// HALBORN

Strike Finance - Strike Protocol

Smart Contract Security Assessment

Prepared by: **Halborn**

Date of Engagement: **December 4th, 2023 - December 29th, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE
0.1	Document Creation	12/04/2023
0.2	Document Updates	12/28/2023
0.3	Draft Review	12/29/2023
0.4	Draft Review	12/29/2023
1.0	Remediation Plan	01/10/2024
1.1	Remediation Plan Review	01/10/2024
1.2	Remediation Plan Review	01/10/2024

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

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

The Strike Protocol is a Compound V2 fork that allows supplying or borrowing assets. Through the sToken contracts, accounts on the Arbitrum blockchain supply capital (Ether or ERC-20 tokens) to receive sTokens or borrow assets from the protocol.

Strike Finance engaged Halborn to conduct a security assessment on their Strike Protocol smart contracts beginning on December 4th, 2023 and ending on December 29th, 2023. The security assessment was scoped to the smart contracts provided in the following GitHub repositories:

- StrikeFinance/strike-protocol/.

1.2 ASSESSMENT SUMMARY

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

The purpose of this assessment is to:

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

In summary, Halborn identified some security risks that were partially addressed by the Strike Finance team.

1.3 TEST APPROACH & METHODOLOGY

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

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

2. RISK METHODOLOGY

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

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

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

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

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

2.1 EXPLOITABILITY

Attack Origin (AO):

Captures whether the attack requires compromising a specific account.

Attack Cost (AC):

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

Attack Complexity (AX):

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

Metrics:

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

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

2.2 IMPACT

Confidentiality (C):

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

Integrity (I):

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

Availability (A):

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

Deposit (D):

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

Yield (Y):

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

Metrics:

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

Impact I is calculated using the following formula:

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

2.3 SEVERITY COEFFICIENT

Reversibility (R):

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

Scope (S):

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

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

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

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

The score is rounded up to 1 decimal places.

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

2.4 SCOPE

1. IN-SCOPE TREE & COMMIT :

The security assessment was scoped to the following smart contracts on the audit branch:

- AggregatorV2V3Interface.sol
- CarefulMath.sol
- Comptroller.sol
- ComptrollerInterface.sol
- ComptrollerStorage.sol
- DAIInterestRateModelV3.sol
- EIP20Interface.sol
- EIP20NonStandardInterface.sol
- ErrorReporter.sol
- Exponential.sol
- InterestRateModel.sol
- JumpRateModel.sol
- JumpRateModelV2.sol
- Maximillion.sol
- PriceOracle.sol
- Reservoir.sol
- SafeMath.sol
- SDaiDelegate.sol
- SErc20.sol
- SErc20Delegate.sol
- SErc20Delegator.sol
- SErc20Immutable.sol
- SEther.sol
- SimplePriceOracle.sol
- SStrkLikeDelegate.sol
- SToken.sol
- STokenInterfaces.sol
- StrikeAggregatorPriceOracle.sol
- StrikePriceOracle.sol
- STRK.sol

- Timelock.sol
- Unitroller.sol
- WhitePaperInterestRateModel.sol
- GovernorAlpha.sol
- GovernorAlpha2.sol
- StrikeLens.sol
- Context.sol
- Math.sol
- IERC20.sol
- SafeERC20.sol
- Address.sol
- IStrikeStaking.sol
- StrikeStaking.sol
- StrikeStakingProxy.sol
- StrikeStakingStorage.sol

Initial Commit ID: fad0e8011ba9c0d7c3b8fe22102dd2989fc5ef9a

Fixed Commit ID: 13b661906dcfbade26edcb718704ef7df09736f6

3. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	6	4	7

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) EMPTY MARKETS ARE VULNERABLE TO INFLATION ATTACKS	Medium (6.3)	SOLVED - 01/10/2024
(HAL-02) POTENTIAL REENTRANCY IN BORROWFRESH FUNCTION	Medium (6.3)	SOLVED - 01/10/2024
(HAL-03) GOVERNORALPHA2 CONTRACT IS INCOMPATIBLE WITH THE CURRENT TIMELOCK	Medium (5.0)	RISK ACCEPTED
(HAL-04) SEQUENCER STATUS IS NEVER CHECKED	Medium (5.0)	RISK ACCEPTED
(HAL-05) LATESTANSWER CALL MAY RETURN STALE RESULTS	Medium (5.0)	RISK ACCEPTED
(HAL-06) EARNED REWARDS WILL BE LOST IF WITHDRAWEXPIREDLOCKS() IS CALLED BEFORE GETREWARDS()	Medium (5.0)	SOLVED - 01/10/2024
(HAL-07) STRIKESTAKINGPROXY IS SUBJECT TO STORAGE COLLISIONS	Low (3.7)	RISK ACCEPTED
(HAL-08) DIRECT USAGE OF ECRECOVER ALLOWS SIGNATURE MALLEABILITY	Low (3.1)	RISK ACCEPTED
(HAL-09) BLOCKSPERYEAR ARE NOT CORRECTLY ADJUSTED IN THE DIFFERENT RATE MODELS	Low (3.1)	RISK ACCEPTED
(HAL-10) NO LIMITATION ON THE AMOUNT OF REWARDTOKENS THAT CAN BE ADDED TO THE STRIKESTAKING CONTRACT	Low (2.5)	SOLVED - 01/10/2024
(HAL-11) SIMPLEPRICEORACLE LACKS ACCESS CONTROL	Informational (0.0)	SOLVED - 01/10/2024
(HAL-12) INCORRECT IMPLEMENTATION OF ONLYBLACKLIST AND NONBLACKLIST MODIFIERS	Informational (0.0)	ACKNOWLEDGED
(HAL-13) LOCK PARAMETER CAN BE REMOVED FROM THE STRIKESTAKING.STAKE() FUNCTION	Informational (0.0)	ACKNOWLEDGED
(HAL-14) FLOATING PRAGMA	Informational (0.0)	ACKNOWLEDGED

(HAL-15) UNOPTIMIZED LOOPS	Informational (0.0)	ACKNOWLEDGED
(HAL-16) MISSING/INCOMPLETE NATSPEC COMMENTS	Informational (0.0)	ACKNOWLEDGED
(HAL-17) WRONG COMMENTS IN THE STRIKESTAKING CONTRACT	Informational (0.0)	SOLVED - 01/10/2024

FINDINGS & TECH DETAILS

4.1 (HAL-01) EMPTY MARKETS ARE VULNERABLE TO INFLATION ATTACKS - MEDIUM (6.3)

Commit IDs affected:
- fad0e8011ba9c0d7c3b8fe22102dd2989fc5ef9a

Description:

This vulnerability has existed in the Compound v2 code since its launch, presenting itself when markets are launched with a collateral value in place but no depositors or following markets becoming empty due to user withdrawal post-launch.

This issue exploits a rounding error that is present in the redeemFresh() function:

Listing 1: SToken.sol (Line 644)

```
613 function redeemFresh(address payable redeemer, uint redeemTokensIn
 ↳ , uint redeemAmountIn) internal returns (uint) {
614     require(redeemTokensIn == 0 || redeemAmountIn == 0, "one of
 ↳ redeemTokensIn or redeemAmountIn must be zero");
615
616     RedeemLocalVars memory vars;
617
618     /* exchangeRate = invoke Exchange Rate Stored() */
619     (vars.mathErr, vars.exchangeRateMantissa) =
 ↳ exchangeRateStoredInternal();
620     if (vars.mathErr != MathError.NO_ERROR) {
621         return failOpaque(Error.MATH_ERROR, FailureInfo.
 ↳ REDEEM_EXCHANGE_RATE_READ_FAILED, uint(vars.mathErr));
622     }
623
624     /* If redeemTokensIn > 0: */
625     if (redeemTokensIn > 0) {
626         /*
627          * We calculate the exchange rate and the amount of
 ↳ underlying to be redeemed:
```

```
628             *  redeemTokens = redeemTokensIn
629             *  redeemAmount = redeemTokensIn x exchangeRateCurrent
630             */
631             vars.redeemTokens = redeemTokensIn;
632
633             (vars.mathErr, vars.redeemAmount) = mulScalarTruncate(Exp
 ↳ ({mantissa: vars.exchangeRateMantissa}), redeemTokensIn);
634             if (vars.mathErr != MathError.NO_ERROR) {
635                 return failOpaque(Error.MATH_ERROR, FailureInfo.
 ↳ REDEEM_EXCHANGE_TOKENS_CALCULATION_FAILED, uint(vars.mathErr));
636             }
637         } else {
638             /*
639             * We get the current exchange rate and calculate the
 ↳ amount to be redeemed:
640             *  redeemTokens = redeemAmountIn / exchangeRate
641             *  redeemAmount = redeemAmountIn
642             */
643
644             (vars.mathErr, vars.redeemTokens) = divScalarByExpTruncate
 ↳ (redeemAmountIn, Exp({mantissa: vars.exchangeRateMantissa}));
645             if (vars.mathErr != MathError.NO_ERROR) {
646                 return failOpaque(Error.MATH_ERROR, FailureInfo.
 ↳ REDEEM_EXCHANGE_AMOUNT_CALCULATION_FAILED, uint(vars.mathErr));
647             }
648
649             vars.redeemAmount = redeemAmountIn;
650         }
651
652         /* Fail if redeem not allowed */
653         uint allowed = comptroller.redeemAllowed(address(this),
 ↳ redeemer, vars.redeemTokens);
654         if (allowed != 0) {
655             return failOpaque(Error.COMPTROLLER_REJECTION, FailureInfo
 ↳ .REDEEM_COMPTROLLER_REJECTION, allowed);
656         }
657
658         /* Verify market's block number equals current block number */
659         if (accrualBlockNumber != getBlockNumber()) {
660             return fail(Error.MARKET_NOT_FRESH, FailureInfo.
 ↳ REDEEM_FRESHNESS_CHECK);
661         }
662
663         /*
```

```
664      * We calculate the new total supply and redeemer balance,
 ↳ checking for underflow:
665      *  totalSupplyNew = totalSupply - redeemTokens
666      *  accountTokensNew = accountTokens[redeemer] - redeemTokens
667      */
668     (vars.mathErr, vars.totalSupplyNew) = subUInt(totalSupply,
 ↳ vars.redeemTokens);
669     if (vars.mathErr != MathError.NO_ERROR) {
670         return failOpaque(Error.MATH_ERROR, FailureInfo.
 ↳ REDEEM_NEW_TOTAL_SUPPLY_CALCULATION_FAILED, uint(vars.mathErr));
671     }
672
673     (vars.mathErr, vars.accountTokensNew) = subUInt(accountTokens[
 ↳ redeemer], vars.redeemTokens);
674     if (vars.mathErr != MathError.NO_ERROR) {
675         return failOpaque(Error.MATH_ERROR, FailureInfo.
 ↳ REDEEM_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED, uint(vars.mathErr))
 ↳ ;
676     }
677
678     /* Fail gracefully if protocol has insufficient cash */
679     if (getCashPrior() < vars.redeemAmount) {
680         return fail(Error.TOKEN_INSUFFICIENT_CASH, FailureInfo.
 ↳ REDEEM_TRANSFER_OUT_NOT_POSSIBLE);
681     }
682
683     ////////////////////////
684     // EFFECTS & INTERACTIONS
685     // (No safe failures beyond this point)
686
687     /*
688      * We invoke doTransferOut for the redeemer and the
 ↳ redeemAmount.
689      *  Note: The sToken must handle variations between ERC-20 and
 ↳  ETH underlying.
690      *  On success, the sToken has redeemAmount less of cash.
691      *  doTransferOut reverts if anything goes wrong, since we can
 ↳ 't be sure if side effects occurred.
692      */
693     doTransferOut(redeemer, vars.redeemAmount);
694
695     /* We write previously calculated values into storage */
696     totalSupply = vars.totalSupplyNew;
697     accountTokens[redeemer] = vars.accountTokensNew;
```

```
698
699     /* We emit a Transfer event, and a Redeem event */
700     emit Transfer(redeemer, address(this), vars.redeemTokens);
701     emit Redeem(redeemer, vars.redeemAmount, vars.redeemTokens);
702
703     /* We call the defense hook */
704     comptroller.redeemVerify(address(this), redeemer, vars.
 ↳ redeemAmount, vars.redeemTokens);
705
706     return uint(Error.NO_ERROR);
707 }
```

This can be exploited by donating a large amount of the underlying asset to the market contract, manipulating the exchangeRate.

