidator, vars.liquidatorSeizeTokens) (contracts/CToken.sol#1130)
  - liquidateBorrowFresh(msg.sender, borrower, repayAmount, cTokenCollateral) (contracts/CToken.sol#969)
- Transfer(borrower, address(this), vars.protocolSeizeTokens) (contracts/CToken.sol#1130)
  - liquidateBorrowFresh(msg.sender, borrower, repayAmount, cTokenCollateral) (contracts/CToken.sol#969)
Reentrancy in CToken.liquidateBorrowInternal(address, uint256, CTokenInterface) (contracts/CToken.sol#956-978):
  External calls:
```

```

- allowed = comptroller.mintAllowed(address(this), minter, mintAmount) (contracts/CToken.sol#513)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(Error.MANAGER_NOT_FRESH, FailureInfo.MINT_FRESHNESS_CHECK_0, 0) (contracts/CToken.sol#520)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(Error.COMPTROLLER_REJECTION, FailureInfo.MINT_COMPTROLLER_REJECTION, allowed, 0) (contracts/CToken.sol#515)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(Error.MATH_ERROR, FailureInfo.MINT_DCDMMG_RATE_READ_FAILED, uint256(vars.mathErr), 0) (contracts/CToken.sol#527)
- Mint(minter, vars.actualMintAmount, vars.mintTokens) (contracts/CToken.sol#473)
- Transfer(address(this), minter, vars.mintTokens) (contracts/CToken.sol#487)
Reentrancy in CToken.mintInternal(uint256) (contracts/CToken.sol#484-492):
External calls:
- error = accrueInterest() (contracts/CToken.sol#485)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
- nonReentrant(false) (contracts/CToken.sol#484)
- comptroller._beforeNonReentrant() (contracts/CToken.sol#1571)
- comptroller._afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- AccrueInterest(cashPrior, interestAccumulated, borrowIndexNew, totalBorrowsNew) (contracts/CToken.sol#470)
- error = accrueInterest() (contracts/CToken.sol#485)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(TokenErrorReporter.Error(error), FailureInfo.MINT_ACCRUE_INTEREST_FAILED, 0) (contracts/CToken.sol#488)
Reentrancy in CToken.mintInternal(uint256) (contracts/CToken.sol#484-492):
External calls:
- error = accrueInterest() (contracts/CToken.sol#485)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
- mintFresh(msg.sender, mintAmount) (contracts/CToken.sol#491)
- allowed = comptroller.mintAllowed(address(this), minter, mintAmount) (contracts/CToken.sol#513)
- comptroller._mintVerify(address(this), minter, vars.actualMintAmount, vars.mintTokens) (contracts/CToken.sol#577)
- nonReentrant(false) (contracts/CToken.sol#484)
- comptroller._beforeNonReentrant() (contracts/CToken.sol#1571)
- comptroller._afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- mintFresh(msg.sender, mintAmount) (contracts/CToken.sol#491)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- mintFresh(msg.sender, mintAmount) (contracts/CToken.sol#491)
- Mint(minter, vars.actualMintAmount, vars.mintTokens) (contracts/CToken.sol#473)
- mintFresh(msg.sender, mintAmount) (contracts/CToken.sol#491)
- Transfer(address(this), minter, vars.mintTokens) (contracts/CToken.sol#487)
Reentrancy in CToken.redemFresh(address, uint256, uint256) (contracts/CToken.sol#632-726):
External calls:
- allowed = comptroller.redemAllowed(address(this), redeemer, vars.redemTokens) (contracts/CToken.sol#672)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(Error.MANAGER_NOT_FRESH, FailureInfo.REDEM_FRESHNESS_CHECK, 0) (contracts/CToken.sol#679)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(Error.COMPTROLLER_REJECTION, FailureInfo.REDEM_COMPTROLLER_REJECTION, allowed) (contracts/CToken.sol#674)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(Error.TOKEN_INSUFFICIENT_CASH, FailureInfo.REDEM_TRANSFER_OUT_NOT_POSSIBLE) (contracts/CToken.sol#699)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(Error.ERROR, FailureInfo.REDEM_NEW_TOTM_SUPPLY_CALCULATION_FAILED, uint256(vars.mathErr)) (contracts/CToken.sol#689)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(Error.MATH_ERROR, FailureInfo.REDEM_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED, uint256(vars.mathErr)) (contracts/CToken.sol#696)
- Redeem(redeemer, vars.redemAmount, vars.redemTokens) (contracts/CToken.sol#720)
- Transfer(redeemer, address(this), vars.redemTokens) (contracts/CToken.sol#719)
Reentrancy in CToken.redemInternal(uint256) (contracts/CToken.sol#588-596):
External calls:
- error = accrueInterest() (contracts/CToken.sol#589)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
- nonReentrant(false) (contracts/CToken.sol#588)
- comptroller._beforeNonReentrant() (contracts/CToken.sol#1571)
- comptroller._afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- AccrueInterest(cashPrior, interestAccumulated, borrowIndexNew, totalBorrowsNew) (contracts/CToken.sol#470)
- error = accrueInterest() (contracts/CToken.sol#589)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(TokenErrorReporter.Error(error), FailureInfo.REDEM_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#592)
Reentrancy in CToken.redemInternal(uint256) (contracts/CToken.sol#588-596):
External calls:
- error = accrueInterest() (contracts/CToken.sol#589)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
- allowed = comptroller.redemAllowed(address(this), redeemer, vars.redemTokens) (contracts/CToken.sol#672)
- comptroller._redeemVerify(address(this), redeemer, vars.redemAmount, vars.redemTokens) (contracts/CToken.sol#723)
- nonReentrant(false) (contracts/CToken.sol#588)
- comptroller._beforeNonReentrant() (contracts/CToken.sol#1571)
- comptroller._afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- redeemFresh(msg.sender, redeemTokens, 0) (contracts/CToken.sol#595)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- redeemFresh(msg.sender, redeemTokens, 0) (contracts/CToken.sol#595)
- Redeem(redeemer, vars.redemAmount, vars.redemTokens) (contracts/CToken.sol#720)
- redeemFresh(msg.sender, redeemTokens, 0) (contracts/CToken.sol#595)
- Transfer(redeemer, address(this), vars.redemTokens) (contracts/CToken.sol#719)
- redeemFresh(msg.sender, redeemTokens, 0) (contracts/CToken.sol#595)
Reentrancy in CToken.redemUnderlyingInternal(uint256) (contracts/CToken.sol#684-612):
External calls:
- error = accrueInterest() (contracts/CToken.sol#685)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
- nonReentrant(false) (contracts/CToken.sol#684)
- comptroller._beforeNonReentrant() (contracts/CToken.sol#1571)
- comptroller._afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- AccrueInterest(cashPrior, interestAccumulated, borrowIndexNew, totalBorrowsNew) (contracts/CToken.sol#470)
- error = accrueInterest() (contracts/CToken.sol#685)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(TokenErrorReporter.Error(error), FailureInfo.REDEM_ACCRUE_INTEREST_FAILED) (contracts/CToken.sol#688)
Reentrancy in CToken.redemUnderlyingInternal(uint256) (contracts/CToken.sol#684-612):
External calls:
- error = accrueInterest() (contracts/CToken.sol#685)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
- redeemFresh(msg.sender, 0, redeemAmount) (contracts/CToken.sol#611)
- allowed = comptroller.redemAllowed(address(this), redeemer, vars.redemTokens) (contracts/CToken.sol#672)
- comptroller._redeemVerify(address(this), redeemer, vars.redemAmount, vars.redemTokens) (contracts/CToken.sol#723)
- nonReentrant(false) (contracts/CToken.sol#684)
- comptroller._beforeNonReentrant() (contracts/CToken.sol#1571)
- comptroller._afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- redeemFresh(msg.sender, 0, redeemAmount) (contracts/CToken.sol#611)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- redeemFresh(msg.sender, 0, redeemAmount) (contracts/CToken.sol#611)
- Redeem(redeemer, vars.redemAmount, vars.redemTokens) (contracts/CToken.sol#720)
- redeemFresh(msg.sender, 0, redeemAmount) (contracts/CToken.sol#611)
- Transfer(redeemer, address(this), vars.redemTokens) (contracts/CToken.sol#719)
- redeemFresh(msg.sender, 0, redeemAmount) (contracts/CToken.sol#611)
Reentrancy in CToken.repayBorrowBehalfInternal(address, uint256) (contracts/CToken.sol#850-858):
External calls:
- error = accrueInterest() (contracts/CToken.sol#851)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
- nonReentrant(false) (contracts/CToken.sol#850)
- comptroller._beforeNonReentrant() (contracts/CToken.sol#1571)
- comptroller._afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- AccrueInterest(cashPrior, interestAccumulated, borrowIndexNew, totalBorrowsNew) (contracts/CToken.sol#470)
- error = accrueInterest() (contracts/CToken.sol#851)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(TokenErrorReporter.Error(error), FailureInfo.REPAY_BORROW_SELF_ACCRUE_INTEREST_FAILED, 0) (contracts/CToken.sol#854)
Reentrancy in CToken.repayBorrowBehalfInternal(address, uint256) (contracts/CToken.sol#850-858):
External calls:
- error = accrueInterest() (contracts/CToken.sol#851)
- address(InterestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#473)
- repayBorrowFresh(msg.sender, borrower, repayAmount) (contracts/CToken.sol#857)
- allowed = comptroller.repayBorrowAllowed(address(this), payer, borrower, repayAmount) (contracts/CToken.sol#888)
- nonReentrant(false) (contracts/CToken.sol#850)
- comptroller._beforeNonReentrant() (contracts/CToken.sol#1571)
- comptroller._afterNonReentrant() (contracts/CToken.sol#1582)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- repayBorrowFresh(msg.sender, borrower, repayAmount) (contracts/CToken.sol#857)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)

```

```

- repayBorrowFresh(msg.sender, borrower, repayAmount) (contracts/CToken.sol#857)
- repayBorrow(payer, borrower, vars.actualRepayAmount, vars.accountBorrowNew, vars.totalBorrowNew) (contracts/CToken.sol#938)
- repayBorrowFresh(msg.sender, borrower, repayAmount) (contracts/CToken.sol#857)
Reentrancy in CToken.repayBorrowFresh(address,address,uint256) (contracts/CToken.sol#878-945):
External calls:
- allowed = controller.repayBorrowAllowed(address(this), payer, borrower, repayAmount) (contracts/CToken.sol#888)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(Error.MATH_ERROR, FailureInfo.REPAY_BORROW_FRESH_CHECK, 0) (contracts/CToken.sol#887)
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
- failOpaque(Error.MATH_ERROR, FailureInfo.REPAY_BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED, uint256(vars.mathErr), 0) (contracts/CToken.sol#898)
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
- failOpaque(Error.COMPTROLLER_REJECTION, FailureInfo.REPAY_BORROW_COMPROLLER_REJECTION, allowed), 0) (contracts/CToken.sol#882)
- repayBorrow(payer, borrower, vars.actualRepayAmount, vars.accountBorrowNew, vars.totalBorrowNew) (contracts/CToken.sol#938)
Reentrancy in CToken.repayBorrowInternal(uint256) (contracts/CToken.sol#833-862):
External calls:
- error = accrueInterest() (contracts/CToken.sol#833)
- address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#673)
- nonReentrant(false) (contracts/CToken.sol#834)
- controller.beforeNonReentrant() (contracts/CToken.sol#571)
- controller.afterNonReentrant() (contracts/CToken.sol#582)
Event emitted after the call(s):
- AccrueInterest(cashPrior, interestAccumulated, borrowIndexNew, totalBorrowNew) (contracts/CToken.sol#479)
- error = accrueInterest() (contracts/CToken.sol#833)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(TokenErrorReporter.Error(error), FailureInfo.REPAY_BORROW_ACCRUE_INTEREST_FAILED, 0) (contracts/CToken.sol#858)
Reentrancy in CToken.repayBorrowInternal(uint256) (contracts/CToken.sol#833-862):
External calls:
- error = accrueInterest() (contracts/CToken.sol#833)
- address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#673)
- repayBorrowFresh(msg.sender, msg.sender, repayAmount) (contracts/CToken.sol#841)
- allowed = controller.repayBorrowAllowed(address(this), payer, borrower, repayAmount) (contracts/CToken.sol#888)
- nonReentrant(false) (contracts/CToken.sol#834)
- controller.beforeNonReentrant() (contracts/CToken.sol#571)
- controller.afterNonReentrant() (contracts/CToken.sol#582)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- repayBorrowFresh(msg.sender, msg.sender, repayAmount) (contracts/CToken.sol#841)
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
- repayBorrowFresh(msg.sender, msg.sender, repayAmount) (contracts/CToken.sol#841)
- repayBorrow(payer, borrower, vars.actualRepayAmount, vars.accountBorrowNew, vars.totalBorrowNew) (contracts/CToken.sol#938)
- repayBorrowFresh(msg.sender, msg.sender, repayAmount) (contracts/CToken.sol#841)
Reentrancy in CToken.seize(address,address,uint256) (contracts/CToken.sol#1861-1863):
External calls:
- seizeInternal(msg.sender, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1862)
- allowed = controller.seizeAllowed(address(this), seizerToken, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1889)
- nonReentrant(true) (contracts/CToken.sol#1861)
- controller.beforeNonReentrant() (contracts/CToken.sol#571)
- controller.afterNonReentrant() (contracts/CToken.sol#582)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- seizeInternal(msg.sender, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1862)
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
- seizeInternal(msg.sender, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1862)
- ReserveAdded(address(this), vars.protocolSeizeAmount, vars.totalReservesNew) (contracts/CToken.sol#1548)
- seizeInternal(msg.sender, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1862)
- Transfer(borrower, liquidator, vars.liquidatorSeizeTokens) (contracts/CToken.sol#1138)
- seizeInternal(msg.sender, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1862)
- Transfer(borrower, liquidator, vars.liquidatorSeizeTokens) (contracts/CToken.sol#1139)
- seizeInternal(msg.sender, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1862)
- seizeInternal(msg.sender, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1862)
Reentrancy in CToken.seizeInternal(address,address,address,uint256) (contracts/CToken.sol#1087-1147):
External calls:
- allowed = controller.seizeAllowed(address(this), seizerToken, liquidator, borrower, seizeTokens) (contracts/CToken.sol#1089)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
- failOpaque(Error.MATH_ERROR, FailureInfo.LIQUIDATE_SEIZE_BALANCE_INCREMENT_FAILED, uint256(vars.mathErr)) (contracts/CToken.sol#1188)
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
- failOpaque(Error.MATH_ERROR, FailureInfo.LIQUIDATE_SEIZE_BALANCE_INCREMENT_FAILED, uint256(vars.mathErr)) (contracts/CToken.sol#1124)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(Error.INVALID_ACCOUNT_PAIR, FailureInfo.LIQUIDATE_SEIZE_LIQUIDATOR_IS_BORROWER) (contracts/CToken.sol#1096)
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
- failOpaque(Error.COMPTROLLER_REJECTION, FailureInfo.LIQUIDATE_SEIZE_COMPROLLER_REJECTION, allowed) (contracts/CToken.sol#1091)
- ReserveAdded(address(this), vars.protocolSeizeAmount, vars.totalReservesNew) (contracts/CToken.sol#1148)
- Transfer(borrower, liquidator, vars.liquidatorSeizeTokens) (contracts/CToken.sol#1138)
- Transfer(borrower, address(this), vars.protocolSeizeTokens) (contracts/CToken.sol#1139)
Reentrancy in CToken.totalBorrowCurrent() (contracts/CToken.sol#278-281):
External calls:
- require(bool,string)(accrueInterest() == uint256(Error.NO_ERROR), accrue interest failed) (contracts/CToken.sol#279)
- address(interestRateModel).call(abi.encodeWithSignature(checkpointInterest(uint256), borrowRateMantissa)) (contracts/CToken.sol#673)
- nonReentrant(false) (contracts/CToken.sol#278)
- controller.beforeNonReentrant() (contracts/CToken.sol#571)
- controller.afterNonReentrant() (contracts/CToken.sol#582)
Event emitted after the call(s):
- AccrueInterest(cashPrior, interestAccumulated, borrowIndexNew, totalBorrowNew) (contracts/CToken.sol#479)
- require(bool,string)(accrueInterest() == uint256(Error.NO_ERROR), accrue interest failed) (contracts/CToken.sol#279)
Reentrancy in CToken.transfer(address,uint256) (contracts/CToken.sol#162-164):
External calls:
- transferTokens(msg.sender, msg.sender, dst, amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#163)
- allowed = controller.transferAllowed(address(this), src, dst, tokens) (contracts/CToken.sol#495)
- nonReentrant(false) (contracts/CToken.sol#162)
- controller.beforeNonReentrant() (contracts/CToken.sol#571)
- controller.afterNonReentrant() (contracts/CToken.sol#582)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- transferTokens(msg.sender, msg.sender, dst, amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#163)
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
- transferTokens(msg.sender, msg.sender, dst, amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#163)
- Transfer(src, dst, tokens) (contracts/CToken.sol#147)
- transferTokens(msg.sender, msg.sender, dst, amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#163)
Reentrancy in CToken.transferFrom(address,address,uint256) (contracts/CToken.sol#173-175):
External calls:
- transferTokens(msg.sender, src, dst, amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#174)
- allowed = controller.transferAllowed(address(this), src, dst, tokens) (contracts/CToken.sol#495)
- nonReentrant(false) (contracts/CToken.sol#173)
- controller.beforeNonReentrant() (contracts/CToken.sol#571)
- controller.afterNonReentrant() (contracts/CToken.sol#582)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
- transferTokens(msg.sender, src, dst, amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#174)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- transferTokens(msg.sender, src, dst, amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#174)
- Transfer(src, dst, tokens) (contracts/CToken.sol#147)
- transferTokens(msg.sender, src, dst, amount) == uint256(Error.NO_ERROR) (contracts/CToken.sol#174)
Reentrancy in CToken.transferTokens(address,address,address,uint256) (contracts/CToken.sol#93-164):
External calls:
- allowed = controller.transferAllowed(address(this), src, dst, tokens) (contracts/CToken.sol#95)
Event emitted after the call(s):
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(Error.MATH_ERROR, FailureInfo.TRANSFER_TOO_MUCH) (contracts/CToken.sol#131)
- Failure(uint256(err), uint256(info), opaqueError) (contracts/ErrorReporter.sol#235)
- failOpaque(Error.COMPTROLLER_REJECTION, FailureInfo.TRANSFER_COMPROLLER_REJECTION, allowed) (contracts/CToken.sol#97)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(Error.MATH_ERROR, FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/CToken.sol#121)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(Error.MATH_ERROR, FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/CToken.sol#126)
- Failure(uint256(err), uint256(info), 0) (contracts/ErrorReporter.sol#226)
- fail(Error.BAD_INPUT, FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/CToken.sol#182)
- Transfer(src, dst, tokens) (contracts/CToken.sol#147)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
CToken._functionCall(address,bytes,string) (contracts/CToken.sol#1596-1615) uses assembly
CToken._INLINE_ASM (contracts/CToken.sol#1485-1488)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#assembly-usage
CToken._functionCall(address,bytes,string) (contracts/CToken.sol#1596-1615) is never used and should be removed
CToken.borrowFresh(address,uint256) (contracts/CToken.sol#758-827) is never used and should be removed
CToken.borrowInternal(uint256) (contracts/CToken.sol#733-742) is never used and should be removed
CToken.doTransfer(address,uint256) (contracts/CToken.sol#2543) is never used and should be removed
CToken.liquidateBorrowFresh(address,address,uint256,CTokenInterface) (contracts/CToken.sol#981-1050) is never used and should be removed
CToken.liquidateBorrowInternal(address,address,uint256,CTokenInterface) (contracts/CToken.sol#955-970) is never used and should be removed
CToken.mintFresh(address,uint256) (contracts/CToken.sol#1511-1580) is never used and should be removed
CToken.mintInternal(uint256) (contracts/CToken.sol#484-492) is never used and should be removed
CToken.redemFresh(address,uint256) (contracts/CToken.sol#629-726) is never used and should be removed
CToken.redemInternal(uint256) (contracts/CToken.sol#588-596) is never used and should be removed

```

```

CToken_redeemUnderlyingInternal(uint256) (contracts/CToken.sol#604-612) is never used and should be removed
CToken_repayBorrowBehalfInternal(address,uint256) (contracts/CToken.sol#858-858) is never used and should be removed
CToken_repayBorrowBehalfInternal(address,uint256) (contracts/CToken.sol#858-858) is never used and should be removed
CToken_repayBorrowInternal(uint256) (contracts/CToken.sol#834-842) is never used and should be removed
ComptrollerErrorReporter_fall(ComptrollerErrorReporter_Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#69-73) is never used and should be removed
ComptrollerErrorReporter_fallInternal(ComptrollerErrorReporter_Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#78-82) 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#108-102) is never used and should be removed
Exponential_divScalarByExp(uint256,ExponentialNoError_Exp) (contracts/Exponential.sol#93-98) 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_mulExp(ExponentialNoError_Exp,ExponentialNoError_Exp) (contracts/Exponential.sol#135-151) 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_mulScalarTruncate(uint256,ExponentialNoError_Exp,uint256) (contracts/Exponential.sol#178-203) 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#191-193) is never used and should be removed
ExponentialNoError_double_(ExponentialNoError_Double,ExponentialNoError_Double) (contracts/ExponentialNoError.sol#187-189) is never used and should be removed
ExponentialNoError_div_(ExponentialNoError_Double,ExponentialNoError_Double) (contracts/ExponentialNoError.sol#171-173) is never used and should be removed
ExponentialNoError_div_(ExponentialNoError_Double,uint256) (contracts/ExponentialNoError.sol#175-177) is never used and should be removed
ExponentialNoError_div_(ExponentialNoError_Exp,ExponentialNoError_Exp) (contracts/ExponentialNoError.sol#159-163) is never used and should be removed
ExponentialNoError_div_(ExponentialNoError_Exp,uint256) (contracts/ExponentialNoError.sol#163-166) is never used and should be removed
ExponentialNoError_div_(uint256,ExponentialNoError_Double) (contracts/ExponentialNoError.sol#179-181) is never used and should be removed
ExponentialNoError_div_(uint256,ExponentialNoError_Exp) (contracts/ExponentialNoError.sol#167-169) is never used and should be removed
ExponentialNoError_div_(uint256,uint256) (contracts/ExponentialNoError.sol#183-185) is never used and should be removed
ExponentialNoError_div_(uint256,uint256,string) (contracts/ExponentialNoError.sol#187-190) is never used and should be removed
Function CToken_fractoin(uint256,uint256) (contracts/CToken.sol#194) is never used and should be removed
ExponentialNoError_greaterThanExp(ExponentialNoError_Exp,ExponentialNoError_Exp) (contracts/ExponentialNoError.sol#166-68) is never used and should be removed
ExponentialNoError_isZeroExp(ExponentialNoError_Exp) (contracts/ExponentialNoError.sol#179-179) is never used and should be removed
ExponentialNoError_lessThanExp(ExponentialNoError_Exp,ExponentialNoError_Exp) (contracts/ExponentialNoError.sol#185-185) is never used and should be removed
ExponentialNoError_lessThanEqualExp(ExponentialNoError_Exp,ExponentialNoError_Exp) (contracts/ExponentialNoError.sol#189-191) is never used and should be removed
ExponentialNoError_mul_(ExponentialNoError_Double,ExponentialNoError_Double) (contracts/ExponentialNoError.sol#124-124) 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#122-124) is never used and should be removed
ExponentialNoError_mul_(ExponentialNoError_Exp,uint256) (contracts/ExponentialNoError.sol#126-126) is never used and should be removed
ExponentialNoError_safe224(uint256,string) (contracts/ExponentialNoError.sol#177-180) is never used and should be removed
ExponentialNoError_safe32(uint256,string) (contracts/ExponentialNoError.sol#182-185) is never used and should be removed
ExponentialNoError_sub_(ExponentialNoError_Double,ExponentialNoError_Double) (contracts/ExponentialNoError.sol#189-191) is never used and should be removed
ExponentialNoError_sub_(ExponentialNoError_Exp,ExponentialNoError_Exp) (contracts/ExponentialNoError.sol#185-187) is never used and should be removed
UnitrollerAdminStorage_hasAdminRights() (contracts/ComptrollerStorage.sol#36-38) is never used and should be removed
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#dead-code

Low level call in CToken_finishInterestAccrual(uint256,uint256,uint256,uint256) (contracts/CToken.sol#437-474):
- address(interestModel).call(abi.encodeWithSignature("borrowMaxMantissa")) (contracts/CToken.sol#473)
Low level call in CToken_setInterestRateModelFresh(InterestRateModel) (contracts/CToken.sol#1670-1512):
- address(interestRateModel).call(abi.encodeWithSignature("resetInterestCheckpoints()")) (contracts/CToken.sol#1586)
- address(interestRateModel).call(abi.encodeWithSignature("setInterestRateModel()")) (contracts/CToken.sol#1589)
Low level call in CToken_functionCall(address,bytes,string) (contracts/CToken.sol#1596-1615):
- (success,returnData) = target.call(data) (contracts/CToken.sol#1597)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#low-level-calls