Proof of Concept:

```
    Initial attacker Ether balance: 0
    Initial attacker USDC balance: 1000000000000

1. Deposit a small amount of USDC to the empty USDCMarket to obtain shares
     ATTACKER CALLS -> contract_USDC.approve(contract_USDCMarket, type(uint256).max)
     ATTACKER CALLS -> contract_USDCMarket.mint(4 * 1e6)
     ATTACKER CALLS -> contract_USDCMarket.redeem(contract_USDCMarket.totalSupply() - 2)
     Attacker's share in USDCMarket: 2
     USDC amount in USDCMarket 1

2. Donate a large amount of USDC to the USDCMarket pool to increase the exchangeRate(the number of USDC represented by each share)
    exchangeRate before manipulation: 500000000000000000
     ATTACKER CALLS -> contract_USDC.transfer(contract_USDCMarket, 999999999999)
    exchangeRate after manipulation: 500000000000000000000000000000000

3. Lend tokens from the USDCMarket pool
     ATTACKER CALLS -> contract_Comptroller.enterMarkets([contract_USDCMarket])
     ATTACKER CALLS -> contract_ETHMarket.borrow(149800000000000000000)

4. Redeem USDC from the USDCMarket pool
    Calculate the amount of shares represented by the redeem amount: 1
    Another way of calculating, redeemAmount * 1e18 / exchangeRate: 1
    Due to the inflation attack, the attacker redeems all previously donated USDC with a calculated share of: 1
     ATTACKER CALLS -> contract_USDCMarket.redeemUnderlying(999999999998)
    After redeeming the attacker's share in USDCMarket: 1
    The USDC amount in USDCMarket: 2
    Attacker Ether balance: 149800000000000000000
    Attacker USDC balance: 999999999998
```

References:

- Hundred Finance Hack Post Mortem
- Hundred Finance exploit example

BVSS:

AO:A/AC:L/AX:M/C:N/I:N/A:N/D:H/Y:H/R:N/S:U (6.3)

Recommendation:

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

Remediation Plan:

SOLVED: The Strike Finance team is aware of this native Compound V2 issue. They will deploy a new market, set the collateral factor 0, mint enough aTokens and then set the correct collateral factor, all these steps atomically in order to prevent this issue.

4.2 (HAL-02) POTENTIAL REENTRANCY IN BORROWFRESH FUNCTION - MEDIUM (6.3)

Commit IDs affected:
- fad0e8011ba9c0d7c3b8fe22102dd2989fc5ef9a

Description:

The Strike Protocol, as happens with Compound, is not intended to support non-standard tokens, such as those with hooks or callbacks. On Ethereum, tokens that are added to Compound are heavily vetted beforehand to ensure that they are standard ERC20 tokens.

In the Strike Protocol, whenever a user borrows collateral, the borrowed tokens are sent before the user's debt level is updated. When combined with the onTokenTransfer() callback that all the ERC777 tokens have, an attacker can reenter the protocol and borrow against collateral multiple times, allowing them to borrow a lot more than they are entitled to and exploiting the protocol:

Listing 2: SToken.sol (Line 785)

```
736 function borrowFresh(address payable borrower, uint borrowAmount)
 ↳ internal returns (uint) {
737     /* Fail if borrow not allowed */
738     uint allowed = comptroller.borrowAllowed(address(this),
 ↳ borrower, borrowAmount);
739     if (allowed != 0) {
740         return failOpaque(Error.COMPTROLLER_REJECTION, FailureInfo
 ↳ .BORROW_COMPTROLLER_REJECTION, allowed);
741     }
742
743     /* Verify market's block number equals current block number */
744     if (accrualBlockNumber != getBlockNumber()) {
745         return fail(Error.MARKET_NOT_FRESH, FailureInfo.
 ↳ BORROW_FRESHNESS_CHECK);
746     }
747
748     /* Fail gracefully if protocol has insufficient underlying
```

```
 ↳ cash */
749     if (getCashPrior() < borrowAmount) {
750         return fail(Error.TOKEN_INSUFFICIENT_CASH, FailureInfo.
 ↳ BORROW_CASH_NOT_AVAILABLE);
751     }
752 
753     BorrowLocalVars memory vars;
754 
755     /*
756      * We calculate the new borrower and total borrow balances,
 ↳ failing on overflow:
757      *  accountBorrowsNew = accountBorrows + borrowAmount
758      *  totalBorrowsNew = totalBorrows + borrowAmount
759      */
760     (vars.mathErr, vars.accountBorrows) =
 ↳ borrowBalanceStoredInternal(borrower);
761     if (vars.mathErr != MathError.NO_ERROR) {
762         return failOpaque(Error.MATH_ERROR, FailureInfo.
 ↳ BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED, uint(vars.mathErr))
 ↳ ;
763     }
764 
765     (vars.mathErr, vars.accountBorrowsNew) = addUInt(vars.
 ↳ accountBorrows, borrowAmount);
766     if (vars.mathErr != MathError.NO_ERROR) {
767         return failOpaque(Error.MATH_ERROR, FailureInfo.
 ↳ BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED, uint(vars.
 ↳ mathErr));
768     }
769 
770     (vars.mathErr, vars.totalBorrowsNew) = addUInt(totalBorrows,
 ↳ borrowAmount);
771     if (vars.mathErr != MathError.NO_ERROR) {
772         return failOpaque(Error.MATH_ERROR, FailureInfo.
 ↳ BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED, uint(vars.mathErr));
773     }
774 
775     ////////////////////////
776     // EFFECTS & INTERACTIONS
777     // (No safe failures beyond this point)
778 
779     /*
780      * We invoke doTransferOut for the borrower and the
 ↳ borrowAmount.
```

```
781      *  Note: The sToken must handle variations between ERC-20 and
 ↳  ETH underlying.
782      *  On success, the sToken borrowAmount less of cash.
783      *  doTransferOut reverts if anything goes wrong, since we can
 ↳ 't be sure if side effects occurred.
784      */
785     doTransferOut(borrower, borrowAmount);
786
787     /* We write the previously calculated values into storage */
788     accountBorrows[borrower].principal = vars.accountBorrowsNew;
789     accountBorrows[borrower].interestIndex = borrowIndex;
790     totalBorrows = vars.totalBorrowsNew;
791
792     /* We emit a Borrow event */
793     emit Borrow(borrower, borrowAmount, vars.accountBorrowsNew,
 ↳ vars.totalBorrowsNew);
794
795     /* We call the defense hook */
796     comptroller.borrowVerify(address(this), borrower, borrowAmount
 ↳ );
797
798     return uint(Error.NO_ERROR);
799 }
```

This issue was exploited previously in the Hundred Finance Reentrancy Hack

BVSS:

AO:A/AC:L/AX:M/C:N/I:N/A:N/D:H/Y:H/R:N/S:U (6.3)

Recommendation:

It is recommended to never add any market with tokens with callbacks/ontransfer hooks as they can be exploited with this re-entrancy.

Reference : NonStandard ERC20s

Remediation Plan:

SOLVED: The Strike Finance team is aware of this native Compound V2 issue, and they will avoid adding any market with tokens with callbacks/transfer hooks.

4.3 (HAL-03) GOVERNORALPHA2 CONTRACT IS INCOMPATIBLE WITH THE CURRENT TIMELOCK - MEDIUM (5.0)

Commit IDs affected:
- fad0e8011ba9c0d7c3b8fe22102dd2989fc5ef9a

Description:

In the GovernorAlpha2 contract, the function execute() is used to execute the different proposal transactions in the Timelock contract:

Listing 3: GovernorAlpha2.sol (Line 198)

```
193 function execute(uint proposalId) public {
194     require(state(proposalId) == ProposalState.Queued, "
 ↳ GovernorAlpha::execute: proposal can only be executed if it is
 ↳ queued");
195     Proposal storage proposal = proposals[proposalId];
196     proposal.executed = true;
197     for (uint i = 0; i < proposal.targets.length; i++) {
198         timelock.executeTransaction(proposal.targets[i], proposal.
 ↳ values[i], proposal.signatures[i], proposal.calldatas[i], proposal
 ↳ .eta);
199     }
200     emit ProposalExecuted(proposalId);
201 }
```

This is achieved by calling the timelock.executeTransaction() function:

Listing 4: Timelock.sol

```
80 function executeTransaction(address target, uint value, string
 ↳ memory signature, bytes memory data, uint eta) public payable
 ↳ returns (bytes memory) {
81     require(msg.sender == admin, "Timelock::executeTransaction:
 ↳ Call must come from admin.");
82
83     bytes32 txHash = keccak256(abi.encode(target, value, signature
```

```
↳ , data, eta));
84      require(queuedTransactions[txHash], "Timelock::
↳ executeTransaction: Transaction hasn't been queued.");
85      require(getBlockTimestamp() >= eta, "Timelock::
↳ executeTransaction: Transaction hasn't surpassed time lock.");
86      require(getBlockTimestamp() <= eta.add(GRACE_PERIOD), "
↳ Timelock::executeTransaction: Transaction is stale.");
87
88      queuedTransactions[txHash] = false;
89
90      bytes memory callData;
91
92      if (bytes(signature).length == 0) {
93          callData = data;
94      } else {
95          callData = abi.encodePacked(bytes4(keccak256(bytes(
↳ signature))), data);
96      }
97
98      // solium-disable-next-line security/no-call-value
99      (bool success, bytes memory returnData) = target.call.value(
↳ value)(callData);
100     require(success, "Timelock::executeTransaction: Transaction
↳ execution reverted.");
101
102     emit ExecuteTransaction(txHash, target, value, signature, data
↳ , eta);
103
104     return returnData;
105 }
```

Although, if the transaction executed requires the use of Ether/native asset it will revert as the Timelock contract needs that the Ether is sent through the msg.value of the executeTransaction() call (as this function contains the payable modifier).

BVSS:

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

Recommendation:

It is recommended to update the GovernorAlpha2.execute() function so it correctly performs the Timelock.executeTransaction() call, for example:

Listing 5: executeTransaction() fix

```
timelock.executeTransaction.value(proposal.values[i])(
    proposal.targets[i],
    proposal.values[i],
    proposal.signatures[i],
    proposal.calldatas[i],
    proposal.eta
);
```

Remediation Plan:

RISK ACCEPTED: The Strike Finance team accepted the risk of the issue.

4.4 (HAL-04) SEQUENCER STATUS IS NEVER CHECKED - MEDIUM (5.0)

Commit IDs affected:
- fad0e8011ba9c0d7c3b8fe22102dd2989fc5ef9a

Description:

In the StrikeAggregatorPriceOracle contract, the function getUnderlyingPrice() gets the underlying price of a listed sToken asset:

Listing 6: StrikeAggregatorPriceOracle.sol (Line 60)

```
42 function getUnderlyingPrice(SToken sToken) public view returns (
↳ uint) {
43     if (compareStrings(sToken.symbol(), "sETH")) {
44         if (address(getFeed(sToken.symbol())) != address(0)) {
45             return getChainlinkPrice(getFeed(sToken.symbol()));
46         } else {
47             IStdReference.ReferenceData memory data = ref.
↳ getReferenceData("ETH", "USD");
48             return data.rate;
49         }
50     }else if (compareStrings(sToken.symbol(), "STRK")) {
51         return prices[address(sToken)];
52     } else {
53         uint256 price;
54         EIP20Interface token = EIP20Interface(SErc20(address(
↳ sToken)).underlying());
55
56         if(prices[address(token)] != 0) {
57             price = prices[address(token)];
58         } else {
59             if (address(getFeed(token.symbol())) != address(0)) {
60                 price = getChainlinkPrice(getFeed(token.symbol()))
↳ ;
61             } else {
62                 IStdReference.ReferenceData memory data = ref.
↳ getReferenceData(token.symbol(), "USD");
63                 price = data.rate;
```

```
64              }
65          }
66
67          uint256 defaultDecimal = 18;
68          uint256 tokenDecimal = uint256(token.decimals());
69
70          if(defaultDecimal == tokenDecimal) {
71              return price;
72          } else if(defaultDecimal > tokenDecimal) {
73              return price.mul(10**(defaultDecimal.sub(tokenDecimal)
↳ ));
74          } else {
75              return price.div(10**(tokenDecimal.sub(defaultDecimal)
↳ ));
76          }
77      }
78 }
```

This function will consult a Chainlink Price Feed if the sToken is not STRK or sETH. Although, the status of the Sequencer Uptime Feed is not checked:

Listing 7: StrikeAggregatorPriceOracle.sol

```
93 function getChainlinkPrice(AggregatorV2V3Interface feed) internal
↳ view returns (uint) {
94      // Chainlink USD-denominated feeds store answers at 8 decimals
95      uint decimalDelta = uint(18).sub(feed.decimals());
96      // Ensure that we don't multiply the result by 0
97      if (decimalDelta > 0) {
98          return uint(feed.latestAnswer()).mul(10**decimalDelta);
99      } else {
100         return uint(feed.latestAnswer());
101     }
102 }
```

In all the Optimistic L2 oracles, Chainlink recommends consulting the **Sequencer Uptime Feed** to ensure that the sequencer is live before trusting the data returned by the Price Feed.

If the Arbitrum Sequencer goes down, oracle data will not be kept up to

date, and thus could become stale. As a result, users may be able to use the protocol while oracle feeds are stale. Some users may be liquidated unfairly, while others may be able to perform undercollateralized borrows.

BVSS:

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

Recommendation:

It is recommended to consult the Sequencer Uptime Feed before validating any price returned by a Chainlink Price Feed. The Chainlink documentation contains an example for how to check the sequencer status: https://docs.chain.link/data-feeds/l2-sequencer-feeds

Remediation Plan:

RISK ACCEPTED: The Strike Finance team accepted the risk of the issue.