Function CToken_setAdminFee(uint256) (contracts/CToken.sol#176-186) is not in mixedCase
Function CToken_setInterestRateModel(uint256) (contracts/CToken.sol#1215-1249) is not in mixedCase
Function CToken_reduceReserve(uint256) (contracts/CToken.sol#1286-1293) is not in mixedCase
Function CToken_withdrawFees(uint256) (contracts/CToken.sol#1348-1356) is not in mixedCase
Function CToken_withdrawFeesInternal(uint256) (contracts/CToken.sol#1418-1423) is not in mixedCase
Function CToken_setInterestRateModel(InterestRateModel) (contracts/CToken.sol#1463-1471) is not in mixedCase
Function CToken_setNameAndSymbol(string,string) (contracts/CToken.sol#1521-1528) is not in mixedCase
Parameter CToken_setNameAndSymbol(string,string).name (contracts/CToken.sol#1521) is not in mixedCase
Parameter CToken_setNameAndSymbol(string,string).symbol (contracts/CToken.sol#1521) is not in mixedCase
Constant CTokenAdminStorage_fuseAdmin (contracts/CTokenInterfaces.sol#112) is not in UPPER_CASE_WITH_UNDERSCORES
Variable CTokenStorage_notEntered (contracts/CTokenInterfaces.sol#129) is not in mixedCase
Constant CTokenStorage_borrowRateMaxMantissa (contracts/CTokenInterfaces.sol#139) is not in UPPER_CASE_WITH_UNDERSCORES
Constant CTokenStorage_reserveFactorPlusFeesMaxMantissa (contracts/CTokenInterfaces.sol#144) is not in UPPER_CASE_WITH_UNDERSCORES
Variable CTokenStorage_notEntered (contracts/CTokenInterfaces.sol#129) is not in mixedCase
Constant CTokenStorage_protocolSeizeShareMantissa (contracts/CTokenInterfaces.sol#144) is not in UPPER_CASE_WITH_UNDERSCORES
Function CTokenInterface_setReserveFactor(uint256) (contracts/CTokenInterfaces.sol#267) is not in mixedCase
Function CTokenInterface_setReserveFactor(uint256) (contracts/CTokenInterfaces.sol#266) is not in mixedCase
Function CTokenInterface_setInterestRateModel(InterestRateModel) (contracts/CTokenInterfaces.sol#249) is not in mixedCase
Constant CTokenInterface_isCToken (contracts/CTokenInterfaces.sol#151) is not in UPPER_CASE_WITH_UNDERSCORES
Constant CTokenInterface_isCether (contracts/CTokenInterfaces.sol#154) is not in UPPER_CASE_WITH_UNDERSCORES
Constant CTokenInterface_isCether (contracts/CTokenInterfaces.sol#1297) is not in UPPER_CASE_WITH_UNDERSCORES
Function CDelegatedInterface_becomeImplementation(bytes) (contracts/CTokenInterfaces.sol#326) is not in mixedCase
Function CDelegatedInterface_prepare(bytes) (contracts/CTokenInterfaces.sol#332) is not in mixedCase
Function ComptrollerInterface_beforeNonReentrant() (contracts/ComptrollerInterface.sol#276) is not in mixedCase
Function ComptrollerInterface_afterNonReentrant() (contracts/ComptrollerInterface.sol#277) is not in mixedCase
Constant ComptrollerStorage_admin (contracts/ComptrollerStorage.sol#11) is not in UPPER_CASE_WITH_UNDERSCORES
Constant UnitrollerAdminStorage_fuseAdmin (contracts/ComptrollerStorage.sol#11) is not in UPPER_CASE_WITH_UNDERSCORES
Variable ComptrollerV3Storage_minGuardianPaused (contracts/ComptrollerStorage.sol#145) is not in mixedCase
Variable ComptrollerV3Storage_notEnteredPaused (contracts/ComptrollerStorage.sol#146) is not in mixedCase
Variable ComptrollerV3Storage_notEntered (contracts/ComptrollerStorage.sol#172) is not in mixedCase
Variable ComptrollerV3Storage_notEnteredInitialized (contracts/ComptrollerStorage.sol#175) is not in mixedCase
Function ExponentialNoError_mul_ScalarTruncateAddUnit(ExponentialNoError_Exp,uint256,uint256) (contracts/ExponentialNoError.sol#44-47) is not in mixedCase
Constant ExponentialNoError_expScale (contracts/ExponentialNoError.sol#1) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError_expScale (contracts/ExponentialNoError.sol#2) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError_halfExpScale (contracts/ExponentialNoError.sol#11) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ExponentialNoError_mantissaOne (contracts/ExponentialNoError.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant InterestRateModel_interestRateModel (contracts/InterestRateModel.sol#1) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PriceOracle_isPriceOracle (contracts/PriceOracle.sol#7) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable CToken_seize(address,address,uint256).seizeTokens (contracts/CToken.sol#1041) is too similar to CToken_seizeInternal(address,address,uint256).seizeToken (contracts/CToken.sol#1007)
Variable CToken_seize(address,address,uint256).seizeTokens (contracts/CToken.sol#1007) is too similar to CToken_seizeInternal(address,address,uint256).seizeToken (contracts/CToken.sol#1007)
Variable CTokenInterface_seize(address,address,uint256).seizeTokens (contracts/CTokenInterfaces.sol#202) is too similar to CToken_seizeInternal(address,address,uint256).seizeToken (contracts/CToken.sol#1007)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#variable-names-are-too-similar

CToken (contracts/CToken.sol#16-1616) does not implement functions:
- CToken.doTransferIn(address,uint256) (contracts/CToken.sol#1543)
- CToken.doTransferOut(address,uint256) (contracts/CToken.sol#1558)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#unimplemented-functions

CTokenStorage_pendingAdmin (contracts/CTokenInterfaces.sol#69) is never used in CToken (contracts/CToken.sol#16-1616)
ComptrollerV3Storage_maxAssets (contracts/ComptrollerStorage.sol#70) is never used in ComptrollerV3Storage (contracts/ComptrollerStorage.sol#153-176)
ComptrollerV3Storage_borrowerIndexes (contracts/ComptrollerStorage.sol#117) is never used in ComptrollerV3Storage (contracts/ComptrollerStorage.sol#153-176)
ComptrollerV3Storage_whitelistIndexes (contracts/ComptrollerStorage.sol#137) is never used in ComptrollerV3Storage (contracts/ComptrollerStorage.sol#153-176)
ComptrollerV3Storage_notEntered (contracts/ComptrollerStorage.sol#172) is never used in ComptrollerV3Storage (contracts/ComptrollerStorage.sol#153-176)
ComptrollerV3Storage_notEnteredInitialized (contracts/ComptrollerStorage.sol#175) is never used in ComptrollerV3Storage (contracts/ComptrollerStorage.sol#153-176)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#unused-state-variables

CDelegationStorage_implementation (contracts/CTokenInterfaces.sol#304) should be constant
CDelegatedStorage_underlying (contracts/CTokenInterfaces.sol#276) should be constant
CTokenStorage_pendingAdmin (contracts/CTokenInterfaces.sol#69) should be constant
ComptrollerV3Storage_classFactorMantissa (contracts/ComptrollerStorage.sol#60) should be constant
ComptrollerV3Storage_maxAssets (contracts/ComptrollerStorage.sol#70) should be constant
ComptrollerV3Storage_borrowGuardianPaused (contracts/ComptrollerStorage.sol#146) should be constant
ComptrollerV3Storage_minGuardianPaused (contracts/ComptrollerStorage.sol#145) should be constant
ComptrollerV3Storage_notEnteredPaused (contracts/ComptrollerStorage.sol#146) should be constant
ComptrollerV3Storage_notEntered (contracts/ComptrollerStorage.sol#172) should be constant
ComptrollerV3Storage_pausedGuardian (contracts/ComptrollerStorage.sol#144) should be constant
ComptrollerV3Storage_reserveFactor (contracts/ComptrollerStorage.sol#148) should be constant
ComptrollerV3Storage_reserveFactorPaused (contracts/ComptrollerStorage.sol#149) should be constant
ComptrollerV3Storage_notEntered (contracts/ComptrollerStorage.sol#172) should be constant
ComptrollerV3Storage_borrowCapGuardian (contracts/ComptrollerStorage.sol#140) should be constant
UnitrollerAdminStorage_admin (contracts/ComptrollerStorage.sol#16) should be constant
UnitrollerAdminStorage_adminRights (contracts/ComptrollerStorage.sol#21) should be constant
UnitrollerAdminStorage_comptrollerImplementation (contracts/ComptrollerStorage.sol#43) should be constant
UnitrollerAdminStorage_fuseAdminRights (contracts/ComptrollerStorage.sol#26) should be constant
UnitrollerAdminStorage_pendingAdmin (contracts/ComptrollerStorage.sol#22) should be constant
UnitrollerAdminStorage_pendingComptrollerImplementation (contracts/ComptrollerStorage.sol#40) should be constant
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8,uint256,uint256) should be declared external:
- CToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8,uint256,uint256) (contracts/CToken.sol#134-175)
- CTokenInterface_initialize(ComptrollerInterface,InterestRateModel) should be declared external:
- CToken_setInterestRateModel(InterestRateModel) (contracts/CToken.sol#1463-1471)
- CTokenInterface_setInterestRateModel(InterestRateModel) (contracts/CTokenInterfaces.sol#269)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#public-function-that-could-be-declared-external

```

CurvePlainPoolOracle.sol

CurvePlainPoolOracle.registerCurvePool(address), coin (contracts/oracles/CurvePlainPoolOracle.sol#24) is a local variable never initialized
 Reference: <https://github.com/crytic/sliether/wiki/Detector-Documents#uninitialized-local-variables>

CurvePlainPoolOracle.registerCurvePool(address) (contracts/oracles/CurvePlainPoolOracle.sol#17-34) ignores return value by !CurvePool(lpToken).coins(i) (contracts/oracles/CurvePlainPoolOracle.sol#24-28)
 Reference: <https://github.com/crytic/sliether/wiki/Detector-Documents#unused-return>

CurvePlainPoolOracle.registerCurvePool(address) (contracts/oracles/CurvePlainPoolOracle.sol#17-34) has external calls inside a loop: !CurvePool(lpToken).coins(i) (contracts/oracles/CurvePlainPoolOracle.sol#24-28)
 Reference: <https://github.com/crytic/sliether/wiki/Detector-Documents#calls-inside-a-loop>

Variable 'CurvePlainPoolOracle.registerCurvePool(address)', coin (contracts/oracles/CurvePlainPoolOracle.sol#24) in CurvePlainPoolOracle.registerCurvePool(address) (contracts/oracles/CurvePlainPoolOracle.sol#17-34)
 Token.push_coin (contracts/oracles/CurvePlainPoolOracle.sol#25)
 Reference: <https://github.com/crytic/sliether/wiki/Detector-Documents#pre-declaration-usage-of-local-variables>

Different versions of Solidity is used:

- Version used: ["0.8.4", "0.8.0", "0.8.4"]
- 0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4)
- 0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
- 0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4)
- 0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
- 0.8.4 (contracts/external/latest/interfaces.sol#2)
- 0.8.4 (contracts/external/latest/curve/ICurvePool.sol#2)
- 0.8.4 (contracts/oracles/CurvePlainPoolOracle.sol#2)

Reference: <https://github.com/crytic/sliether/wiki/Detector-Documents#different-pragma-directives-are-used>

Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
 ERC20._burn(address, uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#280-295) is never used and should be removed
 ERC20._mint(address, uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#257-267) is never used and should be removed
 Reference: <https://github.com/crytic/sliether/wiki/Detector-Documents#dead-code>

Pragma version0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4) allow old versions
 Pragma version0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4) allows old versions
 Pragma version0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4) allows old versions
 Reference: <https://github.com/crytic/sliether/wiki/Detector-Documents#incorrect-version-of-solidity>

Function Comptroller._setWhiteListEnforcement(bool) (contracts/external/latest/interfaces.sol#56) is not in mixedCase
 Function Comptroller._setWhiteListStatuses(address[], bool[]) (contracts/external/latest/interfaces.sol#56-59) is not in mixedCase
 Function Comptroller._become(address) (contracts/external/latest/interfaces.sol#61) is not in mixedCase
 Function Comptroller._setPauserRole(address) (contracts/external/latest/interfaces.sol#63) is not in mixedCase
 Function Comptroller._setCollateralFactor(address, uint256) (contracts/external/latest/interfaces.sol#69-72) is not in mixedCase
 Function Comptroller._acceptAdmin() (contracts/external/latest/interfaces.sol#83) is not in mixedCase
 Function Comptroller._setCollateralFactor(uint256) (contracts/external/latest/interfaces.sol#85-87) is not in mixedCase
 Function Comptroller._setLiquidationIncentive(uint256) (contracts/external/latest/interfaces.sol#89-91) is not in mixedCase
 Function Comptroller._supportMarket(address) (contracts/external/latest/interfaces.sol#93) is not in mixedCase
 Function Comptroller._setPauseBorrowCaps(address[], uint256[]) (contracts/external/latest/interfaces.sol#95-98) is not in mixedCase
 Function Comptroller._setMarketSupplyCaps(address[], uint256[]) (contracts/external/latest/interfaces.sol#100-103) is not in mixedCase
 Function Comptroller._setMintPaused(address, bool) (contracts/external/latest/interfaces.sol#105) is not in mixedCase
 Function Comptroller._setPauseGuardian(address) (contracts/external/latest/interfaces.sol#107-109) is not in mixedCase
 Function Comptroller._setTransferPaused(bool) (contracts/external/latest/interfaces.sol#111) is not in mixedCase
 Function Comptroller._setSeizePaused(bool) (contracts/external/latest/interfaces.sol#113) is not in mixedCase
 Function Comptroller._setReserveDistributor(address) (contracts/external/latest/interfaces.sol#115-117) is not in mixedCase
 Function Comptroller._deployMarket(bool, bytes, uint256) (contracts/external/latest/interfaces.sol#119-123) is not in mixedCase
 Function Comptroller._unsupportMarket(CToken) (contracts/external/latest/interfaces.sol#125) is not in mixedCase
 Function Comptroller._setPauseGuardian(address) (contracts/external/latest/interfaces.sol#127-129) is not in mixedCase
 Function Comptroller._setBorrowCapGuardian(address) (contracts/external/latest/interfaces.sol#131) is not in mixedCase
 Function CToken._supportMarketAndSetCollateralFactor(CToken, uint256) (contracts/external/latest/interfaces.sol#184-187) is not in mixedCase
 Function CToken._setAdminFee(uint256) (contracts/external/latest/interfaces.sol#189-193) is not in mixedCase
 Function CToken._setReserveFactor(uint256) (contracts/external/latest/interfaces.sol#193-195) is not in mixedCase
 Function CToken._setInterestRateModel(address) (contracts/external/latest/interfaces.sol#197-199) is not in mixedCase
 Function CToken._setMinCollateralRatio(string, string) (contracts/external/latest/interfaces.sol#201-202) is not in mixedCase
 Function CToken._withdrawAdminFees(uint256) (contracts/external/latest/interfaces.sol#204-206) is not in mixedCase
 Function CToken._reduceReserves(uint256) (contracts/external/latest/interfaces.sol#208) is not in mixedCase
 Function IUniswapV2Router01.WETH() (contracts/external/latest/interfaces.sol#154) is not in mixedCase
 Function IUniswapV2Pair.DOMAIN_SEPARATOR() (contracts/external/latest/interfaces.sol#50) is not in mixedCase
 Function IUniswapV2Pair.PERMIT_TYPEHASH() (contracts/external/latest/interfaces.sol#52) is not in mixedCase
 Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (contracts/external/latest/interfaces.sol#53) is not in mixedCase
 Function ICurvePool.get_virtual_price() (contracts/external/latest/curve/ICurvePool.sol#6) is not in mixedCase
 Function ICurvePool.remove_liquidity_one_coin(uint256, int128, uint256) (contracts/external/latest/curve/ICurvePool.sol#9-13) is not in mixedCase
 Function ICurvePool.remove_liquidity_one_coin(uint256, int128, uint256, bool) (contracts/external/latest/curve/ICurvePool.sol#15-20) is not in mixedCase
 Reference: <https://github.com/crytic/sliether/wiki/Detector-Documents#conformance-to-solidity-naming-conventions>

Variable IUniswapV2Router01.addLiquidity(address, address, uint256, uint256, uint256, address, uint256).amountDesired (contracts/external/latest/interfaces.sol#319) is too similar to IUniswapV2Router01.addLiquidity(address, address, uint256) (contracts/external/latest/interfaces.sol#320)
 Reference: <https://github.com/crytic/sliether/wiki/Detector-Documents#variable-names-are-too-similar>

name() should be declared external:

- ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#62-64)

symbol() should be declared external:

- ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)

decimals() should be declared external:

- ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#87-89)

totalSupply() should be declared external:

- ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#94-96)

balanceOf(address) should be declared external:

- ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#101-103)

transfer(address, uint256) should be declared external:

- ERC20.transfer(address, uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#113-117)

approve(address, uint256) should be declared external:

- ERC20.approve(address, uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#136-140)

transferFrom(address, address, uint256) should be declared external:

- ERC20.transferFrom(address, address, uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#158-167)

increaseAllowance(address, uint256) should be declared external:

- ERC20.increaseAllowance(address, uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#181-185)

decreaseAllowance(address, uint256) should be declared external:

- ERC20.decreaseAllowance(address, uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#201-210)

Reference: <https://github.com/crytic/sliether/wiki/Detector-Documents#public-function-that-could-be-declared-external>

FusePoolDirectory.sol

```

DisableUpgradeable_gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#74) shadow:
  Contextual module @openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#81
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#state-variable-shadowing

FusePoolDirectory.deployPool(string,address,bool,uint256,uint256,address) (contracts/FusePoolDirectory.sol#146-187) uses a dangerous strict equality:
  - require(bool,string)(unitroller._setPendingImplementation(implementation) == 0,Failed to set pending implementation on Unitroller.) (contracts/FusePoolDirectory.sol#166)
FusePoolDirectory.deployPool(string,address,bool,uint256,uint256,address) (contracts/FusePoolDirectory.sol#146-187) uses a dangerous strict equality:
  - require(bool,string)(controllerProxy._setCloseFactor(closeFactor) == 0,Failed to set pool close factor.) (contracts/FusePoolDirectory.sol#171)
FusePoolDirectory.deployPool(string,address,bool,uint256,uint256,address) (contracts/FusePoolDirectory.sol#146-187) uses a dangerous strict equality:
  - require(bool,string)(controllerProxy._setLiquidityIncentive(liquidityIncentive) == 0,Failed to set pool liquidity incentive.) (contracts/FusePoolDirectory.sol#172)
FusePoolDirectory.deployPool(string,address,bool,uint256,uint256,address) (contracts/FusePoolDirectory.sol#146-187) uses a dangerous strict equality:
  - require(bool,string)(controllerProxy._setPriceOracle(priceOracle) == 0,Failed to set pool price oracle.) (contracts/FusePoolDirectory.sol#173)
FusePoolDirectory.deployPool(string,address,bool,uint256,uint256,address) (contracts/FusePoolDirectory.sol#146-187) uses a dangerous strict equality:
  - require(bool,string)(controllerProxy._toggleAutoImplementation(true) == 0,Failed to enable pool auto implementations.) (contracts/FusePoolDirectory.sol#179)
FusePoolDirectory.deployPool(string,address,bool,uint256,uint256,address) (contracts/FusePoolDirectory.sol#146-187) uses a dangerous strict equality:
  - require(bool,string)(Unitroller(proxy)._setPendingAdmin(controllerAdmin) == 0,Failed to set pending admin on Unitroller.) (contracts/FusePoolDirectory.sol#203)
FusePoolDirectory.setPoolName(uint256,string) (contracts/FusePoolDirectory.sol#272-276) uses a dangerous strict equality:
  - require(bool)(msg.sender == controller.admin() && controller.admin().hasControllerRights() || msg.sender == owner()) (contracts/FusePoolDirectory.sol#274)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#dangerous-strict-equality

FusePoolDirectory.getPublicPoolsByVerification(bool).admin_scope_3 (contracts/FusePoolDirectory.sol#321) is a local variable never initialized
FusePoolDirectory.getPublicPoolsByVerification(bool).admin (contracts/FusePoolDirectory.sol#303) is a local variable never initialized
FusePoolDirectory.getPublicPoolsByVerification(bool).enforceWhitelist (contracts/FusePoolDirectory.sol#300) is a local variable never initialized
FusePoolDirectory.getPublicPools().enforceWhitelist_scope_3 (contracts/FusePoolDirectory.sol#292) is a local variable never initialized
FusePoolDirectory.getPublicPools().enforceWhitelist (contracts/FusePoolDirectory.sol#230) is a local variable never initialized
FusePoolDirectory.getPublicPoolsByVerification(bool).enforceWhitelist_scope_2 (contracts/FusePoolDirectory.sol#318) is a local variable never initialized
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#local-variables

FusePoolDirectory.getPublicPools() (contracts/FusePoolDirectory.sol#226-252) ignores return value by controller[pool[i]].controller.enforceWhitelist() (contracts/FusePoolDirectory.sol#230-232)
FusePoolDirectory.getPublicPools() (contracts/FusePoolDirectory.sol#226-252) ignores return value by controller[pool[s_scope_0]].controller.enforceWhitelist() (contracts/FusePoolDirectory.sol#242-244)
FusePoolDirectory.getPublicPoolsByVerification(bool) (contracts/FusePoolDirectory.sol#294-332) ignores return value by controller.enforceWhitelist() (contracts/FusePoolDirectory.sol#300-306)
FusePoolDirectory.getPublicPoolsByVerification(bool) (contracts/FusePoolDirectory.sol#294-332) ignores return value by controller.admin() (contracts/FusePoolDirectory.sol#303-305)
FusePoolDirectory.getPublicPoolsByVerification(bool) (contracts/FusePoolDirectory.sol#294-332) ignores return value by controller_scope_1.enforceWhitelist() (contracts/FusePoolDirectory.sol#318-324)
FusePoolDirectory.getPublicPoolsByVerification(bool) (contracts/FusePoolDirectory.sol#294-332) ignores return value by controller_scope_1.admin() (contracts/FusePoolDirectory.sol#321-323)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#unused-return

FusePoolDirectory.getPublicPools() (contracts/FusePoolDirectory.sol#226-252) has external calls inside a loop: Cmpcontroller[pool[s]].controller.enforceWhitelist() (contracts/FusePoolDirectory.sol#230-232)
FusePoolDirectory.getPublicPoolsByVerification(bool) (contracts/FusePoolDirectory.sol#294-332) has external calls inside a loop: controller.enforceWhitelist() (contracts/FusePoolDirectory.sol#242-244)
FusePoolDirectory.getPublicPoolsByVerification(bool) (contracts/FusePoolDirectory.sol#294-332) has external calls inside a loop: controller.admin() (contracts/FusePoolDirectory.sol#303-305)
FusePoolDirectory.getPublicPoolsByVerification(bool) (contracts/FusePoolDirectory.sol#294-332) has external calls inside a loop: controller_scope_1.enforceWhitelist() (contracts/FusePoolDirectory.sol#318-324)
FusePoolDirectory.getPublicPoolsByVerification(bool) (contracts/FusePoolDirectory.sol#294-332) has external calls inside a loop: controller_scope_1.admin() (contracts/FusePoolDirectory.sol#321-323)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#calls-inside-a-loop

Variable 'FusePoolDirectory.getPublicPools().enforceWhitelist (contracts/FusePoolDirectory.sol#230)' in FusePoolDirectory.getPublicPools() (contracts/FusePoolDirectory.sol#226-252) potentially used before declared
Variable 'FusePoolDirectory.getPublicPools().enforceWhitelist_scope_1 (contracts/FusePoolDirectory.sol#242)' in FusePoolDirectory.getPublicPools() (contracts/FusePoolDirectory.sol#226-252) potentially used before declared
Variable 'FusePoolDirectory.getPublicPools().admin (contracts/FusePoolDirectory.sol#303)' in FusePoolDirectory.getPublicPoolsByVerification(bool) (contracts/FusePoolDirectory.sol#294-332) potentially used before declared
Variable 'FusePoolDirectory.getPublicPoolsByVerification(bool).admin (contracts/FusePoolDirectory.sol#303)' in FusePoolDirectory.getPublicPoolsByVerification(bool) (contracts/FusePoolDirectory.sol#294-332) potentially used before declared
Variable 'FusePoolDirectory.getPublicPoolsByVerification(bool).enforceWhitelist_scope_2 (contracts/FusePoolDirectory.sol#318)' in FusePoolDirectory.getPublicPoolsByVerification(bool) (contracts/FusePoolDirectory.sol#294-332) potentially used before declared
Variable 'FusePoolDirectory.getPublicPoolsByVerification(bool).admin_scope_3 (contracts/FusePoolDirectory.sol#321)' in FusePoolDirectory.getPublicPoolsByVerification(bool) (contracts/FusePoolDirectory.sol#294-332) potentially used before declared
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in FusePoolDirectory.deployPool(string,address,bool,uint256,uint256,address) (contracts/FusePoolDirectory.sol#146-187):
  External calls:
  - require(bool,string)(unitroller._setPendingImplementation(implementation) == 0,Failed to set pending implementation on Unitroller.) (contracts/FusePoolDirectory.sol#166)
  - controllerImplementation._become(controller) (contracts/FusePoolDirectory.sol#167)
  - require(bool,string)(controllerProxy._setCloseFactor(closeFactor) == 0,Failed to set pool close factor.) (contracts/FusePoolDirectory.sol#171)
  - require(bool,string)(controllerProxy._setLiquidityIncentive(liquidityIncentive) == 0,Failed to set pool liquidity incentive.) (contracts/FusePoolDirectory.sol#172)
  - require(bool,string)(controllerProxy._setPriceOracle(priceOracle) == 0,Failed to set pool price oracle.) (contracts/FusePoolDirectory.sol#173)
  - require(bool,string)(controllerProxy._setWhitelistEnforcement(true) == 0,Failed to enforce supplier/borrower whitelist.) (contracts/FusePoolDirectory.sol#176)
  - require(bool,string)(controllerProxy._toggleAutoImplementation(true) == 0,Failed to enable pool auto implementations.) (contracts/FusePoolDirectory.sol#179)
  - finishDeployPool(proxy) (contracts/FusePoolDirectory.sol#183)
  - require(bool,string)(Unitroller(proxy)._setPendingAdmin(controllerAdmin) == 0,Failed to set pending admin on Unitroller.) (contracts/FusePoolDirectory.sol#203)
  - _marketAdmin(controllerAdmin,acceptAdmin) (contracts/FusePoolDirectory.sol#207)
  - controllerAdmin._transferAdminDeployer(adminDeployer).deploy(proxy,msg.sender) (contracts/FusePoolDirectory.sol#208)
  State variables written after the call(s):
  - _registerPool(name,proxy,proxy) (contracts/FusePoolDirectory.sol#186)
  - _poolByAccess[msg.sender].push(pool,length - 1) (contracts/FusePoolDirectory.sol#190)
  - _registerPool(name,proxy,proxy) (contracts/FusePoolDirectory.sol#186)
  - poolExists(controller) == true (contracts/FusePoolDirectory.sol#181)
  - _registerPool(name,proxy,proxy) (contracts/FusePoolDirectory.sol#186)
  - pools.push(pool) (contracts/FusePoolDirectory.sol#189)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in FusePoolDirectory.deployPool(string,address,bool,uint256,uint256,address) (contracts/FusePoolDirectory.sol#146-187):
  External calls:
  - require(bool,string)(unitroller._setPendingImplementation(implementation) == 0,Failed to set pending implementation on Unitroller.) (contracts/FusePoolDirectory.sol#166)
  - controllerImplementation._become(controller) (contracts/FusePoolDirectory.sol#167)
  - require(bool,string)(controllerProxy._setCloseFactor(closeFactor) == 0,Failed to set pool close factor.) (contracts/FusePoolDirectory.sol#171)
  - require(bool,string)(controllerProxy._setLiquidityIncentive(liquidityIncentive) == 0,Failed to set pool liquidity incentive.) (contracts/FusePoolDirectory.sol#172)
  - require(bool,string)(controllerProxy._setPriceOracle(priceOracle) == 0,Failed to set pool price oracle.) (contracts/FusePoolDirectory.sol#173)
  - require(bool,string)(controllerProxy._setWhitelistEnforcement(true) == 0,Failed to enforce supplier/borrower whitelist.) (contracts/FusePoolDirectory.sol#176)
  - require(bool,string)(controllerProxy._toggleAutoImplementation(true) == 0,Failed to enable pool auto implementations.) (contracts/FusePoolDirectory.sol#179)
  - finishDeployPool(proxy) (contracts/FusePoolDirectory.sol#183)
  - require(bool,string)(Unitroller(proxy)._setPendingAdmin(controllerAdmin) == 0,Failed to set pending admin on Unitroller.) (contracts/FusePoolDirectory.sol#203)
  - _marketAdmin(controllerAdmin,acceptAdmin) (contracts/FusePoolDirectory.sol#207)
  - controllerAdmin._transferAdminDeployer(adminDeployer).deploy(proxy,msg.sender) (contracts/FusePoolDirectory.sol#208)
  Event emitted after the call(s):
  - PoolRegistered(pool,length - 1,pool) (contracts/FusePoolDirectory.sol#193)
  - _registerPool(name,proxy,proxy) (contracts/FusePoolDirectory.sol#186)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