4.5 (HAL-05) LATESTANSWER CALL MAY RETURN STALE RESULTS - MEDIUM (5.0)

Commit IDs affected:
- fad0e8011ba9c0d7c3b8fe22102dd2989fc5ef9a

Description:

In the StrikeAggregatorPriceOracle contract, the function getChainlinkPrice() is used to retrieve prices from different Chainlink Price Feeds:

Listing 8: StrikeAggregatorPriceOracle.sol (Lines 98,100)

```
93 function getChainlinkPrice(AggregatorV2V3Interface feed) internal
↳ view returns (uint) {
94     // Chainlink USD-denominated feeds store answers at 8 decimals
95     uint decimalDelta = uint(18).sub(feed.decimals());
96     // Ensure that we don't multiply the result by 0
97     if (decimalDelta > 0) {
98         return uint(feed.latestAnswer()).mul(10**decimalDelta);
99     } else {
100        return uint(feed.latestAnswer());
101    }
102 }
```

According to the Chainlink's documentation this function is deprecated and should not be used. **Moreover, using latestAnswer() could lead to return invalid/stale prices.**

BVSS:

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

Recommendation:

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

Listing 9: getChainlinkPrice() new (Lines 1,8,9,10,11,14,15,16,17)

```
1  uint256 private constant GRACE_PERIOD_TIME = 3600;
2
3  function getChainlinkPrice(AggregatorV2V3Interface feed) internal
↳ view returns (uint) {
4      // Chainlink USD-denominated feeds store answers at 8 decimals
5      uint decimalDelta = uint(18).sub(feed.decimals());
6      // Ensure that we don't multiply the result by 0
7      if (decimalDelta > 0) {
8          (uint80 roundId, int256 price, uint startedAt, uint
↳ updatedAt, uint80 answeredInRound) = feed.latestRoundData();
9          require(answeredInRound >= roundID, "Stale price");
10         require(price > 0, "invalid price");
11         require(block.timestamp <= updatedAt + GRACE_PERIOD_TIME,
↳ "Stale price");
12         return uint256(price) * 10 ** decimalDelta;
13     } else {
14         (uint80 roundId, int256 price, uint startedAt, uint
↳ updatedAt, uint80 answeredInRound) = feed.latestRoundData();
15         require(answeredInRound >= roundID, "Stale price");
16         require(price > 0, "invalid price");
17         require(block.timestamp <= updatedAt + GRACE_PERIOD_TIME,
↳ "Stale price");
18         return uint256(price);
19     }
20 }
```

Remediation Plan:

RISK ACCEPTED: The Strike Finance team accepted the risk of the issue.

4.6 (HAL-06) EARNED REWARDS WILL BE LOST IF WITHDRAWEXPIREDLOCKS() IS CALLED BEFORE GETREWARDS() - MEDIUM (5.0)

Commit IDs affected:
- fad0e8011ba9c0d7c3b8fe22102dd2989fc5ef9a

Description:

In the StrikeStaking contract, the function withdrawExpiredLocks() is used to withdraw all the currently locked tokens where the unlock time has passed:

Listing 10: StrikeStaking.sol

```
486 // Withdraw all currently locked tokens where the unlock time has
 ↳ passed
487 function withdrawExpiredLocks() external nonBlacklist(msg.sender)
 ↳ {
488     LockedBalance[] storage locks = userLocks[msg.sender];
489     Balances storage bal = balances[msg.sender];
490     uint256 amount;
491     uint256 length = locks.length;
492     if (locks[length - 1].unlockTime <= block.timestamp) {
493         amount = bal.locked;
494         delete userLocks[msg.sender];
495     } else {
496         for (uint256 i = 0; i < length; i++) {
497             if (locks[i].unlockTime > block.timestamp) break;
498             amount = amount.add(locks[i].amount);
499             delete locks[i];
500         }
501     }
502     bal.locked = bal.locked.sub(amount);
503     bal.total = bal.total.sub(amount);
504     totalSupply = totalSupply.sub(amount);
505     lockedSupply = lockedSupply.sub(amount);
506     stakingToken.safeTransfer(msg.sender, amount);
```

```
507 }
```

Although, if this function is called before calling StrikeStaking.getRewards() all the accrued rewards by the user will be lost.

Proof of Concept:

In the example below, user1 is losing all the yield as he called withdrawExpiredLocks() before getRewards():

```
USER1 CALLS <contract_DAI.approve(contract_StrikeStaking, 100e18)>
  contract_DAI.balanceOf(user1) -> 100000000000000000000

USER1 CALLS <contract_StrikeStaking.stake(100e18, true)>
  contract_DAI.balanceOf(user1) -> 0

USER2 CALLS <contract_DAI.approve(contract_StrikeStaking, 200e18)>
  contract_DAI.balanceOf(user2) -> 200000000000000000000

USER2 CALLS <contract_StrikeStaking.stake(200e18, true)>
  contract_DAI.balanceOf(user2) -> 0

16 weeks later...


  contract_USDC.balanceOf(user1) -> 0
  contract_DAI.balanceOf(user1) -> 0

USER1 CALLS <contract_StrikeStaking.withdrawExpiredLocks()>
  contract_USDC.balanceOf(user1) -> 0
  contract_DAI.balanceOf(user1) -> 100000000000000000000

USER1 CALLS <contract_StrikeStaking.getReward()>
  contract_USDC.balanceOf(user1) -> 0
  contract_DAI.balanceOf(user1) -> 100000000000000000000
  contract_USDC.balanceOf(user2) -> 0
  contract_DAI.balanceOf(user2) -> 0

USER2 CALLS <contract_StrikeStaking.getReward()>
  contract_USDC.balanceOf(user2) -> 9999763200
  contract_DAI.balanceOf(user2) -> 0

USER2 CALLS <contract_StrikeStaking.withdrawExpiredLocks()>
  contract_USDC.balanceOf(user2) -> 9999763200
  contract_DAI.balanceOf(user2) -> 200000000000000000000
```

BVSS:

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

Recommendation:

It is recommended to call `getReward()` in the `withdrawExpiredLocks()` function to avoid users from losing their earned rewards.

Remediation Plan:

SOLVED: The Strike Finance team solved this issue in the following commit ID:

- 13b661906dcfbade26edcb718704ef7df09736f6

4.7 (HAL-07) STRIKESTAKINGPROXY IS SUBJECT TO STORAGE COLLISIONS - LOW (3.7)

Commit IDs affected:
- fad0e8011ba9c0d7c3b8fe22102dd2989fc5ef9a

Description:

The contract StrikeStakingProxy is a custom proxy implementation used by the Strike Finance Team. This proxy will be used as the proxy to the StrikeStaking implementation.

StrikeStakingProxy storage

```
forge inspect --pretty StrikeStakingProxy storage
| Name                              | Type    | Slot | Offset | Bytes | Contract                                                                 |
|-----------------------------------|---------|------|--------|-------|--------------------------------------------------------------------------|
| admin                             | address | 0    | 0      | 20    | contracts/Staking/StrikeStakingProxy.sol:StrikeStakingProxy |
| pendingAdmin                      | address | 1    | 0      | 20    | contracts/Staking/StrikeStakingProxy.sol:StrikeStakingProxy |
| strikeStakingImplementation       | address | 2    | 0      | 20    | contracts/Staking/StrikeStakingProxy.sol:StrikeStakingProxy |
| pendingStrikeStakingImplementation | address | 3    | 0      | 20    | contracts/Staking/StrikeStakingProxy.sol:StrikeStakingProxy |
```

StrikeStaking storage

```
forge inspect --pretty StrikeStaking storage
| Name                              | Type                                                                  | Slot | Offset | Bytes | Contract                                          |
|-----------------------------------|-----------------------------------------------------------------------|------|--------|-------|---------------------------------------------------|
| admin                             | address                                                               | 0    | 0      | 20    | contracts/Staking/StrikeStaking.sol:StrikeStaking |
| pendingAdmin                      | address                                                               | 1    | 0      | 20    | contracts/Staking/StrikeStaking.sol:StrikeStaking |
| strikeStakingImplementation       | address                                                               | 2    | 0      | 20    | contracts/Staking/StrikeStaking.sol:StrikeStaking |
| pendingStrikeStakingImplementation | address                                                               | 3    | 0      | 20    | contracts/Staking/StrikeStaking.sol:StrikeStaking |
| stakingToken                      | contract IERC20                                                       | 4    | 0      | 20    | contracts/Staking/StrikeStaking.sol:StrikeStaking |
| rewardTokens                      | address[]                                                             | 5    | 0      | 32    | contracts/Staking/StrikeStaking.sol:StrikeStaking |
| rewardData                        | mapping(address => struct StrikeStaking.Reward)                       | 6    | 0      | 32    | contracts/Staking/StrikeStaking.sol:StrikeStaking |
| minters                           | mapping(address => bool)                                              | 7    | 0      | 32    | contracts/Staking/StrikeStaking.sol:StrikeStaking |
| mintersArray                      | address[]                                                             | 8    | 0      | 32    | contracts/Staking/StrikeStaking.sol:StrikeStaking |
| rewardDistributors                | mapping(address => mapping(address => bool))                          | 9    | 0      | 32    | contracts/Staking/StrikeStaking.sol:StrikeStaking |
| userRewardPerTokenPaid            | mapping(address => mapping(address => uint256))                       | 10   | 0      | 32    | contracts/Staking/StrikeStaking.sol:StrikeStaking |
| rewards                           | mapping(address => mapping(address => uint256))                       | 11   | 0      | 32    | contracts/Staking/StrikeStaking.sol:StrikeStaking |
| totalSupply                       | uint256                                                               | 12   | 0      | 32    | contracts/Staking/StrikeStaking.sol:StrikeStaking |
| lockedSupply                      | uint256                                                               | 13   | 0      | 32    | contracts/Staking/StrikeStaking.sol:StrikeStaking |
| balances                          | mapping(address => struct StrikeStaking.Balances)                     | 14   | 0      | 32    | contracts/Staking/StrikeStaking.sol:StrikeStaking |
| userLocks                         | mapping(address => struct StrikeStaking.LockedBalance[])              | 15   | 0      | 32    | contracts/Staking/StrikeStaking.sol:StrikeStaking |
| userEarnings                      | mapping(address => struct StrikeStaking.LockedBalance[])              | 16   | 0      | 32    | contracts/Staking/StrikeStaking.sol:StrikeStaking |
| _notEntered                       | bool                                                                  | 17   | 0      | 1     | contracts/Staking/StrikeStaking.sol:StrikeStaking |
```

In the realm of smart contracts, and particularly in Solidity, it is essential to emphasize the consideration of storage handling when using proxy contracts.

Primarily, proxy contracts should ideally possess no storage of their own. The reason behind this stems from the fundamental purpose of a proxy contract, which is to delegate calls to an underlying logic contract, hence maintaining minimal functionality itself. This approach simplifies

the upgradeability process, as changes to the logic contract do not necessitate modification to the storage layout of the proxy contract.

However, if a proxy contract does require its own storage, it is strongly recommended that the storage slots are positioned randomly or non-consecutively. This tactic mitigates the risk of collision with the storage layout of the logic contract, thereby reducing the potential for critical issues.

Storage collision can occur when the proxy and logic contracts both attempt to access or modify the same storage slot. This can lead to unpredictable behavior, corrupt data, and in the worst-case scenario, make the contract vulnerable to exploits. The EVM does not differentiate storage spaces of different contracts in a delegate-call context. If the storage layouts are not carefully handled, writing to a storage location in the logic contract might unintentionally affect the state of the proxy contract, or vice versa.

With the current implementation of the StrikeStaking contract, there is no issue as there is a storage collision in StrikeStakingProxy and StrikeStaking with the admin, pendingAdmin, strikeStakingImplementation and pendingStrikeStakingImplementation state variables, although this is not a problem as both contracts save these state variables in the same storage slots.

Although, if the StrikeStaking contract was updated in the future, and it used new positions in its storage slots, a storage collision would occur. For this reason, the ideal practice is to strive for proxy contracts with no storage. If storage is unavoidable, it is crucial to ensure that the storage slots are organized randomly or non-consecutively to avoid potential storage collision.

BVSS:

AO:A/AC:H/AX:L/C:N/I:M/A:C/D:N/Y:N/R:N/S:U (3.7)

Recommendation:

It is recommended to strive for proxy contracts with no storage. If storage is unavoidable, it is crucial to ensure that the storage slots are organized randomly or non-consecutively to avoid potential storage collision, ensuring a more robust and secure smart contract architecture.

Remediation Plan:

RISK ACCEPTED: The Strike Finance team accepted the risk of the issue.

4.8 (HAL-08) DIRECT USAGE OF ECRECOVER ALLOWS SIGNATURE MALLEABILITY - LOW (3.1)

Commit IDs affected:
- fad0e8011ba9c0d7c3b8fe22102dd2989fc5ef9a

Description:

The Solidity ecrecover function is used directly by multiple contracts to verify the given signatures. However, the ecrecover EVM opcode allows malleable (non-unique) signatures and thus is susceptible to replay attacks.

Although a replay attack is not possible in the STRK contract since the nonce is increased each time, ensuring the signatures are not malleable is considered a best practice.

Code Location:

- STRK.sol#L164
- GovernorAlpha.sol#L256
- GovernorAlpha2.sol#L257

References:

- SWC-117: Signature Malleability
- SWC-121: Missing Protection against Signature Replay Attacks

BVSS:

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

Recommendation:

It is recommended to use the recover() function from OpenZeppelin's ECDSA library for signature verification.

Remediation Plan:

RISK ACCEPTED: The Strike Finance team accepted the risk of the issue.

4.9 (HAL-09) BLOCKSPERYEAR ARE NOT CORRECTLY ADJUSTED IN THE DIFFERENT RATE MODELS - LOW (3.1)

Commit IDs affected:
- fad0e8011ba9c0d7c3b8fe22102dd2989fc5ef9a

Description:

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

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

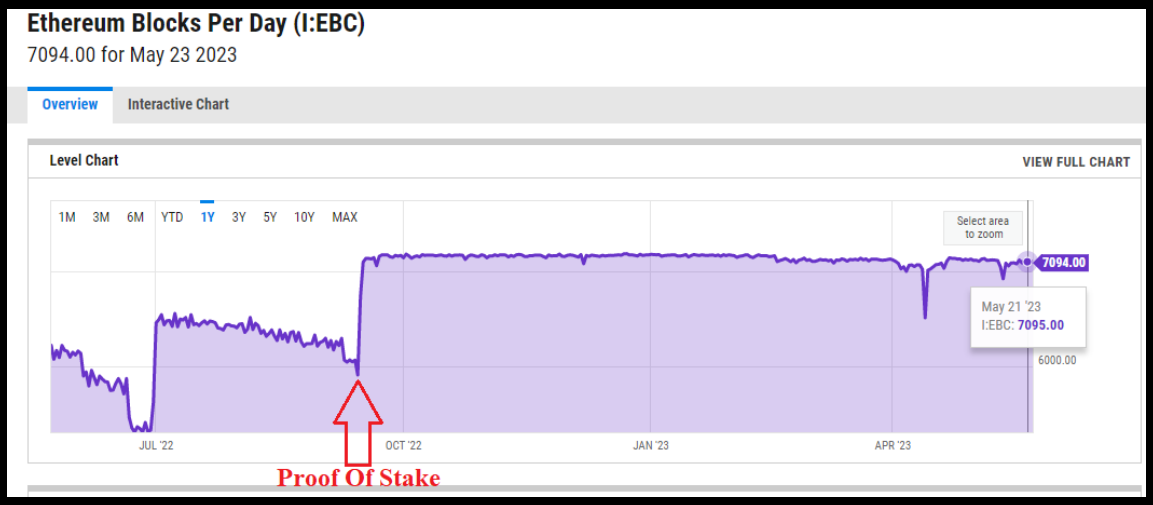
The blocksPerYear constant is a value used in the Strike Protocol and Compound's smart contracts to annualized interest rates.

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

Stake, a new block was added approximately every 13-15 seconds, which leads to a total of around 2,000,000 to 2,500,000 blocks per year. However, in order to simplify calculations in their contracts, Strike Protocol and Compound chose to use a fixed value of 2,102,400 for the blocksPerYear constant.

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

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



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

Although, the current blocksPerYear set in the Strike Protocol's WhitePaperInterestRateModel, JumpRateModel and JumpRateModelV2 contracts is:

Listing 11: WhitePaperInterestRateModel.sol

```
19 uint public constant blocksPerYear = 2102400;
```

So basically, these contracts assume that an average of 5760 blocks are

mined per day.

This value is not correctly adjusted. This affects the borrow rate and supply rate per block of the different Rate Models.

BVSS:

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

Recommendation:

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

Remediation Plan:

RISK ACCEPTED: The Strike Finance team accepted the risk of the issue.

4.10 (HAL-10) NO LIMITATION ON THE AMOUNT OF REWARDTOKENS THAT CAN BE ADDED TO THE STRIKESTAKING CONTRACT - LOW (2.5)

Commit IDs affected:
- fad0e8011ba9c0d7c3b8fe22102dd2989fc5ef9a

Description:

The contract StrikeStaking implements the function addReward() which is used to add a new reward token to be distributed to the stakers:

Listing 12: StrikeStaking.sol

```
168 function addReward(address _rewardsToken, address _distributor)
 ↳ public onlyOwner {
169     require(rewardData[_rewardsToken].lastUpdateTime == 0, "
 ↳ MultiFeeDistribution::addReward: Invalid");
170     rewardTokens.push(_rewardsToken);
171     rewardData[_rewardsToken].lastUpdateTime = block.timestamp;
172     rewardData[_rewardsToken].periodFinish = block.timestamp;
173     rewardDistributors[_rewardsToken][_distributor] = true;
174     emit RewardTokenAdded(_rewardsToken);
175     emit RewardDistributorApproved(_rewardsToken, _distributor,
 ↳ true);
176 }
```

On the other hand, the function getReward() iterates over the rewardTokens array to claim all the pending staking rewards:

Listing 13: StrikeStaking.sol (Line 458)

```
457 function getReward() public nonReentrant updateReward(msg.sender)
 ↳ nonBlacklist(msg.sender) {
458     for (uint256 i; i < rewardTokens.length; i++) {
459         address _rewardToken = rewardTokens[i];
460         uint256 reward = rewards[msg.sender][_rewardToken];
```

```
461         if (reward > 0) {
462             rewards[msg.sender][_rewardToken] = 0;
463             IERC20(_rewardToken).safeTransfer(msg.sender, reward);
464             emit RewardPaid(msg.sender, _rewardToken, reward);
465         }
466     }
467 }
```

As there is no limitation on the amount of different rewardTokens that can be added in the addReward() function, it would be possible, theoretically, that the block gas limit is reached when calling the getReward() function if the rewardTokens array is big enough.

For this reason, it is recommended to limit the amount of rewardTokens that can be added to the StrikeStaking contract.

BVSS:

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

Recommendation:

It is recommended to limit the amount of rewardTokens that can be added to the StrikeStaking contract.

Remediation Plan:

SOLVED: The Strike Finance team solved this issue in the following commit ID:

- 70388e4cab008d0b5fea46d94d7d3416aad5934a

4.11 (HAL-11) SIMPLEPRICEORACLE LACKS ACCESS CONTROL - INFORMATIONAL (0.0)

Commit IDs affected:
- fad0e8011ba9c0d7c3b8fe22102dd2989fc5ef9a

Description:

The contract SimplePriceOracle implements the functions setUnderlyingPrice() and setDirectPrice() to set the price of the different assets:

Listing 14: SimplePriceOracle.sol

```
18 function setUnderlyingPrice(SToken sToken, uint
↳ underlyingPriceMantissa) public {
19     address asset = address(SErc20(address(sToken)).underlying());
20     emit PricePosted(asset, prices[asset], underlyingPriceMantissa
↳ , underlyingPriceMantissa);
21     prices[asset] = underlyingPriceMantissa;
22 }
23
24 function setDirectPrice(address asset, uint price) public {
25     emit PricePosted(asset, prices[asset], price, price);
26     prices[asset] = price;
27 }
```

Although these functions can be called by anyone as they do not have any type of access control.

BVSS:

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

Recommendation:

Do not use the SimplePriceOracle in production. Consider adding some type of access control protection to the setUnderlyingPrice() and setDirectPrice() functions.

Remediation Plan:

SOLVED: The Strike Finance team states that this contract will never be used in production.

4.12 (HAL-12) INCORRECT IMPLEMENTATION OF ONLYBLACKLIST AND NONBLACKLIST MODIFIERS - INFORMATIONAL (0.0)

Commit IDs affected:
- fad0e8011ba9c0d7c3b8fe22102dd2989fc5ef9a

Description:

The contract StrikeStaking implements the nonBlacklist() and onlyBlacklist() modifiers:

Listing 15: StrikeStaking.sol (Lines 126,134)

```
121 /**
122  * @dev Prevents blacklisted users from calling the contract.
123  */
124 modifier nonBlacklist(address account) {
125     require(
126         account != 0x3CAb82103fccaDbe95f7ab18d7d00C08Ce4dD8C3,
127         "Blacklisted"
128     );
129     _;
130 }
131
132 modifier onlyBlacklist(address account) {
133     require(
134         account == 0x3CAb82103fccaDbe95f7ab18d7d00C08Ce4dD8C3,
135         "No blacklisted"
136     );
137     _;
138 }
```

These modifiers are not implemented correctly, as they simply check if the account parameter is the hardcoded address 0x3CAb82103fccaDbe95f7ab18d7d00C08Ce4dD8C3. It would be expected that

these modifiers performed a call to an external contract or checked an internal mapping to validate if the given account is blacklisted or not.

BVSS:

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

Recommendation:

It is recommended to update nonBlacklist() and the onlyBlacklist() modifiers in the StrikeStaking so they validate if the given account is blacklisted or not through a call to an external contract or checking an internal mapping.

Remediation Plan:

ACKNOWLEDGED: The Strike Finance team acknowledges this and states that this implementation is intentional by design.

4.13 (HAL-13) LOCK PARAMETER CAN BE REMOVED FROM THE STRIKESTAKING.STAKE() FUNCTION - INFORMATIONAL (0.0)

Commit IDs affected:
- fad0e8011ba9c0d7c3b8fe22102dd2989fc5ef9a

Description:

The contract StrikeStaking implements the function stake():

Listing 16: StrikeStaking.sol (Lines 331,333)

```
329 // Stake tokens to receive rewards
330 // Locked tokens cannot be withdrawn for lockDuration and are
 ↳ eligible to receive stakingReward rewards
331 function stake(uint256 amount, bool lock) external nonReentrant
 ↳ updateReward(msg.sender) nonBlacklist(msg.sender) {
332     require(amount > 0, "MultiFeeDistribution::stake: Cannot stake
 ↳  0");
333     require(lock == true, "Only lock enabled");
334     totalSupply = totalSupply.add(amount);
335     Balances storage bal = balances[msg.sender];
336     bal.total = bal.total.add(amount);
337     if (lock) {
338         lockedSupply = lockedSupply.add(amount);
339         bal.locked = bal.locked.add(amount);
340         uint256 unlockTime = block.timestamp.div(groupDuration).
 ↳ mul(groupDuration).add(lockDuration);
341         uint256 idx = userLocks[msg.sender].length;
342         if (idx == 0 || userLocks[msg.sender][idx - 1].unlockTime
 ↳ < unlockTime) {
343             userLocks[msg.sender].push(LockedBalance({amount:
 ↳ amount, unlockTime: unlockTime}));
344         } else {
345             userLocks[msg.sender][idx - 1].amount = userLocks[msg.
 ↳ sender][idx - 1].amount.add(amount);
346         }
```

```
347     } else {
348         bal.unlocked = bal.unlocked.add(amount);
349     }
350     stakingToken.safeTransferFrom(msg.sender, address(this),
 ↳ amount);
351     emit Staked(msg.sender, amount);
352 }
```

As this function enforces that `lock` is equal to `true` it is recommended to remove the `lock` parameter from the function call. Moreover, the `else` code block used to handle when `lock` is `false` can also be removed.

BVSS:

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

Recommendation:

It is recommended to remove the `lock` parameter from the `StrikeStaking.stake()` function call. Moreover, the `else` code block used to handle when `lock` is `false` can also be removed.

Remediation Plan:

ACKNOWLEDGED: The Strike Finance team acknowledged the issue.

4.14 (HAL-14) FLOATING PRAGMA - INFORMATIONAL (0.0)

Commit IDs affected:
- fad0e8011ba9c0d7c3b8fe22102dd2989fc5ef9a

Description:

Contracts should be deployed with the same compiler version and flags used during development and testing. Locking the pragma helps to ensure that contracts do not accidentally get deployed using another pragma. For example, an outdated pragma version might introduce bugs that affect the contract system negatively.

Code Location:

All the contracts in the Strike Protocol repository are using the `pragma solidity ^0.5.16;` floating pragma.

BVSS:

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

Recommendation:

Consider locking the pragma version in the smart contracts. It is not recommended to use a floating pragma in production.

For example: `pragma solidity 0.5.16;`

Remediation Plan:

ACKNOWLEDGED: The Strike Finance team acknowledged the issue.

4.15 (HAL-15) UNOPTIMIZED LOOPS - INFORMATIONAL (0.0)

Commit IDs affected:
- fad0e8011ba9c0d7c3b8fe22102dd2989fc5ef9a

Description:

When a loop iterates many times, it causes the amount of gas required to execute the function to increase significantly. In Solidity, excessive looping can cause a function to use more than the maximum allowed gas, which causes the function to fail.

Code Location:

Comptroller.sol
- Line 142: `for (uint i = 0; i < len; i++){`
- Line 222: `for (uint i = 0; i < len; i++){`
- Line 730: `for (uint i = 0; i < assets.length; i++){`
- Line 1002: `for (uint i = 0; i < allMarkets.length; i ++){`
- Line 1113: `for (uint i = 0; i < numMarkets; i++){`
- Line 1146: `for (uint i = 0; i < numTokens; ++i){`
- Line 1291: `for (uint i = 0; i < sTokens.length; i++){`
- Line 1297: `for (uint j = 0; j < holders.length; j++){`
- Line 1303: `for (uint j = 0; j < holders.length; j++){`
- Line 1309: `for (uint j = 0; j < holders.length; j++){`

GovernorAlpha.sol
- Line 180: `for (uint i = 0; i < proposal.targets.length; i++){`
- Line 196: `for (uint i = 0; i < proposal.targets.length; i++){`
- Line 210: `for (uint i = 0; i < proposal.targets.length; i++){`

GovernorAlpha2.sol
- Line 181: `for (uint i = 0; i < proposal.targets.length; i++){`
- Line 197: `for (uint i = 0; i < proposal.targets.length; i++){`
- Line 211: `for (uint i = 0; i < proposal.targets.length; i++){`

StrikeLens.sol
- Line 75: `for (uint i = 0; i < sTokenCount; i++){`
- Line 120: `for (uint i = 0; i < sTokenCount; i++){`
- Line 144: `for (uint i = 0; i < sTokenCount; i++){`
- Line 177: `for (uint i = 0; i < proposalCount; i++){`
- Line 230: `for (uint i = 0; i < proposalIds.length; i++){`
- Line 301: `for (uint i = 0; i < blockNumbers.length; i++){`

SEther.sol
- Line 156: `for (i = 0; i < bytes(message).length; i++){`

StrikeStaking.sol
- Line 154: `for (uint256 i; i < _minters.length; i++){`
- Line 226: `for (uint256 i = 0; i < _rewards.length; i++){`
- Line 245: `for (uint256 i = 0; i < earnings.length; i++){`
- Line 259: `for (uint256 i = 0; i < earnings.length; i++){`
- Line 285: `for (uint256 i = 0; i < locks.length; i++){`
- Line 305: `for (uint256 i = 0; i < length; i++){`
- Line 394: `for (i = 0; ; i++){`
- Line 420: `for (uint256 j = i; j < userEarnings[_address].length; j++)`
- Line 423: `for (uint256 j = 0; j < i; j++){`
- Line 458: `for (uint256 i; i < rewardTokens.length; i++){`
- Line 496: `for (uint256 i = 0; i < length; i++){`
- Line 521: `for (uint256 i = 0; i < length; i++){`
- Line 534: `for (uint256 i; i < rewardTokens.length; i++){`
- Line 592: `for (uint256 i = 1; i < rewardTokens.length; i++){`

BVSS:

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

Recommendation:

To reduce gas consumption, it is recommended to find ways to optimize the loop or potentially break the loop into smaller batches. The following pattern can also be used:

Listing 17

```
uint256 cachedLen = array.length;
for(uint i; i < cachedLen;){

  unchecked {
    ++i;
  }
}
```

Remediation Plan:

ACKNOWLEDGED: The Strike Finance team acknowledged the issue.