FusePoolDirectory.getPublicPools() (contracts/FusePoolDirectory.sol#226-252) uses timestamp for comparisons
  - 1 < pools.length (contracts/FusePoolDirectory.sol#229)
  - 1 < scope_0 < pools.length (contracts/FusePoolDirectory.sol#241)
FusePoolDirectory.setPoolName(uint256,string) (contracts/FusePoolDirectory.sol#272-276) uses timestamp for comparisons
  - require(bool)(msg.sender == controller.admin() && controller.admin().hasRights() || msg.sender == owner()) (contracts/FusePoolDirectory.sol#274)
FusePoolDirectory.getPublicPoolsByVerification(bool) (contracts/FusePoolDirectory.sol#294-332) uses timestamp for comparisons
  - 1 < pools.length (contracts/FusePoolDirectory.sol#297)
  - 1 < scope_0 < pools.length (contracts/FusePoolDirectory.sol#315)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#block-timestamp

AddressUpgradeable.isContract(address) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#26-35) uses assembly
  - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#33)
AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#147-144) uses assembly
  - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#150-150)
FusePoolDirectory.deployPool(string,address,bool,uint256,uint256,address) (contracts/FusePoolDirectory.sol#146-187) uses assembly
  - INLINE ASM (contracts/FusePoolDirectory.sol#156-161)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
  - Version used: '0.6.12', '>=0.4.24<0.8.0', '>=0.6.0<0.8.0', '>=0.6.24<0.8.0'
  - >=0.4.24<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#8)
  - >=0.4.24<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/Initializable.sol#4)
  - >=0.6.24<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#3)
  - >=0.4.24<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#3)
  - 0.6.12 (contracts/FusePoolDirectory.sol#2)
  - ABIEncoderV2 (contracts/FusePoolDirectory.sol#3)
  - 0.6.12 (contracts/external/compound/Compound.sol#2)
  - 0.6.12 (contracts/external/compound/Compound.sol#2)
  - 0.6.12 (contracts/external/compound/PriceOracle.sol#2)
  - 0.6.12 (contracts/external/compound/RebaseDistributor.sol#2)
  - 0.6.12 (contracts/external/compound/Unitroller.sol#2)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#different-pragmas-directives-are-used

AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#147-144) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#179-81) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#180-92) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#180-104) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#181-112) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#129-121) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#139-146) is never used and should be removed

```

```

AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/upgradeable/AddressUpgradeable.sol#53-59) is never used and should be removed
ContextUpgradeable._context_init() (node_modules/@openzeppelin/contracts-upgradeable/upgradeable/ContextUpgradeable.sol#11-19) is never used and should be removed
ContextUpgradeable._msgData() (node_modules/@openzeppelin/contracts-upgradeable/upgradeable/ContextUpgradeable.sol#27-30) is never used and should be removed
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#dead-code

Pragma version=6.0.0 (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#3) is too complex
Pragma version=4.24.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/Initializable.sol#4) is too complex
Pragma version=6.2.0 (node_modules/@openzeppelin/contracts-upgradeable/upgradeable/AddressUpgradeable.sol#3) is too complex
Pragma version=6.0.0 (node_modules/@openzeppelin/contracts-upgradeable/upgradeable/ContextUpgradeable.sol#1) is too complex
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/upgradeable/AddressUpgradeable.sol#53-59):
- (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts-upgradeable/upgradeable/AddressUpgradeable.sol#57)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/upgradeable/AddressUpgradeable.sol#114-121):
- (success, returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/upgradeable/AddressUpgradeable.sol#115)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/upgradeable/AddressUpgradeable.sol#139-145):
- (success, returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts-upgradeable/upgradeable/AddressUpgradeable.sol#143)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#low-level-calls

Function OwnableUpgradeable._Ownable_init() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#27-30) is not in mixedCase
Function OwnableUpgradeable._Ownable_init_unchecked() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#32-36) is not in mixedCase
Variable OwnableUpgradeable._gas (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#74) is not in mixedCase
Function ContextUpgradeable._Context_init() (node_modules/@openzeppelin/contracts-upgradeable/upgradeable/ContextUpgradeable.sol#11-19) is not in mixedCase
Function ContextUpgradeable._Context_init_unchecked() (node_modules/@openzeppelin/contracts-upgradeable/upgradeable/ContextUpgradeable.sol#21-22) is not in mixedCase
Variable ContextUpgradeable._gas (node_modules/@openzeppelin/contracts-upgradeable/upgradeable/ContextUpgradeable.sol#31) is not in mixedCase
Parameter FusePoolDirectory.initialize(bool,address[]) (node_modules/@openzeppelin/contracts-upgradeable/contracts/FusePoolDirectory.sol#88) is not in mixedCase
Function FusePoolDirectory._setDeployerWhitelistEnforcement(bool) (node_modules/@openzeppelin/contracts-upgradeable/contracts/FusePoolDirectory.sol#104-106) is not in mixedCase
Function FusePoolDirectory._editDeployerWhitelist(address[]) (node_modules/@openzeppelin/contracts-upgradeable/contracts/FusePoolDirectory.sol#113-116) is not in mixedCase
Function FusePoolDirectory._editAdminWhitelist(address[]) (node_modules/@openzeppelin/contracts-upgradeable/contracts/FusePoolDirectory.sol#206-208) is not in mixedCase
Function Comptroller._setPriceOracle(PriceOracle) (node_modules/@openzeppelin/contracts-upgradeable/contracts/external/Comptroller.sol#28) is not in mixedCase
Function Comptroller._setLossFactor(uint256) (node_modules/@openzeppelin/contracts-upgradeable/contracts/external/Comptroller.sol#29) is not in mixedCase
Function Comptroller._setLiquidationIncentive(uint256) (node_modules/@openzeppelin/contracts-upgradeable/contracts/external/Comptroller.sol#38) is not in mixedCase
Function Comptroller._become(Untrroller) (node_modules/@openzeppelin/contracts-upgradeable/contracts/external/Comptroller.sol#41) is not in mixedCase
Function Comptroller._setWhitelistEnforcement(bool) (node_modules/@openzeppelin/contracts-upgradeable/contracts/external/Comptroller.sol#42) is not in mixedCase
Function Comptroller._toggleWhitelistStatuses(address[]) (node_modules/@openzeppelin/contracts-upgradeable/contracts/external/Comptroller.sol#43) is not in mixedCase
Function Comptroller._toggleAutoImplementations(bool) (node_modules/@openzeppelin/contracts-upgradeable/contracts/external/Comptroller.sol#45) is not in mixedCase
Function Untrroller._setPendingImplementation(address) (node_modules/@openzeppelin/contracts-upgradeable/contracts/external/Comptroller.sol#10) is not in mixedCase
Function Untrroller._setPendingAdmin(address) (node_modules/@openzeppelin/contracts-upgradeable/contracts/external/Comptroller.sol#11) is not in mixedCase
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression *this (node_modules/@openzeppelin/contracts-upgradeable/upgradeable/ContextUpgradeable.sol#28) in ContextUpgradeable (node_modules/@openzeppelin/contracts-upgradeable/upgradeable/ContextUpgradeable.sol#16)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#redundant-statements

Variable FusePoolDirectory.getPublicPools().enforceWhitelist_scope_1 (node_modules/@openzeppelin/contracts-upgradeable/contracts/FusePoolDirectory.sol#242) is too similar to FusePoolDirectory.getPublicPoolsVerification().enforceWhitelist_scope_2 (node_modules/@openzeppelin/contracts-upgradeable/contracts/FusePoolDirectory.sol#242)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#variable-names-are-too-similar

OwnableUpgradeable._gas (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#74) is never used in FusePoolDirectory (node_modules/@openzeppelin/contracts-upgradeable/contracts/FusePoolDirectory.sol#24-34)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#unused-state-variable

renounceOwnership() should be declared external:
- OwnableUpgradeable.renounceOwnership() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#60-63)
transferOwnership(address) should be declared external:
- OwnableUpgradeable.transferOwnership(address) (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#69-73)
initialize(bool,address[]) should be declared external:
- FusePoolDirectory.initialize(bool,address[]) (node_modules/@openzeppelin/contracts-upgradeable/contracts/FusePoolDirectory.sol#88-96)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#public-function-that-could-be-declared-external

```

MarketAdminDeployer.sol

```

MarketAdmin.doTransferOut(CToken,uint256,address) (contracts/MarketAdmin.sol#132-145) sends eth to arbitrary user
Dangerous call:
- (sent) = to.call{value: amt}() (contracts/MarketAdmin.sol#139)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/upgradeable/Address.sol#201-221) uses assembly
- INLINE ASM (node_modules/@openzeppelin/contracts/upgradeable/Address.sol#212-224)
MarketAdminDeployer.deploy(address,address) (contracts/MarketAdminDeployer.sol#18-52) uses assembly
- INLINE ASM (contracts/MarketAdminDeployer.sol#37-47)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
- Version used: "0.8.4", "0.8.0", "0.8.2", "0.8.4"
- "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/ERC20.sol#4)
- "0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
- "0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4)
- "0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/upgradeable/SafeERC20.sol#4)
- "0.8.1 (node_modules/@openzeppelin/contracts/upgradeable/Context.sol#4)
- "0.8.0 (node_modules/@openzeppelin/contracts/upgradeable/Context.sol#4)
- "0.8.4 (contracts/MarketAdmin.sol#42)
- "0.8.4 (contracts/MarketAdminDeployer.sol#42)
- "0.8.4 (contracts/external/latest/Interfaces.sol#2)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/upgradeable/Address.sol#85-87) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/upgradeable/Address.sol#114-121) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/upgradeable/Address.sol#174-176) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/upgradeable/Address.sol#184-193) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/upgradeable/Address.sol#247-249) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/upgradeable/Address.sol#257-267) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/upgradeable/Address.sol#66-68) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/upgradeable/Context.sol#21-23) is never used and should be removed
ERC20._burn(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#288-295) is never used and should be removed
ERC20._mint(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#257-267) is never used and should be removed
SafeERC20.safeApprove(ERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/upgradeable/SafeERC20.sol#242-250) is never used and should be removed
SafeERC20.safeDecreaseAllowance(ERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/upgradeable/SafeERC20.sol#469-480) is never used and should be removed
SafeERC20.safeIncreaseAllowance(ERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/upgradeable/SafeERC20.sol#480-479) is never used and should be removed
SafeERC20.safeTransferFrom(ERC20,address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/upgradeable/SafeERC20.sol#479-480) is never used and should be removed
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#dead-code

Pragma version=0.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4) allows old versions
Pragma version=0.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4) allows old versions
Pragma version=0.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4) allows old versions
Pragma version=0.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
Pragma version=0.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4) allows old versions
Pragma version=0.0 (node_modules/@openzeppelin/contracts/token/ERC20/upgradeable/SafeERC20.sol#4) allows old versions
Pragma version=0.1 (node_modules/@openzeppelin/contracts/upgradeable/Context.sol#4) allows old versions
Pragma version=0.0 (node_modules/@openzeppelin/contracts/upgradeable/Context.sol#4) allows old versions
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/upgradeable/Address.sol#66-68):
- (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/upgradeable/Address.sol#63)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/upgradeable/Address.sol#128-139):
- (success, returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/upgradeable/Address.sol#137)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/upgradeable/Address.sol#157-166):
- (success, returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/upgradeable/Address.sol#164)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/upgradeable/Address.sol#184-193):
- (success, returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/upgradeable/Address.sol#191)
Low level call in MarketAdmin.doTransferOut(CToken,uint256,address) (contracts/MarketAdmin.sol#132-145):
- (sent) = to.call{value: amt}() (contracts/MarketAdmin.sol#139)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#low-level-calls

Constant MarketAdmin.isMarketAdmin (contracts/MarketAdmin.sol#19) is not in UPPER_CASE_WITH_UNDERSCORES
Function Comptroller._setWhitelistEnforcement(bool) (node_modules/@openzeppelin/contracts/external/latest/Interfaces.sol#54) is not in mixedCase
Function Comptroller._become(address) (node_modules/@openzeppelin/contracts/external/latest/Interfaces.sol#61) is not in mixedCase
Function Comptroller._setPriceOracle(address) (node_modules/@openzeppelin/contracts/external/latest/Interfaces.sol#63) is not in mixedCase
Function Comptroller._setLossFactor(address,uint256) (node_modules/@openzeppelin/contracts/external/latest/Interfaces.sol#69-72) is not in mixedCase
Function Comptroller._acceptAdmin() (node_modules/@openzeppelin/contracts/external/latest/Interfaces.sol#83) is not in mixedCase

```

```

Function Controller_setCloseFactor(uint256) (contracts/external/latest/Interfaces.sol#86-87) is not in mixedCase
Function Controller_setLiquidityIncentive(uint256) (contracts/external/latest/Interfaces.sol#89-91) is not in mixedCase
Function Controller_supportMarket(address) (contracts/external/latest/Interfaces.sol#93) is not in mixedCase
Function Controller_setMarketBorrowCaps(address,uint256[]) (contracts/external/latest/Interfaces.sol#95-98) is not in mixedCase
Function Controller_setMarketSupplyCaps(address,uint256[]) (contracts/external/latest/Interfaces.sol#100-103) is not in mixedCase
Function Controller_setReservePaused(bool) (contracts/external/latest/Interfaces.sol#105) is not in mixedCase
Function Controller_setBorrowPaused(address,bool) (contracts/external/latest/Interfaces.sol#107-109) is not in mixedCase
Function Controller_setTransferPaused(bool) (contracts/external/latest/Interfaces.sol#111) is not in mixedCase
Function Controller_setRewardsDistributor(address) (contracts/external/latest/Interfaces.sol#113) is not in mixedCase
Function Controller_addRewardsDistributor(address) (contracts/external/latest/Interfaces.sol#115-117) is not in mixedCase
Function Controller_deployMarket(bool,bytes,uint256) (contracts/external/latest/Interfaces.sol#119-123) is not in mixedCase
Function Controller_unsupportMarket(CToken) (contracts/external/latest/Interfaces.sol#125) is not in mixedCase
Function Controller_setPauseGuardian(address) (contracts/external/latest/Interfaces.sol#127-129) is not in mixedCase
Function Controller_setBorrowCapGuardian(address) (contracts/external/latest/Interfaces.sol#131) is not in mixedCase
Function CToken_supportMarketAndSetCollateralFactor(CToken,uint256) (contracts/external/latest/Interfaces.sol#184-187) is not in mixedCase
Function CToken_setAdminFee(uint256) (contracts/external/latest/Interfaces.sol#189-191) is not in mixedCase
Function CToken_setReserveFactor(uint256) (contracts/external/latest/Interfaces.sol#193-195) is not in mixedCase
Function CToken_setInterestRateModel(address) (contracts/external/latest/Interfaces.sol#197-199) is not in mixedCase
Function CToken_setNameAndSymbol(string,string) (contracts/external/latest/Interfaces.sol#201-202) is not in mixedCase
Function CToken_withdrawAdminFees(uint256) (contracts/external/latest/Interfaces.sol#204-206) is not in mixedCase
Function CToken_reduceReserves(uint256) (contracts/external/latest/Interfaces.sol#208) is not in mixedCase
Function UniwapV2Router01.WETH() (contracts/external/latest/Interfaces.sol#314) is not in mixedCase
Function UniwapV2Pair.DOMAIN_SEPARATOR() (contracts/external/latest/Interfaces.sol#500) is not in mixedCase
Function UniwapV2Pair.PERMIT_TYPEHASH() (contracts/external/latest/Interfaces.sol#552) is not in mixedCase
Function UniwapV2Pair.MINIMUM_LIQUIDITY() (contracts/external/latest/Interfaces.sol#583) is not in mixedCase
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

```

```

Variable UniwapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,address,uint256).amountDesired (contracts/external/latest/Interfaces.sol#319) is too similar to UniwapV2Router01.addLiquidity
as,uint256).amountDesired (contracts/external/latest/Interfaces.sol#320)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

```

```

MarketAdminDeployer.deploy(address,address) (contracts/MarketAdminDeployer.sol#16-52) uses literals with too many digits:
creationCode = abi.encodeWithSignature((MarketAdmin).creationCode,abi.encode(controller,manager)) (contracts/MarketAdminDeployer.sol#30-33)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#too-many-digits

```

```

renounceOwnership() should be declared external:
- Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#64-66)
transferOwnership(address) should be declared external:
- Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#62-65)
name() should be declared external:
- ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#62-64)
symbol() should be declared external:
- ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)
decimals() should be declared external:
- ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#67-69)
totalSupply() should be declared external:
- ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#94-96)
balanceOf(address) should be declared external:
- ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#101-103)
transfer(address,uint256) should be declared external:
- ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#113-117)
approve(address,uint256) should be declared external:
- ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#136-140)
transferFrom(address,address,uint256) should be declared external:
- ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#158-167)
increaseAllowance(address,uint256) should be declared external:
- ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#181-185)
decreaseAllowance(address,uint256) should be declared external:
- ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#201-210)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

```

MasterPriceOracleV2.sol

```

Different versions of Solidity is used:
- Version used: ["0.8.4", "0.8.0", "0.8.4"]
- 0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#6)
- 0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4)
- 0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4)
- 0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/ERC20Metadata.sol#4)
- 0.8.0 (node_modules/@openzeppelin/contracts/contract/Context.sol#4)
- 0.8.4 (contracts/external/latest/Interfaces.sol#2)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

```

```

Contract supports() (node_modules/@openzeppelin/contracts/utils/Context.sol#291) is never used and should be removed
ERC20_burn(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#280-290) is never used and should be removed
ERC20_mint(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#257-267) is never used and should be removed
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#dead-code

```

```

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4) allows old versions
Pragma version^0.8 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/ERC20Metadata.sol#4) allows old versions
Pragma version^0.8 (node_modules/@openzeppelin/contracts/contract/Context.sol#4) allows old versions
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#contract-versions-of-solidity

```

```

Function Controller_setWhitelistForcement(bool) (contracts/external/latest/Interfaces.sol#56) is not in mixedCase
Function Controller_setWhitelistStatuses(address[],bool[]) (contracts/external/latest/Interfaces.sol#64-69) is not in mixedCase
Function Controller_become(address) (contracts/external/latest/Interfaces.sol#61) is not in mixedCase
Function Controller_setCollateralFactor(address) (contracts/external/latest/Interfaces.sol#69-72) is not in mixedCase
Function Controller_setCollateralFactor(address,uint256) (contracts/external/latest/Interfaces.sol#74-77) is not in mixedCase
Function Controller_setLiquidityIncentive(uint256) (contracts/external/latest/Interfaces.sol#89-91) is not in mixedCase
Function Controller_supportMarket(address) (contracts/external/latest/Interfaces.sol#93) is not in mixedCase
Function Controller_setMarketBorrowCaps(address,uint256[]) (contracts/external/latest/Interfaces.sol#95-98) is not in mixedCase
Function Controller_setMarketSupplyCaps(address,uint256[]) (contracts/external/latest/Interfaces.sol#100-103) is not in mixedCase
Function Controller_setMinPaused(address,bool) (contracts/external/latest/Interfaces.sol#105) is not in mixedCase
Function Controller_setBorrowPaused(address,bool) (contracts/external/latest/Interfaces.sol#107-109) is not in mixedCase
Function Controller_setTransferPaused(bool) (contracts/external/latest/Interfaces.sol#111) is not in mixedCase
Function Controller_setRewardsDistributor(address) (contracts/external/latest/Interfaces.sol#113) is not in mixedCase
Function Controller_addRewardsDistributor(address) (contracts/external/latest/Interfaces.sol#115-117) is not in mixedCase
Function Controller_deployMarket(bool,bytes,uint256) (contracts/external/latest/Interfaces.sol#119-123) is not in mixedCase
Function Controller_unsupportMarket(CToken) (contracts/external/latest/Interfaces.sol#125) is not in mixedCase
Function Controller_setPauseGuardian(address) (contracts/external/latest/Interfaces.sol#127-129) is not in mixedCase
Function Controller_setBorrowCapGuardian(address) (contracts/external/latest/Interfaces.sol#131) is not in mixedCase
Function CToken_supportMarketAndSetCollateralFactor(CToken,uint256) (contracts/external/latest/Interfaces.sol#184-187) is not in mixedCase
Function CToken_setAdminFee(uint256) (contracts/external/latest/Interfaces.sol#189-191) is not in mixedCase
Function CToken_setReserveFactor(uint256) (contracts/external/latest/Interfaces.sol#193-195) is not in mixedCase
Function CToken_setInterestRateModel(address) (contracts/external/latest/Interfaces.sol#197-199) is not in mixedCase
Function CToken_setNameAndSymbol(string,string) (contracts/external/latest/Interfaces.sol#201-202) is not in mixedCase
Function CToken_withdrawAdminFees(uint256) (contracts/external/latest/Interfaces.sol#204-206) is not in mixedCase
Function CToken_reduceReserves(uint256) (contracts/external/latest/Interfaces.sol#208) is not in mixedCase
Function UniwapV2Router01.WETH() (contracts/external/latest/Interfaces.sol#314) is not in mixedCase
Function UniwapV2Pair.DOMAIN_SEPARATOR() (contracts/external/latest/Interfaces.sol#500) is not in mixedCase
Function UniwapV2Pair.PERMIT_TYPEHASH() (contracts/external/latest/Interfaces.sol#552) is not in mixedCase
Function UniwapV2Pair.MINIMUM_LIQUIDITY() (contracts/external/latest/Interfaces.sol#583) is not in mixedCase
Parameter MasterPriceOracleV2.setOraclesForUnderlyings(address[],BasePriceOracle[]) (contracts/oracles/MasterPriceOracleV2.sol#28) is not in mixedCase
Parameter MasterPriceOracleV2.setOraclesForUnderlyings(address[],BasePriceOracle[]) (contracts/oracles/MasterPriceOracleV2.sol#38) is not in mixedCase
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

```

```

Variable UniwapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,address,uint256).amountDesired (contracts/external/latest/Interfaces.sol#319) is too similar to UniwapV2Router01.addLiquidity
as,uint256).amountDesired (contracts/external/latest/Interfaces.sol#320)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

```

```

renounceOwnership() should be declared external:
- Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#64-66)
transferOwnership(address) should be declared external:
- Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#62-65)
name() should be declared external:
- ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#62-64)
symbol() should be declared external:
- ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)
decimals() should be declared external:
- ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#67-69)
totalSupply() should be declared external:
- ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#94-96)
balanceOf(address) should be declared external:
- ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#101-103)
transfer(address,uint256) should be declared external:
- ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#113-117)
approve(address,uint256) should be declared external:
- ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#136-140)
transferFrom(address,address,uint256) should be declared external:
- ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#158-167)
increaseAllowance(address,uint256) should be declared external:
- ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#181-185)
decreaseAllowance(address,uint256) should be declared external:
- ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#201-210)
setOraclesForUnderlyings(address[],BasePriceOracle[]) should be declared external:
- MasterPriceOracleV2.setOraclesForUnderlyings(address[],BasePriceOracle[]) (contracts/oracles/MasterPriceOracleV2.sol#36-49)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

```

PreferredPriceOracle.sol

```

ERC20Upgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#512) shadows:
  ContextUpgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
Reference: https://github.com/crytic/solther/wiki/Detector-Documentation#state-variable-shadowing

ChainlinkPriceOracleV2.constructor(address,bool)_admin (contracts/oracles/ChainlinkPriceOracleV2.sol#67) lacks a zero-check on :
  admin = _admin (contracts/oracles/ChainlinkPriceOracleV2.sol#68)
ChainlinkPriceOracleV2.changeAdmin(address)_newAdmin (contracts/oracles/ChainlinkPriceOracleV2.sol#75) lacks a zero-check on :
  _admin = newAdmin (contracts/oracles/ChainlinkPriceOracleV2.sol#77)
Reference: https://github.com/crytic/solther/wiki/Detector-Documentation#missing-zero-address-validation