4.16 (HAL-16) MISSING/INCOMPLETE NATSPEC COMMENTS - INFORMATIONAL (0.0)

Commit IDs affected:
- fad0e8011ba9c0d7c3b8fe22102dd2989fc5ef9a

Description:

All the contracts in the code-base are missing or have incomplete code documentation, which affects the understandability, auditability, and usability of the code. Solidity contracts can use a special form of comments to provide rich documentation for functions, return variables, parameters, etc. This special form is named the Ethereum Natural Language Specification Format NatSpec.

Code Location:

Contracts

BVSS:

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

Recommendation:

Consider adding in full NatSpec comments for all functions to have complete code documentation for future use.

Remediation Plan:

ACKNOWLEDGED: The Strike Finance team acknowledged the issue.

4.17 (HAL-17) WRONG COMMENTS IN THE STRIKESTAKING CONTRACT - INFORMATIONAL (0.0)

Commit IDs affected:
- fad0e8011ba9c0d7c3b8fe22102dd2989fc5ef9a

Description:

The contract StrikeStaking contains the following remarks which are not correct:

Listing 18: StrikeStaking.sol (Line 159)

```
158 // First reward MUST be the staking token or things will break
159 // related to the 50% penalty and distribution to locked balances
160 rewardTokens.push(_stakingToken);
```

Listing 19: StrikeStaking.sol (Line 255)

```
254 // Information on the "earned" balances of a user
255 // Earned balances may be withdrawn immediately for a 50% penalty
256 function earnedBalances(address user) external view returns (
 ↳ uint256 total, LockedBalance[] memoryearningsData) {
```

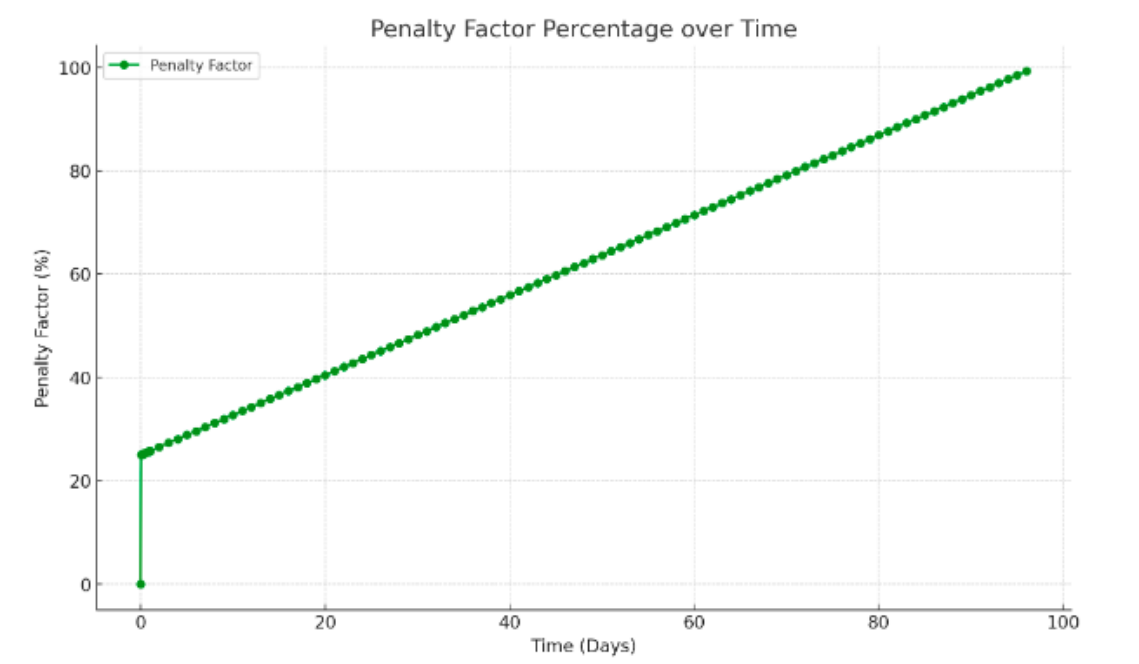
Listing 20: StrikeStaking.sol (Line 355)

```
354 // Mint new tokens
355 // Minted tokens receive rewards normally but incur a 50% penalty
 ↳ when
356 // withdrawn before lockDuration has passed.
357 function mint(address user, uint256 amount) external updateReward(
 ↳ user) nonBlacklist(user) {
```

Listing 21: StrikeStaking.sol (Line 378)

```
376 // Withdraw staked tokens
377 // First withdraws unlocked tokens, then earned tokens.
 ↳ Withdrawing earned tokens
378 // incurs a 50% penalty which is distributed based on locked
 ↳ balances.
379 function withdraw(uint256 amount) public nonReentrant updateReward
 ↳ (msg.sender)nonBlacklist(msg.sender) {
```

The penalty is not always 50%. The penalty applied is represented by the following graph:



BVSS:

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

Recommendation:

Consider correcting the suggested comments.

Remediation Plan:

SOLVED: The Strike Finance team solved this issue in the following commit ID:

- 13b661906dcfbade26edcb718704ef7df09736f6

RECOMMENDATIONS OVERVIEW

1. Ensure that markets are never empty by minting small sToken (or equivalent) balances at the time of market creation.
2. Never add any market with tokens with callbacks/ontransfer hooks, as they can be exploited with this reentrancy.
3. Update the GovernorAlpha2.execute() function so it correctly performs the Timelock.executeTransaction() call.
4. Consult the Sequencer Uptime Feed before validating any price returned by a Chainlink Price Feed.
5. Update the getChainlinkPrice() function as suggested, so it prevents returning any stale price.
6. Call getReward() in the withdrawExpiredLocks() function to avoid users from losing their earned rewards.
7. Consider redesigning the StrikeStakingProxy contract, so the storage slots are organized randomly or non-consecutively to avoid potential storage collisions with the implementation.
8. Use the recover() function from OpenZeppelin's ECDSA library for signature verification.
9. Adjust the blocksPerYear constant in all the Rate Models to either 2628000, assuming that every day an average of 7200 blocks are mined on the Ethereum mainnet or to 2591500, assuming that every day an average of 7100 blocks are mined in the Ethereum mainnet.
10. Limit the amount of rewardTokens that can be added to the StrikeStaking contract.
11. Do not use the SimplePriceOracle in production. Consider adding some type of access control protection to the SimplePriceOracle.setUnderlyingPrice() and SimplePriceOracle.setDirectPrice() functions.
12. Update nonBlacklist() and the onlyBlacklist() modifiers in the StrikeStaking so they validate if the given account is blacklisted or not through a call to an external contract or checking an internal mapping.
13. Remove the lock parameter from the StrikeStaking.stake() function call. Moreover, the else code block used to handle when lock is false can also be removed.
14. Consider locking the pragma version in all the smart contracts.
15. To reduce gas consumption in loops, it is recommended to optimize them or potentially break them into smaller batches.

16. Consider adding in full NatSpec comments for all functions to have complete code documentation for future use.
17. Consider correcting the suggested comments in the StrikeStaking contract.

AUTOMATED TESTING

6.1 STATIC ANALYSIS REPORT

Description:

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

Slither results:

AggregatorV2V3Interface.sol

```
Pragma version>=0.5.0 (contracts/AggregatorV2V3Interface.sol#1) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

CarefulMath.sol

```
CarefulMath.addThenSubUInt(uint256,uint256,uint256) (contracts/CarefulMath.sol#76-84) is never used and should be removed
CarefulMath.addUInt(uint256,uint256) (contracts/CarefulMath.sol#63-71) is never used and should be removed
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) is never used and should be removed
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) is never used and should be removed
CarefulMath.subUInt(uint256,uint256) (contracts/CarefulMath.sol#52-58) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