AddressUpgradeable.isContract(address) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#26-35) uses assembly
  !ILINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#33)
AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#147-164) uses assembly
  !ILINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#156-159)
Reference: https://github.com/crytic/solther/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
  - Version used: ["0.6.12", ">=0.4.24<0.8.0", ">=0.6.0<0.8.0", ">=0.6.24<0.8.0"]
  - >=0.4.24<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#3)
  - >=0.4.24<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/Initializable.sol#4)
  - >=0.4.8<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#3)
  - >=0.4.8<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#3)
  - >=0.4.24<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#3)
  - >=0.4.8<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#3)
  - 0.6.12 (contracts/external/compound/CompoundPriceOracle.sol#2)
  - 0.6.12 (contracts/external/compound/ERC20.sol#2)
  - 0.6.12 (contracts/external/compound/GToken.sol#2)
  - 0.6.12 (contracts/external/compound/GToken.sol#2)
  - 0.6.12 (contracts/oracles/BasePriceOracle.sol#2)
  - 0.6.12 (contracts/oracles/ChainlinkPriceOracleV2.sol#2)
  - 0.6.12 (contracts/oracles/PreferredPriceOracle.sol#2)
Reference: https://github.com/crytic/solther/wiki/Detector-Documentation#different-pragma-directives-are-used

AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#147-164) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#79-81) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#87-91) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#184-186) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#114-121) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#117-121) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#139-145) is never used and should be removed
AddressUpgradeable.isContract(address) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#26-35) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#53-59) is never used and should be removed
ContextUpgradeable._context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#17-20) is never used and should be removed
ContextUpgradeable._context_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is never used and should be removed
ContextUpgradeable._msgData() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#27-30) is never used and should be removed
ERC20Upgradeable._ERC20_init(string,string) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#65-69) is never used and should be removed
ERC20Upgradeable._ERC20_init_unchained(string,string) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#61-65) is never used and should be removed
ERC20Upgradeable.burn(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#255-268) is never used and should be removed
ERC20Upgradeable._mint(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#224-242) is never used and should be removed
ERC20Upgradeable._setupDecimals(uint8) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#293-295) is never used and should be removed
Intillizable._isConstructor() (node_modules/@openzeppelin/contracts-upgradeable/proxy/Intillizable.sol#52-54) is never used and should be removed
SafeMathUpgradeable.div(uint256,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#191-193) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#152-155) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#210-213) is never used and should be removed
SafeMathUpgradeable.sub(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#181-184) is never used and should be removed
SafeMathUpgradeable.tryAdd(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#24-28) is never used and should be removed
SafeMathUpgradeable.tryDiv(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#60-63) is never used and should be removed
SafeMathUpgradeable.tryMul(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#76-79) is never used and should be removed
SafeMathUpgradeable.trySub(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#53-57) is never used and should be removed
SafeMathUpgradeable.trySub(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#95-98) is never used and should be removed
Reference: https://github.com/crytic/solther/wiki/Detector-Documentation#dead-code

Pragma version=>0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#3) is too complex
Pragma version=>0.4.24<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/Initializable.sol#4) is too complex
Pragma version=>0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#3) is too complex
Pragma version=>0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#3) is too complex
Pragma version=>0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#3) is too complex
Pragma version=>0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#3) is too complex
Reference: https://github.com/crytic/solther/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#53-59):
  - (success) recipient.call{value: amount}() (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#57)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#114-121):
  - (success,returnData) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#139-145):
  - (success,returnData) = target.staticCall(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#143)
Reference: https://github.com/crytic/solther/wiki/Detector-Documentation#low-level-calls

Function ERC20Upgradeable._ERC20_init(string,string) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#65-69) is not in mixedCase
Function ERC20Upgradeable._ERC20_init_unchained(string,string) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#61-65) is not in mixedCase
Variable AddressUpgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#31) is not in mixedCase
Function ContextUpgradeable._context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#17-20) is not in mixedCase
Function ContextUpgradeable._context_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is not in mixedCase
Variable ContextUpgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31) is not in mixedCase
Reference: https://github.com/crytic/solther/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#28)" in ContextUpgradeable (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#16)
Reference: https://github.com/crytic/solther/wiki/Detector-Documentation#redundant-statements

ERC20Upgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#31) is never used in ERC20Upgradeable (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#31)
Reference: https://github.com/crytic/solther/wiki/Detector-Documentation#unused-state-variable

name() should be declared external:
  - ERC20Upgradeable.name() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#70-72)
symbol() should be declared external:
  - ERC20Upgradeable.symbol() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#78-80)
decimals() should be declared external:
  - ERC20Upgradeable.decimals() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#95-97)
totalSupply() should be declared external:
  - ERC20Upgradeable.totalSupply() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#102-104)
balanceOf(address) should be declared external:
  - ERC20Upgradeable.balanceOf(address) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#109-111)
transfer(address,uint256) should be declared external:
  - ERC20Upgradeable.transfer(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#121-124)
allowance(address,address) should be declared external:
  - ERC20Upgradeable.allowance(address,address) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#129-131)
approve(address,uint256) should be declared external:
  - ERC20Upgradeable.approve(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#140-143)
transferFrom(address,address,uint256) should be declared external:
  - ERC20Upgradeable.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#158-162)
increaseAllowance(address,uint256) should be declared external:
  - ERC20Upgradeable.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#176-179)
decreaseAllowance(address,uint256) should be declared external:
  - ERC20Upgradeable.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#195-198)
Reference: https://github.com/crytic/solther/wiki/Detector-Documentation#public-function-that-could-be-declared-external

```



```

Function UniswapV2Pair_DOMAIN_SEPARATOR() (contracts/external/uniswap/UniswapV2Pair.sol#19) is not in mixedCase
Function UniswapV2Pair_PERMIT_TYPEHASH() (contracts/external/uniswap/UniswapV2Pair.sol#20) is not in mixedCase
Function UniswapV2Pair_MINIMUM_LIQUIDITY() (contracts/external/uniswap/UniswapV2Pair.sol#23) is not in mixedCase
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression `this (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#28)` inContextUpgradeable (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#16)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#redundant-statements

Variable UniswapTwapPriceOracleV2Root_price@Cumulative (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#43) is too similar to UniswapTwapPriceOracleV2Root_prices@Twap(address),currPxDum (contract
Variable UniswapTwapPriceOracleV2Root_update(address),priceCumulative (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#291) is too similar to UniswapTwapPriceOracleV2Root_update(address),priceCumulative (co
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

ERC20Upgradeable__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#312) is never used in ERC20Upgradeable (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20U
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#unused-state-variable

name() should be declared external:
- ERC20Upgradeable.name() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#78-72)
symbol() should be declared external:
- ERC20Upgradeable.symbol() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#78-80)
decimals() should be declared external:
- ERC20Upgradeable.decimals() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#95-97)
totalSupply() should be declared external:
- ERC20Upgradeable.totalSupply() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#102-104)
balanceOf(address) should be declared external:
- ERC20Upgradeable.balanceOf(address) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#109-111)
transfer(address,uint256) should be declared external:
- ERC20Upgradeable.transfer(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#121-124)
allowance(address,address) should be declared external:
- ERC20Upgradeable.allowance(address,address) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#129-131)
approve(address,uint256) should be declared external:
- ERC20Upgradeable.approve(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#148-143)
transferFrom(address,address,uint256) should be declared external:
- ERC20Upgradeable.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#158-162)
increaseAllowance(address,uint256) should be declared external:
- ERC20Upgradeable.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#176-179)
decreaseAllowance(address,uint256) should be declared external:
- ERC20Upgradeable.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#195-198)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

```

UniswapTwapPriceOracleV2Root.sol

```

ERC20Upgradeable__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#312) shadow:
- ContextUpgradeable__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#state-variable-shadowing

UniswapTwapPriceOracleV2Root_price(address,address) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#117-133) performs a multiplication on the result of a division:
- (price@Twap(pair).div(2 ** 56).mul(baseUnit).div(2 ** 56)) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#128-132)
UniswapTwapPriceOracleV2Root_price(address,address,address) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#117-133) performs a multiplication on the result of a division:
- (price@Twap(pair).div(2 ** 56).mul(baseUnit).div(2 ** 56)) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#128-132)
UniswapTwapPriceOracleV2Root_deviation(address,address) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#227-255) performs a multiplication on the result of a division:
- twapPrice = (price@Twap(pair).div(2 ** 56).mul(baseUnit).div(2 ** 56)) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#241-248)
UniswapTwapPriceOracleV2Root_deviation(address,address) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#227-255) performs a multiplication on the result of a division:
- ratio = spotPrice.mul(1e18).div(twapPrice) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#241-248)
- spotPrice = reserve0.mul(baseUnit).div(reserve1) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#248-250)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#divide-before-multiply

UniswapTwapPriceOracleV2Root_pairsFor(address[],address) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#160-173) has external calls inside a loop: pairs[i] = UniswapV2Factory(factory).getPair(token/
V2Root.sol#171)
UniswapTwapPriceOracleV2Root_workable(address,address,uint256,uint256) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#201-224) has external calls inside a loop: (lastTime = UniswapV2Pair(pair).getReserves()
UniswapTwapPriceOracleV2Root_deviation(address,address) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#227-255) has external calls inside a loop: token0 = UniswapV2Pair(pair).token0() (contracts/oracles/Unit
UniswapTwapPriceOracleV2Root_deviation(address,address) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#227-255) has external calls inside a loop: baseUnit = 10 ** uint256(ERC20Upgradeable(underlying).decimals
256)
UniswapTwapPriceOracleV2Root_currentPx@Cumulative(address) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#177-191) has external calls inside a loop: px@Cumulative = UniswapV2Pair(pair).priceCumulativeLast() (contracts/ori
UniswapTwapPriceOracleV2Root_currentPx@Cumulative(address) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#177-191) has external calls inside a loop: (reserve0.reserve1,lastTime) = UniswapV2Pair(pair).getReserves() (
UniswapTwapPriceOracleV2Root_deviation(address,address) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#227-255) has external calls inside a loop: (reserve0.reserve1) = UniswapV2Pair(pair).getReserves() (cont
UniswapTwapPriceOracleV2Root_currentPx@Cumulative(address) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#177-191) has external calls inside a loop: px@Cumulative = UniswapV2Pair(pair).priceCumulativeLast() (contracts/or
UniswapTwapPriceOracleV2Root_currentPx@Cumulative(address) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#177-191) has external calls inside a loop: (reserve0.reserve1,lastTime) = UniswapV2Pair(pair).getReserves() (
UniswapTwapPriceOracleV2Root_deviation(address,address) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#227-255) has external calls inside a loop: underlying = UniswapV2Pair(pair).token1() (contracts/oracles/
UniswapTwapPriceOracleV2Root_update(address) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#286-301) has external calls inside a loop: priceCumulative = UniswapV2Pair(pair).priceCumulativeLast() (contracts
UniswapTwapPriceOracleV2Root_update(address) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#286-301) has external calls inside a loop: (lastTime = UniswapV2Pair(pair).getReserves() (contracts/oracles/Unisw
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#calls-inside-a-loop

UniswapTwapPriceOracleV2Root_price@Twap(address) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#27-45) uses timestamp for comparisons
Dangerous comparisons:
- lastObservation.timestamp > now - MIN_Twap_TIME (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#33)
- require(bool,string)(elapsedTime >= MIN_Twap_TIME.Bad Twap time.) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#40)
UniswapTwapPriceOracleV2Root_prices@Twap(address) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#62-78) uses timestamp for comparisons
Dangerous comparisons:
- lastObservation.timestamp > now - MIN_Twap_TIME (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#58)
- require(bool,string)(elapsedTime >= MIN_Twap_TIME.Bad Twap time.) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#65)
UniswapTwapPriceOracleV2Root_currentPx@Cumulative(address) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#177-191) uses timestamp for comparisons
Dangerous comparisons:
- lastTime != now (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#187)
UniswapTwapPriceOracleV2Root_currentPx@Cumulative(address) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#177-191) uses timestamp for comparisons
Dangerous comparisons:
- lastTime != currTime (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#188)
UniswapTwapPriceOracleV2Root_workable(address,address,uint256,uint256) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#201-224) uses timestamp for comparisons
Dangerous comparisons:
- (block.timestamp - observations[pair][observationCount[pair] - 1] % OBSERVATION_BUFFER).timestamp > (minPeriod) & lastTime != observations[pair][observationCount[pair] - 1] % OBSERVATION_BUFFER.time
(contract/oracles/UniswapTwapPriceOracleV2Root.sol#213-223)
- (block.timestamp - observations[pair][observationCount[pair] - 1] % OBSERVATION_BUFFER).timestamp > (MIN_Twap_TIME) & lastTime != observations[pair][observationCount[pair] - 1] % OBSERVATION_BUFFER.time
hold (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#213-223)
UniswapTwapPriceOracleV2Root_deviation(address,address) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#227-255) uses timestamp for comparisons
Dangerous comparisons:
- ratio >= 1e18 (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#254)
UniswapTwapPriceOracleV2Root_update(address) (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#258-278) uses timestamp for comparisons
Dangerous comparisons:
- observationCount[pair] <= 0 || (block.timestamp - observations[pair][observationCount[pair] - 1] % OBSERVATION_BUFFER).timestamp > MIN_Twap_TIME (contracts/oracles/UniswapTwapPriceOracleV2Root.sol#263-
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#block-timestamp