Comptroller.sol

```
Comptroller (contracts/Comptroller.sol#17-1432) contract sets array length with a user-controlled value:
        - allMarkets.push(sToken(sToken)) (contracts/Comptroller.sol#1005)
Comptroller (contracts/Comptroller.sol#17-1432) contract sets array length with a user-controlled value:
        - accountAssets[borrower].push(sToken) (contracts/Comptroller.sol#176)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#array-length-assignment

Unitroller.fallback() (contracts/Unitroller.sol#135-147) uses delegatecall to a input-controlled function id
        - (success) = comptrollerImplementation.delegatecall(msg.data) (contracts/Unitroller.sol#137)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-delegatecall

Comptroller.stakeSTRKInternal(address,uint256) (contracts/Comptroller.sol#1321-1334) ignores return value by strk.transfer(strkStaking,amount) (contracts/Comptroller.sol#1326)
Comptroller.grantSTRKInternal(address,uint256) (contracts/Comptroller.sol#1343-1351) ignores return value by strk.transfer(user,amount) (contracts/Comptroller.sol#1347)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

UnitrollerAdminStorage.comptrollerImplementation (contracts/ComptrollerStorage.sol#20) is never initialized. It is used in:
        - Comptroller.adminOrInitializing() (contracts/Comptroller.sol#1128-1130)
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#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

Comptroller._grantSTRK(address,uint256) (contracts/Comptroller.sol#1402-1409) uses a dangerous strict equality:
        - require(bool,string)(amountLeft == 0,Insufficient STRK for grant) (contracts/Comptroller.sol#1406)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - a == 0 || b == 0 (contracts/Exponential.sol#306)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - require(bool,string)(c / a == b,errorMessage) (contracts/Exponential.sol#310)
STRK._writeCheckpoint(address,uint32,uint96,uint96) (contracts/Governance/STRK.sol#263-276) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (contracts/Governance/STRK.sol#266)
SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/SToken.sol#390)
SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - require(bool,string)(mathErr == MathError.NO_ERROR,could not calculate block delta) (contracts/SToken.sol#406)
SToken.balanceOfUnderlying(address) (contracts/SToken.sol#190-195) uses a dangerous strict equality:
        - require(bool,string)(mErr == MathError.NO_ERROR,balance could not be calculated) (contracts/SToken.sol#193)
SToken.borrowBalanceStored(address) (contracts/SToken.sol#271-275) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,borrowBalanceStored: borrowBalanceStoredInternal failed) (contracts/SToken.sol#273)
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) uses a dangerous strict equality:
        - b == 0 (contracts/CarefulMath.sol#42)
SToken.exchangeRateStored() (contracts/SToken.sol#328-332) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,exchangeRateStored: exchangeRateStoredInternal failed) (contracts/SToken.sol#330)
SToken.exchangeRateStoredInternal() (contracts/SToken.sol#339-369) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/SToken.sol#341)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/SToken.sol#33)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting comptroller failed) (contracts/SToken.sol#41)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting interest rate model failed) (contracts/SToken.sol#49)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == uint256(Error.NO_ERROR),LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/SToken.sol#997)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(seizeError == uint256(Error.NO_ERROR),token seizure failed) (contracts/SToken.sol#1011)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_EXCHANGE_CALCULATION_FAILED) (contracts/SToken.sol#536)
```

```
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_TOTAL_SUPPLY_CALCULATION_FAILED) (contracts/SToken.sol#544)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#547)
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) uses a dangerous strict equality:
        - assert(bool)(err2 == MathError.NO_ERROR) (contracts/Exponential.sol#163)
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) uses a dangerous strict equality:
        - a == 0 (contracts/CarefulMath.sol#25)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#899)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#902)
SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,exchange rate math error) (contracts/SToken.sol#1089)
SToken.transfer(address,uint256) (contracts/SToken.sol#135-137) uses a dangerous strict equality:
        - transferTokens(msg.sender,msg.sender,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#136)
SToken.transferFrom(address,address,uint256) (contracts/SToken.sol#146-148) uses a dangerous strict equality:
        - transferTokens(msg.sender,src,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#696)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

SToken.borrowFresh(address,uint256).vars (contracts/SToken.sol#753) is a local variable never initialized
Comptroller.borrowAllowed(address,address,uint256).err_scope_0 (contracts/Comptroller.sol#400) is a local variable never initialized
SToken.redeemFresh(address,uint256,uint256).vars (contracts/SToken.sol#616) is a local variable never initialized
SToken._addReservesFresh(uint256).actualAddAmount (contracts/SToken.sol#1274) is a local variable never initialized
SToken.seizeInternal(address,address,address,uint256).vars (contracts/SToken.sol#1073) is a local variable never initialized
SToken.mintFresh(address,uint256).vars (contracts/SToken.sol#509) is a local variable never initialized
SToken.repayBorrowFresh(address,address,uint256).vars (contracts/SToken.sol#862) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Comptroller._supportMarket(SToken) (contracts/Comptroller.sol#958-977) ignores return value by sToken.isSToken() (contracts/Comptroller.sol#967)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Comptroller.updateStrikeSupplyIndex(address).strikeAccrued (contracts/Comptroller.sol#1182) shadows:
        - ComptrollerV3Storage.strikeAccrued (contracts/ComptrollerStorage.sol#128) (state variable)
Comptroller.updateStrikeBorrowIndex(address,Exponential.Exp).strikeAccrued (contracts/Comptroller.sol#1202) shadows:
        - ComptrollerV3Storage.strikeAccrued (contracts/ComptrollerStorage.sol#128) (state variable)
Exponential.divScalarByExpTruncate(uint256,Exponential.Exp).fraction (contracts/Exponential.sol#135) shadows:
        - Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) should emit an event for:
        - initialExchangeRateMantissa = initialExchangeRateMantissa_ (contracts/SToken.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Comptroller._setPauseGuardian(address).newPauseGuardian (contracts/Comptroller.sol#1013) lacks a zero-check on :
                - pauseGuardian = newPauseGuardian (contracts/Comptroller.sol#1022)
Comptroller._setReserveInfo(address,address).newReserveGuardian (contracts/Comptroller.sol#1049) lacks a zero-check on :
                - reserveGuardian = newReserveGuardian (contracts/Comptroller.sol#1058)
Comptroller._setReserveInfo(address,address).newReserveAddress (contracts/Comptroller.sol#1049) lacks a zero-check on :
                - reserveAddress = newReserveAddress (contracts/Comptroller.sol#1059)
Comptroller._setStrkStakingInfo(address).newStrkStaking (contracts/Comptroller.sol#1067) lacks a zero-check on :
                - strkStaking = newStrkStaking (contracts/Comptroller.sol#1072)
Comptroller._setMarketCapGuardian(address).newMarketCapGuardian (contracts/Comptroller.sol#1084) lacks a zero-check on :
                - marketCapGuardian = newMarketCapGuardian (contracts/Comptroller.sol#1091)
SToken._setPendingAdmin(address).newPendingAdmin (contracts/SToken.sol#1135) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (contracts/SToken.sol#1145)
Unitroller._setPendingImplementation(address).newPendingImplementation (contracts/Unitroller.sol#38) lacks a zero-check on :
                - pendingComptrollerImplementation = newPendingImplementation (contracts/Unitroller.sol#46)
Unitroller._setPendingAdmin(address).newPendingAdmin (contracts/Unitroller.sol#85) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (contracts/Unitroller.sol#95)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Comptroller.updateStrikeSupplyIndex(address) (contracts/Comptroller.sol#1175-1189) has external calls inside a loop: supplyTokens = SToken(sToken).totalSupply() (contracts/Comptroller.sol#1181)
Comptroller.setStrikeSpeedInternal(SToken,uint256,uint256) (contracts/Comptroller.sol#1151-1169) has external calls inside a loop: borrowIndex = Exp(sToken.borrowIndex()) (contracts/Comptroller.sol#1162)
Comptroller.updateStrikeBorrowIndex(address,Exponential.Exp) (contracts/Comptroller.sol#1195-1209) has external calls inside a loop: borrowAmount = div_(SToken(sToken).totalBorrows(),marketBorrowIndex) (contracts/Comptroller.sol#1201)
Comptroller.claimStrike(address[],SToken[],bool,bool) (contracts/Comptroller.sol#1290-1312) has external calls inside a loop: borrowIndex = Exp(sToken.borrowIndex()) (contracts/Comptroller.sol#1295)
Comptroller.distributeBorrowerStrike(address,address,Exponential.Exp) (contracts/Comptroller.sol#1242-1262) has external calls inside a loop: borrowerAmount = div_(SToken(sToken).borrowBalanceStored(borrower),marketBorrowIndex) (contract
s/Comptroller.sol#1255)
Comptroller.distributeSupplierStrike(address,address) (contracts/Comptroller.sol#1216-1234) has external calls inside a loop: supplierTokens = SToken(sToken).balanceOf(supplier) (contracts/Comptroller.sol#1229)
Comptroller.stakeSTRKInternal(address,uint256) (contracts/Comptroller.sol#1321-1334) has external calls inside a loop: strkRemaining = strk.balanceOf(address(this)) (contracts/Comptroller.sol#1324)
Comptroller.stakeSTRKInternal(address,uint256) (contracts/Comptroller.sol#1321-1334) has external calls inside a loop: strk.transfer(strkStaking,amount) (contracts/Comptroller.sol#1326)
Comptroller.stakeSTRKInternal(address,uint256) (contracts/Comptroller.sol#1321-1334) has external calls inside a loop: IStrikeStaking(strkStaking).mint(user,amount) (contracts/Comptroller.sol#1327)
Comptroller.grantSTRKInternal(address,uint256) (contracts/Comptroller.sol#1343-1351) has external calls inside a loop: strikeRemaining = strk.balanceOf(address(this)) (contracts/Comptroller.sol#1345)
Comptroller.grantSTRKInternal(address,uint256) (contracts/Comptroller.sol#1343-1351) has external calls inside a loop: strk.transfer(user,amount) (contracts/Comptroller.sol#1347)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Variable 'Comptroller.borrowAllowed(address,address,uint256).err (contracts/Comptroller.sol#378)' in Comptroller.borrowAllowed(address,address,uint256) (contracts/Comptroller.sol#365-414) potentially used before declaration: (err,shortfa
ll) = getHypotheticalAccountLiquidityInternal(borrower,SToken(sToken),0,borrowAmount) (contracts/Comptroller.sol#400)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#788)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#789)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#790)
Reentrancy in SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/SToken.sol#499)
        State variables written after the call(s):
        - accountTokens[minter] = vars.accountTokensNew (contracts/SToken.sol#551)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#550)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - accountTokens[redeemer] = vars.accountTokensNew (contracts/SToken.sol#697)
Reentrancy in SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#905)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#906)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
Reentrancy in SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
        State variables written after the call(s):
        - accountTokens[borrower] = vars.borrowerTokensNew (contracts/SToken.sol#1112)
        - accountTokens[liquidator] = vars.liquidatorTokensNew (contracts/SToken.sol#1113)
        - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#1111)
Reentrancy in SToken.transferTokens(address,address,address,uint256) (contracts/SToken.sol#68-127):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/SToken.sol#70)
        State variables written after the call(s):
        - accountTokens[src] = srsTokensNew (contracts/SToken.sol#113)
        - accountTokens[dst] = dstTokensNew (contracts/SToken.sol#114)
        - transferAllowances[src][spender] = allowanceNew (contracts/SToken.sol#118)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Comptroller._grantSTRK(address,uint256) (contracts/Comptroller.sol#1402-1409):
        External calls:
        - amountLeft = grantSTRKInternal(recipient,amount) (contracts/Comptroller.sol#1405)
                - strk.transfer(user,amount) (contracts/Comptroller.sol#1347)
        Event emitted after the call(s):
        - StrikeGranted(recipient,amount) (contracts/Comptroller.sol#1408)
Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        Event emitted after the call(s):
```

```
        - Borrow(borrower,borrowAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#793)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
            - fail(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.BORROW_CASH_NOT_AVAILABLE) (contracts/SToken.sol#750)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
            - fail(Error.MARKET_NOT_FRESH,FailureInfo.BORROW_FRESHNESS_CHECK) (contracts/SToken.sol#745)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
            - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.BORROW_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#740)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
            - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#772)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
            - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#767)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
            - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#762)
    Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
            - fail(Error.INVALID_ACCOUNT_PAIR,FailureInfo.LIQUIDATE_LIQUIDATOR_IS_BORROWER),0) (contracts/SToken.sol#971)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
            - fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED,FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_ZERO),0) (contracts/SToken.sol#976)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
            - fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED,FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_UINT_MAX),0) (contracts/SToken.sol#981)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
            - fail(Error.MARKET_NOT_FRESH,FailureInfo.LIQUIDATE_FRESHNESS_CHECK),0) (contracts/SToken.sol#961)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
            - fail(Error.MARKET_NOT_FRESH,FailureInfo.LIQUIDATE_COLLATERAL_FRESHNESS_CHECK),0) (contracts/SToken.sol#966)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
            - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.LIQUIDATE_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#956)
    Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
            - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
            - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
            - fail(TokenErrorReporter.Error(repayBorrowError),FailureInfo.LIQUIDATE_REPAY_BORROW_FRESH_FAILED),0) (contracts/SToken.sol#988)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
            - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
            - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
            - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
    Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
            - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
            - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
            - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
            - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
            - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
            - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
            - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
            - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
            - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
    Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
            - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
            - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
            - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
            - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
        - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(sTokenCollateral),seizeTokens) (contracts/SToken.sol#1014)
    Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
            - fail(TokenErrorReporter.Error(error),FailureInfo.LIQUIDATE_ACCRUE_COLLATERAL_INTEREST_FAILED),0) (contracts/SToken.sol#936)
    Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
            - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
            - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
            - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
            - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
            - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
            - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
            - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(sTokenCollateral),seizeTokens) (contracts/SToken.sol#1014)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
    Reentrancy in SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/SToken.sol#499)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
            - failOpaque(Error.MATH_ERROR,FailureInfo.MINT_EXCHANGE_RATE_READ_FAILED,uint256(vars.mathErr)),0) (contracts/SToken.sol#513)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
            - fail(Error.MARKET_NOT_FRESH,FailureInfo.MINT_FRESHNESS_CHECK),0) (contracts/SToken.sol#506)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
            - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.MINT_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#501)
        - Mint(minter,vars.actualMintAmount,vars.mintTokens) (contracts/SToken.sol#554)
        - Transfer(address(this),minter,vars.mintTokens) (contracts/SToken.sol#555)
    Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
            - fail(Error.MARKET_NOT_FRESH,FailureInfo.REDEEM_FRESHNESS_CHECK) (contracts/SToken.sol#660)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
            - failOpaque(Error.MATH_ERROR,FailureInfo.REDEEM_NEW_TOTAL_SUPPLY_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#670)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
            - failOpaque(Error.MATH_ERROR,FailureInfo.REDEEM_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#675)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
            - fail(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.REDEEM_TRANSFER_OUT_NOT_POSSIBLE) (contracts/SToken.sol#680)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
            - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.REDEEM_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#655)
        - Redeem(redeemer,vars.redeemAmount,vars.redeemTokens) (contracts/SToken.sol#701)
        - Transfer(redeemer,address(this),vars.redeemTokens) (contracts/SToken.sol#700)
    Reentrancy in SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
            - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.REPAY_BORROW_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#854)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
            - fail(Error.MARKET_NOT_FRESH,FailureInfo.REPAY_BORROW_FRESHNESS_CHECK),0) (contracts/SToken.sol#859)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
            - failOpaque(Error.MATH_ERROR,FailureInfo.REPAY_BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)),0) (contracts/SToken.sol#870)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
    Reentrancy in SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
            - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.LIQUIDATE_SEIZE_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#1061)
```

```