AddressUpgradeable.isContract(address) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#26-35) uses assembly
- INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#33)
AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#147-164) uses assembly
- INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#156-159)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:

```

```

- Version used: ["0.6.12", ">=0.4.24<0.8.0", ">=0.6.0<0.8.0", ">=0.6.2<0.8.0"]
-> 0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#3)
-> 0.6.2<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/Initializable.sol#6)
-> 0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#3)
-> 0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#3)
-> 0.6.2<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#3)
-> 0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#3)
- 0.6.12 (contracts/external/Uniswap/UniswapV2Factory.sol#2)
- 0.6.12 (contracts/external/Uniswap/UniswapV2Pair.sol#2)
- 0.6.12 (contracts/oracles/UniswapV2PriceOracleV2Root.sol#2)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#different-pragma-directives-are-used

AddressUpgradeable._verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#147-164) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#79-81) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#89-91) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#104-108) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,string,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#111-121) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#129-133) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#139-145) is never used and should be removed
AddressUpgradeable.isContract(address) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#26-35) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#55-59) is never used and should be removed
ContextUpgradeable._Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#17-19) is never used and should be removed
ContextUpgradeable._Context_init_unchecked() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is never used and should be removed
ContextUpgradeable._msgData() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#27-30) is never used and should be removed
ERC20Upgradeable._ERC20_init(string,string) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#56-59) is never used and should be removed
ERC20Upgradeable._ERC20_init_unchecked(string,string) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#61-65) is never used and should be removed
ERC20Upgradeable._burn(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#235-242) is never used and should be removed
ERC20Upgradeable._mint(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#234-242) is never used and should be removed
ERC20Upgradeable._setupDecimals(uint8) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#293-295) is never used and should be removed
Initializable._isContract() (node_modules/@openzeppelin/contracts-upgradeable/proxy/Initializable.sol#52-54) is never used and should be removed
SafeMathUpgradeable._div(uint256,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#190-193) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#152-155) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#218-223) is never used and should be removed
SafeMathUpgradeable.sub(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#181-184) is never used and should be removed
SafeMathUpgradeable.tryAdd(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#24-28) is never used and should be removed
SafeMathUpgradeable.tryDiv(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#48-53) is never used and should be removed
SafeMathUpgradeable.tryMul(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#78-73) is never used and should be removed
SafeMathUpgradeable.tryMod(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#95-103) is never used and should be removed
SafeMathUpgradeable.trySub(uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#145-151) is never used and should be removed
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#dead-code

Pragma version=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#3) is too complex
Pragma version=0.4.24<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/Initializable.sol#6) is too complex
Pragma version=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#3) is too complex
Pragma version=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#3) is too complex
Pragma version=0.6.2<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#3) is too complex
Pragma version=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#3) is too complex
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#contract-version-9-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#59-59):
- (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#67)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#114-121):
- (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#119)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#139-145):
- (success,returndata) = target.staticCall(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#143)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#low-level-calls

Function ERC20Upgradeable._ERC20_init(string,string) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#56-59) is not in mixedCase
Function ERC20Upgradeable._ERC20_init_unchecked(string,string) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#61-65) is not in mixedCase
Variable ERC20Upgradeable._gas (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#12) is not in mixedCase
Function ContextUpgradeable._Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#17-19) is not in mixedCase
Function ContextUpgradeable._Context_init_unchecked() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is not in mixedCase
Variable ContextUpgradeable._gas (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#13) is not in mixedCase
Function UniswapV2Pair.DOMAIN_SEPARATOR() (contracts/external/Uniswap/UniswapV2Pair.sol#19) is not in mixedCase
Function UniswapV2Pair.PREFIX_TYPERHASH() (contracts/external/Uniswap/UniswapV2Pair.sol#20) is not in mixedCase
Function UniswapV2Pair.MINIMUM_LIQUIDITY() (contracts/external/Uniswap/UniswapV2Pair.sol#37) is not in mixedCase
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression `this (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#28)` in ContextUpgradeable (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#16-18)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#redundant-statements

Variable UniswapV2PriceOracleV2Root.priceTWAP(address).currPxIDum (contracts/oracles/UniswapV2PriceOracleV2Root.sol#41) is too similar to UniswapV2PriceOracleV2Root.priceTWAP(address).currPxIDum (contract
Variable UniswapV2PriceOracleV2Root.update(address).priceCumulative (contracts/oracles/UniswapV2PriceOracleV2Root.sol#29) is too similar to UniswapV2PriceOracleV2Root.update(address).priceCumulative (co
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#variable-names-are-too-similar

ERC20Upgradeable._gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#12) is never used in ERC20Upgradeable (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#unused-state-variable

name() should be declared external:
- ERC20Upgradeable.name() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#78-72)
symbol() should be declared external:
- ERC20Upgradeable.symbol() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#78-80)
decimals() should be declared external:
- ERC20Upgradeable.decimals() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#95-97)
totalSupply() should be declared external:
- ERC20Upgradeable.totalSupply() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#102-104)
balanceOf(address) should be declared external:
- ERC20Upgradeable.balanceOf(address) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#109-111)
transfer(address,uint256) should be declared external:
- ERC20Upgradeable.transfer(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#121-124)
allowance(address,address) should be declared external:
- ERC20Upgradeable.allowance(address,address) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#129-131)
approve(address,uint256) should be declared external:
- ERC20Upgradeable.approve(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#148-143)
transferFrom(address,address,uint256) should be declared external:
- ERC20Upgradeable.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#158-162)
increaseAllowance(address,uint256) should be declared external:
- ERC20Upgradeable.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#176-179)
decreaseAllowance(address,uint256) should be declared external:
- ERC20Upgradeable.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#195-198)
Reference: https://github.com/crytic/sliether/wiki/Detector-Documentation#public-function-that-could-be-declared-external

```


UnwrapAssetOracle.sol

```

Context_msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
ERC20_burn(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#288-295) is never used and should be removed
ERC20_mint(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#267-267) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/ERC20Metadata.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function Comptroller_setWhitelistEnforcement(bool) (contracts/external/latest/Interfaces.sol#54) is not in mixedCase
Function Comptroller_setWhitelistStatuses(address[],bool[]) (contracts/external/latest/Interfaces.sol#56-59) is not in mixedCase
Function Comptroller_become(address) (contracts/external/latest/Interfaces.sol#62) is not in mixedCase
Function Comptroller_setPriceOracle(address) (contracts/external/latest/Interfaces.sol#63) is not in mixedCase
Function Comptroller_setCollateralFactor(address,uint256) (contracts/external/latest/Interfaces.sol#69-72) is not in mixedCase
Function Comptroller_setCloseAdmin() (contracts/external/latest/Interfaces.sol#83) is not in mixedCase
Function Comptroller_setCloseFactor(uint256) (contracts/external/latest/Interfaces.sol#85-87) is not in mixedCase
Function Comptroller_setLiquidationIncentive(uint256) (contracts/external/latest/Interfaces.sol#89-91) is not in mixedCase
Function Comptroller_supportMarket(address) (contracts/external/latest/Interfaces.sol#93) is not in mixedCase
Function Comptroller_setMarketBorrowCaps(address[],uint256[]) (contracts/external/latest/Interfaces.sol#95-98) is not in mixedCase
Function Comptroller_setMarketSupplyCaps(address[],uint256[]) (contracts/external/latest/Interfaces.sol#100-103) is not in mixedCase
Function Comptroller_setMinPaused(address,bool) (contracts/external/latest/Interfaces.sol#106) is not in mixedCase
Function Comptroller_setBorrowPaused(address,bool) (contracts/external/latest/Interfaces.sol#107-109) is not in mixedCase
Function Comptroller_setTransferPaused(bool) (contracts/external/latest/Interfaces.sol#111) is not in mixedCase
Function Comptroller_setSeizePaused(bool) (contracts/external/latest/Interfaces.sol#113) is not in mixedCase
Function Comptroller_addRewardsDistributor(address) (contracts/external/latest/Interfaces.sol#115-117) is not in mixedCase
Function Comptroller_deployMarket(bool,bytes,uint256) (contracts/external/latest/Interfaces.sol#119-123) is not in mixedCase
Function Comptroller_unsupportMarket(CToken) (contracts/external/latest/Interfaces.sol#125) is not in mixedCase
Function Comptroller_setPauseGuardian(address) (contracts/external/latest/Interfaces.sol#127-129) is not in mixedCase
Function Comptroller_setBorrowCapGuardian(address) (contracts/external/latest/Interfaces.sol#131) is not in mixedCase
Function CToken_supportMarketAndSetCollateralFactor(address,uint256) (contracts/external/latest/Interfaces.sol#134-137) is not in mixedCase
Function CToken_setAdminFee(uint256) (contracts/external/latest/Interfaces.sol#139-141) is not in mixedCase
Function CToken_setReserveFactor(uint256) (contracts/external/latest/Interfaces.sol#143-146) is not in mixedCase
Function CToken_setInterestRateModel(address) (contracts/external/latest/Interfaces.sol#147-149) is not in mixedCase
Function CToken_setNameAndSymbol(string,string) (contracts/external/latest/Interfaces.sol#151-152) is not in mixedCase
Function CToken_withdrawAdminFees(uint256) (contracts/external/latest/Interfaces.sol#154-156) is not in mixedCase
Function CToken_reduceReserve(uint256) (contracts/external/latest/Interfaces.sol#158) is not in mixedCase
Function UniswapV2Router01_WETH() (contracts/external/latest/Interfaces.sol#161) is not in mixedCase
Function UniswapV2Pair_DOMAIN_SEPARATOR() (contracts/external/latest/Interfaces.sol#163) is not in mixedCase
Function UniswapV2Pair_PERMIT_TYPEHASH() (contracts/external/latest/Interfaces.sol#165) is not in mixedCase
Function UniswapV2Pair_MINIMUM_LIQUIDITY() (contracts/external/latest/Interfaces.sol#167) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable UniswapV2Router01_addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountDesired (contracts/external/latest/Interfaces.sol#169) is too similar to UniswapV2Router01_addLiquidity
as,uint256).amountDesired (contracts/external/latest/Interfaces.sol#169)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

name() should be declared external:
- ERC20_name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#62-64)
symbol() should be declared external:
- ERC20_symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)
decimals() should be declared external:
- ERC20_decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#87-89)
totalSupply() should be declared external:
- ERC20_totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#94-96)
balanceOf(address) should be declared external:
- ERC20_balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#101-103)
transfer(address,uint256) should be declared external:
- ERC20_transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#113-117)
approve(address,uint256) should be declared external:
- ERC20_approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#136-140)
transferFrom(address,address,uint256) should be declared external:
- ERC20_transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#158-167)
increaseAllowance(address,uint256) should be declared external:
- ERC20_increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#181-185)
decreaseAllowance(address,uint256) should be declared external:
- ERC20_decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#201-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

```

- No major issues were found by Slither.

4.2 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 all the contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

MythX results:

CErc20Delegate.sol

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.

CEtherDelegate.sol

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.

ChainlinkPriceOracleV2.sol

Line	SWC Title	Severity	Short Description
20	(SWC-123) Requirement Violation	Low	Requirement violation.
124	(SWC-110) Assert Violation	Medium	An assertion violation was triggered.
190	(SWC-123) Requirement Violation	Low	Requirement violation.

ComptrollerStorage.sol

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.

CToken.sol

Line	SWC Title	Severity	Short Description
255	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.
473	(SWC-104) Unchecked Call Return Value	Medium	Unchecked return value from low-level external call.
1506	(SWC-104) Unchecked Call Return Value	Medium	Unchecked return value from low-level external call.
1589	(SWC-104) Unchecked Call Return Value	Medium	Unchecked return value from low-level external call.

CTokenInterfaces.sol

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.

FuseFeeDistributor.sol

Line	SWC Title	Severity	Short Description
15	(SWC-123) Requirement Violation	Low	Requirement violation.
65	(SWC-123) Requirement Violation	Low	Requirement violation.
108	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.
129	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.

FusePoolDirectory.sol

Line	SWC Title	Severity	Short Description
128	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.
153	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.

MarketAdmin.sol

Line	SWC Title	Severity	Short Description
85	(SWC-110) Assert Violation	Unknown	Out of bounds array access
98	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered

MarketAdminDeployer.sol

Line	SWC Title	Severity	Short Description
27	(SWC-128) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.

MasterPriceOracleV2.sol

Line	SWC Title	Severity	Short Description
44	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
45	(SWC-110) Assert Violation	Unknown	Out of bounds array access
84	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
87	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
87	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered

UniswapTwapPriceOracleV2Root.sol

Line	SWC Title	Severity	Short Description
14	(SWC-123) Requirement Violation	Low	Requirement violation.
123	(SWC-123) Requirement Violation	Low	Requirement violation.

- The floating pragma flagged by MythX is a false positive, as the pragma is set in the `hardhat.config.js` file to the `0.6.12` version.



THANK YOU FOR CHOOSING

// HALBORN