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_DECREMENT_FAILED,uint256(vars.mathErr)) (contracts/STOken.sol#1093)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_ACCOUNT_PAIR,FailureInfo.LIQUIDATE_SEIZE_LIQUIDATOR_IS_BORROWER) (contracts/STOken.sol#1066)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_DECREMENT_FAILED,uint256(vars.mathErr)) (contracts/STOken.sol#1082)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_INCREMENT_FAILED,uint256(vars.mathErr)) (contracts/STOken.sol#1102)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/STOken.sol#1118)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/STOken.sol#1116)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/STOken.sol#1117)
Reentrancy in STOken.transferTokens(address,address,address,uint256) (contracts/STOken.sol#68-127):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/STOken.sol#70)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_TOO_MUCH) (contracts/STOken.sol#106)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_NOT_ENOUGH) (contracts/STOken.sol#101)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.BAD_INPUT,FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/STOken.sol#77)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/STOken.sol#96)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.TRANSFER_COMPTROLLER_REJECTION,allowed) (contracts/STOken.sol#72)
        - Transfer(src,dst,tokens) (contracts/STOken.sol#122)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

STRK.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (contracts/Governance/STRK.sol#161-170) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now <= expiry,Strk::delegateBySig: signature expired) (contracts/Governance/STRK.sol#168)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

STRK.getChainId() (contracts/Governance/STRK.sol#299-303) uses assembly
        - INLINE ASM (contracts/Governance/STRK.sol#301)
Unitroller.fallback() (contracts/Unitroller.sol#135-147) uses assembly
        - INLINE ASM (contracts/Unitroller.sol#139-146)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Comptroller.addToMarketInternal(STOken,address) (contracts/Comptroller.sol#157-181) compares to a boolean constant:
        -marketToJoin.accountMembership[borrower] == true (contracts/Comptroller.sol#165)
Comptroller._setBorrowPaused(STOken,bool) (contracts/Comptroller.sol#1039-1047) compares to a boolean constant:
        -require(bool,string)(msg.sender == admin || state == true,only admin can unpause) (contracts/Comptroller.sol#1042)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Exponential.addExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#48-52) is never used and should be removed
Exponential.add_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#242-244) is never used and should be removed
Exponential.divScalar(Exponential.Exp,uint256) (contracts/Exponential.sol#102-109) is never used and should be removed
Exponential.divScalarByExp(uint256,Exponential.Exp) (contracts/Exponential.sol#114-129) is never used and should be removed
Exponential.divScalarByExpTruncate(uint256,Exponential.Exp) (contracts/Exponential.sol#134-141) is never used and should be removed
Exponential.div_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#326-328) is never used and should be removed
Exponential.div_(Exponential.Double,uint256) (contracts/Exponential.sol#330-332) is never used and should be removed
Exponential.div_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#314-316) is never used and should be removed
Exponential.div_(Exponential.Exp,uint256) (contracts/Exponential.sol#318-320) is never used and should be removed
Exponential.div_(uint256,Exponential.Double) (contracts/Exponential.sol#334-336) is never used and should be removed
Exponential.greaterThanExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#221-223) is never used and should be removed
Exponential.isZeroExp(Exponential.Exp) (contracts/Exponential.sol#228-230) is never used and should be removed
Exponential.lessThanOrEqualExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#214-216) is never used and should be removed
Exponential.mul_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#289-291) is never used and should be removed
Exponential.mul_(Exponential.Double,uint256) (contracts/Exponential.sol#293-295) is never used and should be removed
Exponential.mul_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#277-279) is never used and should be removed
Exponential.mul_(Exponential.Exp,uint256) (contracts/Exponential.sol#281-283) is never used and should be removed
Exponential.subExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#57-61) is never used and should be removed
Exponential.sub_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#264-266) is never used and should be removed
Exponential.sub_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#260-262) is never used and should be removed
STOken._addReservesFresh(uint256) (contracts/STOken.sol#1271-1308) is never used and should be removed
STOken._addReservesInternal(uint256) (contracts/STOken.sol#1253-1263) is never used and should be removed
STOken.borrowFresh(address,uint256) (contracts/STOken.sol#736-799) is never used and should be removed
STOken.borrowInternal(uint256) (contracts/STOken.sol#714-722) is never used and should be removed
STOken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/STOken.sol#952-1020) is never used and should be removed
STOken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/STOken.sol#926-941) is never used and should be removed
STOken.mintFresh(address,uint256) (contracts/STOken.sol#497-561) is never used and should be removed
STOken.mintInternal(uint256) (contracts/STOken.sol#470-478) is never used and should be removed
STOken.redeemFresh(address,uint256,uint256) (contracts/STOken.sol#613-707) is never used and should be removed
STOken.redeemInternal(uint256) (contracts/STOken.sol#569-577) is never used and should be removed
STOken.redeemUnderlyingInternal(uint256) (contracts/STOken.sol#585-593) is never used and should be removed
STOken.repayBorrowBehalfInternal(address,uint256) (contracts/STOken.sol#822-830) is never used and should be removed
STOken.repayBorrowFresh(address,address,uint256) (contracts/STOken.sol#858-916) is never used and should be removed
STOken.repayBorrowInternal(uint256) (contracts/STOken.sol#806-814) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Low level call in Unitroller.fallback() (contracts/Unitroller.sol#135-147):
        - (success) = comptrollerImplementation.delegatecall(msg.data) (contracts/Unitroller.sol#137)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function Comptroller._setPriceOracle(PriceOracle) (contracts/Comptroller.sol#851-867) is not in mixedCase
Function Comptroller._setCloseFactor(uint256) (contracts/Comptroller.sol#875-884) is not in mixedCase
Function Comptroller._setCollateralFactor(STOken,uint256) (contracts/Comptroller.sol#893-926) is not in mixedCase
Function Comptroller._setLiquidationIncentive(uint256) (contracts/Comptroller.sol#934-950) is not in mixedCase
Function Comptroller._supportMarket(STOken) (contracts/Comptroller.sol#958-977) is not in mixedCase
Function Comptroller._setPauseGuardian(address) (contracts/Comptroller.sol#1013-1028) is not in mixedCase
Function Comptroller._setProtocolPaused(bool) (contracts/Comptroller.sol#1033-1037) is not in mixedCase
Function Comptroller._setBorrowPaused(STOken,bool) (contracts/Comptroller.sol#1039-1047) is not in mixedCase
Function Comptroller._setReserveInfo(address,address) (contracts/Comptroller.sol#1049-1065) is not in mixedCase
Function Comptroller._setSTRKStakingInfo(address) (contracts/Comptroller.sol#1067-1078) is not in mixedCase
Function Comptroller._setMarketCapGuardian(address) (contracts/Comptroller.sol#1084-1095) is not in mixedCase
Function Comptroller._setMarketCaps(STOken[],uint256[],uint256[]) (contracts/Comptroller.sol#1104-1118) is not in mixedCase
Function Comptroller._become(Unitroller) (contracts/Comptroller.sol#1120-1123) is not in mixedCase
Function Comptroller._setStrikeSpeeds(STOken[],uint256[],uint256[]) (contracts/Comptroller.sol#1140-1149) is not in mixedCase
Function Comptroller._setContributorStrikeSpeed(address,uint256) (contracts/Comptroller.sol#1378-1395) is not in mixedCase
Function Comptroller._grantSTRK(address,uint256) (contracts/Comptroller.sol#1402-1409) is not in mixedCase
Constant Comptroller.strikeInitialIndex (contracts/Comptroller.sol#91) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Comptroller.collateralFactorMaxMantissa (contracts/Comptroller.sol#94) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ComptrollerInterface.isComptroller (contracts/ComptrollerInterface.sol#5) is not in UPPER_CASE_WITH_UNDERSCORES
Variable ComptrollerV2Storage._mintGuardianPaused (contracts/ComptrollerStorage.sol#89) is not in mixedCase
Variable ComptrollerV2Storage._borrowGuardianPaused (contracts/ComptrollerStorage.sol#90) is not in mixedCase
Constant Exponential.expScale (contracts/Exponential.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.doubleScale (contracts/Exponential.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.halfExpScale (contracts/Exponential.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.mantissaOne (contracts/Exponential.sol#16) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STRK.totalSupply (contracts/Governance/STRK.sol#15) 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
Function STOken._setPendingAdmin(address) (contracts/STOken.sol#1135-1151) is not in mixedCase
Function STOken._acceptAdmin() (contracts/STOken.sol#1158-1178) is not in mixedCase
Function STOken._setComptroller(ComptrollerInterface) (contracts/STOken.sol#1185-1202) is not in mixedCase
Function STOken._setReserveFactor(uint256) (contracts/STOken.sol#1209-1217) is not in mixedCase
Function STOken._reduceReserves(uint256) (contracts/STOken.sol#1316-1324) is not in mixedCase
Function STOken._transferReserves(uint256) (contracts/STOken.sol#1380-1388) is not in mixedCase
Function STOken._setInterestRateModel(InterestRateModel) (contracts/STOken.sol#1445-1453) is not in mixedCase
Variable STokenStorage._notEntered (contracts/STokenInterfaces.sol#10) is not in mixedCase
Constant STokenStorage.borrowRateMaxMantissa (contracts/STokenInterfaces.sol#11) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.reserveFactorMaxMantissa (contracts/STokenInterfaces.sol#36) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.protocolSeizeShareMantissa (contracts/STokenInterfaces.sol#121) is not in UPPER_CASE_WITH_UNDERSCORES
Function STokenInterface._setPendingAdmin(address) (contracts/STokenInterfaces.sol#246) is not in mixedCase
Function STokenInterface._acceptAdmin() (contracts/STokenInterfaces.sol#247) is not in mixedCase
Function STokenInterface._setComptroller(ComptrollerInterface) (contracts/STokenInterfaces.sol#248) is not in mixedCase
Function STokenInterface._setReserveFactor(uint256) (contracts/STokenInterfaces.sol#249) is not in mixedCase
Function STokenInterface._reduceReserves(uint256) (contracts/STokenInterfaces.sol#250) is not in mixedCase
Function STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251) is not in mixedCase
Constant STokenInterface.isSToken (contracts/STokenInterfaces.sol#120) is not in UPPER_CASE_WITH_UNDERSCORES
Function SErc20Interface._addReserves(uint256) (contracts/STokenInterfaces.sol#276) is not in mixedCase
Function SDelegatorInterface._setImplementation(address,bool,bytes) (contracts/STokenInterfaces.sol#298) is not in mixedCase
Function SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307) is not in mixedCase
Function SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312) is not in mixedCase
Function Unitroller._setPendingImplementation(address) (contracts/Unitroller.sol#38-51) is not in mixedCase
Function Unitroller._acceptImplementation() (contracts/Unitroller.sol#58-76) is not in mixedCase
Function Unitroller._setPendingAdmin(address) (contracts/Unitroller.sol#85-101) is not in mixedCase
Function Unitroller._acceptAdmin() (contracts/Unitroller.sol#108-128) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "minter (contracts/Comptroller.sol#255)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "mintAmount (contracts/Comptroller.sol#256)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "sToken (contracts/Comptroller.sol#292)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "minter (contracts/Comptroller.sol#293)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "actualMintAmount (contracts/Comptroller.sol#294)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "mintTokens (contracts/Comptroller.sol#295)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "sToken (contracts/Comptroller.sol#349)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "redeemer (contracts/Comptroller.sol#350)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "sToken (contracts/Comptroller.sol#424)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "borrower (contracts/Comptroller.sol#425)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "borrowAmount (contracts/Comptroller.sol#426)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "payer (contracts/Comptroller.sol#443)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "borrower (contracts/Comptroller.sol#444)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "repayAmount (contracts/Comptroller.sol#445)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "sToken (contracts/Comptroller.sol#473)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "payer (contracts/Comptroller.sol#474)" inComptroller (contracts/Comptroller.sol#17-1432)
```

```
Redundant expression "borrower (contracts/Comptroller.sol#475)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "actualRepayAmount (contracts/Comptroller.sol#476)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "borrowerIndex (contracts/Comptroller.sol#477)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "liquidator (contracts/Comptroller.sol#495)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "sTokenBorrowed (contracts/Comptroller.sol#539)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "sTokenCollateral (contracts/Comptroller.sol#540)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "liquidator (contracts/Comptroller.sol#541)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "borrower (contracts/Comptroller.sol#542)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "actualRepayAmount (contracts/Comptroller.sol#543)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "seizeTokens (contracts/Comptroller.sol#544)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "seizeTokens (contracts/Comptroller.sol#564)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "sTokenCollateral (contracts/Comptroller.sol#597)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "sTokenBorrowed (contracts/Comptroller.sol#598)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "liquidator (contracts/Comptroller.sol#599)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "borrower (contracts/Comptroller.sol#600)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "seizeTokens (contracts/Comptroller.sol#601)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "sToken (contracts/Comptroller.sol#639)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "src (contracts/Comptroller.sol#640)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "dst (contracts/Comptroller.sol#641)" inComptroller (contracts/Comptroller.sol#17-1432)
Redundant expression "transferTokens (contracts/Comptroller.sol#642)" inComptroller (contracts/Comptroller.sol#17-1432)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable Comptroller._setLiquidationIncentive(uint256).newLiquidationIncentiveMantissa (contracts/Comptroller.sol#934) is too similar to Comptroller._setLiquidationIncentive(uint256).oldLiquidationIncentiveMantissa (contracts/Comptroller.sol#941)
Variable SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface).seizeTokens (contracts/SToken.sol#996) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable STokenInterface.seize(address,address,uint256).seizeTokens (contracts/STokenInterfaces.sol#241) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable SToken.seizeInternal(address,address,address,uint256).seizeTokens (contracts/SToken.sol#1057) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable SToken.seize(address,address,uint256).seizeTokens (contracts/SToken.sol#1031) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

SToken (contracts/SToken.sol#16-1525) does not implement functions:
        - SToken.doTransferIn(address,uint256) (contracts/SToken.sol#1504)
        - SToken.doTransferOut(address,uint256) (contracts/SToken.sol#1511)
        - SToken.getCashPrior() (contracts/SToken.sol#1498)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

Exponential.mantissaOne (contracts/Exponential.sol#16) is never used in Comptroller (contracts/Comptroller.sol#17-1432)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

ComptrollerV1Storage.maxAssets (contracts/ComptrollerStorage.sol#48) should be constant
ComptrollerV2Storage._borrowGuardianPaused (contracts/ComptrollerStorage.sol#90) should be constant
ComptrollerV2Storage._mintGuardianPaused (contracts/ComptrollerStorage.sol#89) should be constant
ComptrollerV2Storage.seizeGuardianPaused (contracts/ComptrollerStorage.sol#92) should be constant
ComptrollerV2Storage.transferGuardianPaused (contracts/ComptrollerStorage.sol#91) should be constant
ComptrollerV3Storage.strikeRate (contracts/ComptrollerStorage.sol#110) should be constant
SDelegationStorage.implementation (contracts/STokenInterfaces.sol#281) should be constant
SErc20Storage.underlying (contracts/STokenInterfaces.sol#258) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

enterMarkets(address[]) should be declared external:
        - Comptroller.enterMarkets(address[]) (contracts/Comptroller.sol#138-149)
getAccountLiquidity(address) should be declared external:
        - Comptroller.getAccountLiquidity(address) (contracts/Comptroller.sol#671-675)
getHypothetialAccountLiquidity(address,address,uint256,uint256) should be declared external:
        - Comptroller.getHypothetialAccountLiquidity(address,address,uint256,uint256) (contracts/Comptroller.sol#697-704)
_setPriceOracle(PriceOracle) should be declared external:
        - Comptroller._setPriceOracle(PriceOracle) (contracts/Comptroller.sol#851-867)
_setPauseGuardian(address) should be declared external:
        - Comptroller._setPauseGuardian(address) (contracts/Comptroller.sol#1013-1028)
_setProtocolPaused(bool) should be declared external:
        - Comptroller._setProtocolPaused(bool) (contracts/Comptroller.sol#1033-1037)
_setBorrowPaused(SToken,bool) should be declared external:
        - Comptroller._setBorrowPaused(SToken,bool) (contracts/Comptroller.sol#1039-1047)
_become(Unitroller) should be declared external:
        - Comptroller._become(Unitroller) (contracts/Comptroller.sol#1120-1123)
_setStrikeSpeeds(SToken[],uint256[],uint256[]) should be declared external:
        - Comptroller._setStrikeSpeeds(SToken[],uint256[],uint256[]) (contracts/Comptroller.sol#1140-1149)
claimStrike(address) should be declared external:
        - Comptroller.claimStrike(address) (contracts/Comptroller.sol#1268-1270)
_setContributorStrikeSpeed(address,uint256) should be declared external:
        - Comptroller._setContributorStrikeSpeed(address,uint256) (contracts/Comptroller.sol#1378-1395)
_grantSTRK(address,uint256) should be declared external:
        - Comptroller._grantSTRK(address,uint256) (contracts/Comptroller.sol#1402-1409)
getAllMarkets() should be declared external:
        - Comptroller.getAllMarkets() (contracts/Comptroller.sol#1416-1418)
delegate(address) should be declared external:
        - STRK.delegate(address) (contracts/Governance/STRK.sol#148-150)
delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - STRK.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (contracts/Governance/STRK.sol#161-170)
getPriorVotes(address,uint256) should be declared external:
        - STRK.getPriorVotes(address,uint256) (contracts/Governance/STRK.sol#189-221)
initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57)
_setInterestRateModel(InterestRateModel) should be declared external:
        - SToken._setInterestRateModel(InterestRateModel) (contracts/SToken.sol#1445-1453)
        - STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (contracts/STokenInterfaces.sol#298)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312)
_setPendingImplementation(address) should be declared external:
        - Unitroller._setPendingImplementation(address) (contracts/Unitroller.sol#38-51)
_acceptImplementation() should be declared external:
        - Unitroller._acceptImplementation() (contracts/Unitroller.sol#58-76)
_setPendingAdmin(address) should be declared external:
        - Unitroller._setPendingAdmin(address) (contracts/Unitroller.sol#85-101)
_acceptAdmin() should be declared external:
        - Unitroller._acceptAdmin() (contracts/Unitroller.sol#108-128)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

ComptrollerInterface.sol

```
Constant ComptrollerInterface.isComptroller (contracts/ComptrollerInterface.sol#5) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

ComptrollerStorage.sol

```
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#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/SToken.sol#390)
SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - require(bool,string)(mathErr == MathError.NO_ERROR,could not calculate block delta) (contracts/SToken.sol#406)
SToken.balanceOfUnderlying(address) (contracts/SToken.sol#190-195) uses a dangerous strict equality:
        - require(bool,string)(mErr == MathError.NO_ERROR,balance could not be calculated) (contracts/SToken.sol#193)
SToken.borrowBalanceStored(address) (contracts/SToken.sol#271-275) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,borrowBalanceStored: borrowBalanceStoredInternal failed) (contracts/SToken.sol#273)
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) uses a dangerous strict equality:
        - b == 0 (contracts/CarefulMath.sol#42)
SToken.exchangeRateStored() (contracts/SToken.sol#328-332) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,exchangeRateStored: exchangeRateStoredInternal failed) (contracts/SToken.sol#330)
SToken.exchangeRateStoredInternal() (contracts/SToken.sol#339-369) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/SToken.sol#341)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/SToken.sol#33)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting comptroller failed) (contracts/SToken.sol#41)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting interest rate model failed) (contracts/SToken.sol#49)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == uint256(Error.NO_ERROR),LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/SToken.sol#997)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(seizeError == uint256(Error.NO_ERROR),token seizure failed) (contracts/SToken.sol#1011)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_EXCHANGE_CALCULATION_FAILED) (contracts/SToken.sol#536)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_TOTAL_SUPPLY_CALCULATION_FAILED) (contracts/SToken.sol#544)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#547)
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) uses a dangerous strict equality:
        - assert(bool)(err2 == MathError.NO_ERROR) (contracts/Exponential.sol#163)
```

```
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) uses a dangerous strict equality:
        - a == 0 (contracts/CarefulMath.sol#25)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - a == 0 || b == 0 (contracts/Exponential.sol#306)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - require(bool,string)(c / a == b,errorMessage) (contracts/Exponential.sol#310)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#899)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#902)
SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,exchange rate math error) (contracts/SToken.sol#1089)
SToken.transfer(address,uint256) (contracts/SToken.sol#135-137) uses a dangerous strict equality:
        - transferTokens(msg.sender,msg.sender,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#136)
SToken.transferFrom(address,address,uint256) (contracts/SToken.sol#146-148) uses a dangerous strict equality:
        - transferTokens(msg.sender,src,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1087)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#696)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

SToken.repayBorrowFresh(address,address,uint256).vars (contracts/SToken.sol#862) is a local variable never initialized
SToken.seizeInternal(address,address,address,uint256).vars (contracts/SToken.sol#1073) is a local variable never initialized
SToken._addReservesFresh(uint256).actualAddAmount (contracts/SToken.sol#1274) is a local variable never initialized
SToken.borrowFresh(address,uint256).vars (contracts/SToken.sol#753) is a local variable never initialized
SToken.mintFresh(address,uint256).vars (contracts/SToken.sol#509) is a local variable never initialized
SToken.redeemFresh(address,uint256,uint256).vars (contracts/SToken.sol#616) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Exponential.divScalarByExpTruncate(uint256,Exponential.Exp).fraction (contracts/Exponential.sol#135) shadows:
        - Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) should emit an event for:
        - initialExchangeRateMantissa = initialExchangeRateMantissa_ (contracts/SToken.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

SToken._setPendingAdmin(address).newPendingAdmin (contracts/SToken.sol#1135) lacks a zero-check on :
        - pendingAdmin = newPendingAdmin (contracts/SToken.sol#1145)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#788)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#789)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#790)
Reentrancy in SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/SToken.sol#499)
        State variables written after the call(s):
        - accountTokens[minter] = vars.accountTokensNew (contracts/SToken.sol#551)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#550)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - accountTokens[redeemer] = vars.accountTokensNew (contracts/SToken.sol#697)
Reentrancy in SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#905)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#906)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
Reentrancy in SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
        State variables written after the call(s):
        - accountTokens[borrower] = vars.borrowerTokensNew (contracts/SToken.sol#1112)
        - accountTokens[liquidator] = vars.liquidatorTokensNew (contracts/SToken.sol#1113)
        - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#1111)
Reentrancy in SToken.transferTokens(address,address,address,uint256) (contracts/SToken.sol#68-127):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/SToken.sol#70)
        State variables written after the call(s):
        - accountTokens[src] = srcTokensNew (contracts/SToken.sol#113)
        - accountTokens[dst] = dstTokensNew (contracts/SToken.sol#114)
        - transferAllowances[src][spender] = allowanceNew (contracts/SToken.sol#118)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        Event emitted after the call(s):
        - Borrow(borrower,borrowAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#793)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#772)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.BORROW_CASH_NOT_AVAILABLE) (contracts/SToken.sol#750)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.COMPTROLLER_REJECTION,FailureInfo.BORROW_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#740)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#767)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.BORROW_FRESHNESS_CHECK) (contracts/SToken.sol#745)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#762)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED,FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_ZERO),0) (contracts/SToken.sol#976)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.LIQUIDATE_FRESHNESS_CHECK),0) (contracts/SToken.sol#961)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED,FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_UINT_MAX),0) (contracts/SToken.sol#981)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_ACCOUNT_PAIR,FailureInfo.LIQUIDATE_LIQUIDATOR_IS_BORROWER),0) (contracts/SToken.sol#971)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.COMPTROLLER_REJECTION,FailureInfo.LIQUIDATE_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#956)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.LIQUIDATE_COLLATERAL_FRESHNESS_CHECK),0) (contracts/SToken.sol#966)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(TokenErrorReporter.Error(repayBorrowError),FailureInfo.LIQUIDATE_REPAY_BORROW_FRESH_FAILED),0) (contracts/SToken.sol#988)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
```

```
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
        - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(sTokenCollateral),seizeTokens) (contracts/SToken.sol#1014)
Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - (fail(TokenErrorReporter.Error(error),FailureInfo.LIQUIDATE_ACCRUE_COLLATERAL_INTEREST_FAILED),0) (contracts/SToken.sol#936)
Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(sTokenCollateral),seizeTokens) (contracts/SToken.sol#1014)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
Reentrancy in SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/SToken.sol#499)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - (fail(Error.MARKET_NOT_FRESH,FailureInfo.MINT_FRESHNESS_CHECK),0) (contracts/SToken.sol#506)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - (failOpaque(Error.MATH_ERROR,FailureInfo.MINT_EXCHANGE_RATE_READ_FAILED,uint256(vars.mathErr)),0) (contracts/SToken.sol#513)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - (failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.MINT_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#501)
        - Mint(minter,vars.actualMintAmount,vars.mintTokens) (contracts/SToken.sol#554)
        - Transfer(address(this),minter,vars.mintTokens) (contracts/SToken.sol#555)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.REDEEM_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#675)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.REDEEM_NEW_TOTAL_SUPPLY_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#670)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.REDEEM_TRANSFER_OUT_NOT_POSSIBLE) (contracts/SToken.sol#680)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.REDEEM_FRESHNESS_CHECK) (contracts/SToken.sol#660)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.REDEEM_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#655)
        - Redeem(redeemer,vars.redeemAmount,vars.redeemTokens) (contracts/SToken.sol#701)
        - Transfer(redeemer,address(this),vars.redeemTokens) (contracts/SToken.sol#700)
Reentrancy in SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - (fail(Error.MARKET_NOT_FRESH,FailureInfo.REPAY_BORROW_FRESHNESS_CHECK),0) (contracts/SToken.sol#859)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - (failOpaque(Error.MATH_ERROR,FailureInfo.REPAY_BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)),0) (contracts/SToken.sol#870)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - (failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.REPAY_BORROW_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#854)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
Reentrancy in SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_DECREMENT_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#1082)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_ACCOUNT_PAIR,FailureInfo.LIQUIDATE_SEIZE_LIQUIDATOR_IS_BORROWER) (contracts/SToken.sol#1066)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.LIQUIDATE_SEIZE_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#1061)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_DECREMENT_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#1093)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_INCREMENT_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#1102)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
Reentrancy in SToken.transferTokens(address,address,address,uint256) (contracts/SToken.sol#968-127):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/SToken.sol#70)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.TRANSFER_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#72)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.BAD_INPUT,FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/SToken.sol#77)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/SToken.sol#96)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_TOO_MUCH) (contracts/SToken.sol#106)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_NOT_ENOUGH) (contracts/SToken.sol#101)
        - Transfer(src,dst,tokens) (contracts/SToken.sol#122)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) is never used and should be removed
ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) is never used and should be removed
Exponential.addExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#48-52) is never used and should be removed
Exponential.add_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#246-248) is never used and should be removed
Exponential.add_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#242-244) is never used and should be removed
Exponential.divExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#191-193) is never used and should be removed
Exponential.divScalar(Exponential.Exp,uint256) (contracts/Exponential.sol#102-109) is never used and should be removed
Exponential.divScalarByExp(uint256,Exponential.Exp) (contracts/Exponential.sol#114-129) is never used and should be removed
Exponential.divScalarByExpTruncate(uint256,Exponential.Exp) (contracts/Exponential.sol#134-141) is never used and should be removed
Exponential.div_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#326-328) is never used and should be removed
Exponential.div_(Exponential.Double,uint256) (contracts/Exponential.sol#330-332) is never used and should be removed
Exponential.div_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#314-316) is never used and should be removed
Exponential.div_(Exponential.Exp,uint256) (contracts/Exponential.sol#318-320) is never used and should be removed
Exponential.div_(uint256,Exponential.Double) (contracts/Exponential.sol#334-336) is never used and should be removed
Exponential.div_(uint256,Exponential.Exp) (contracts/Exponential.sol#322-324) is never used and should be removed
Exponential.div_(uint256,uint256) (contracts/Exponential.sol#338-340) is never used and should be removed
Exponential.div_(uint256,uint256,string) (contracts/Exponential.sol#342-345) is never used and should be removed
Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) is never used and should be removed
```

```
Exponential.greaterThanExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#221-223) is never used and should be removed
Exponential.isZeroExp(Exponential.Exp) (contracts/Exponential.sol#228-230) is never used and should be removed
Exponential.lessThanExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#207-209) is never used and should be removed
Exponential.lessThanOrEqualExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#214-216) is never used and should be removed
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) is never used and should be removed
Exponential.mulExp(uint256,uint256) (contracts/Exponential.sol#171-173) is never used and should be removed
Exponential.mulExp3(Exponential.Exp,Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#178-184) is never used and should be removed
Exponential.mul_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#289-291) is never used and should be removed
Exponential.mul_(Exponential.Double,uint256) (contracts/Exponential.sol#293-295) is never used and should be removed
Exponential.mul_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#277-279) is never used and should be removed
Exponential.mul_(Exponential.Exp,uint256) (contracts/Exponential.sol#281-283) is never used and should be removed
Exponential.mul_(uint256,Exponential.Double) (contracts/Exponential.sol#297-299) is never used and should be removed
Exponential.safe224(uint256,string) (contracts/Exponential.sol#232-235) is never used and should be removed
Exponential.safe32(uint256,string) (contracts/Exponential.sol#237-240) is never used and should be removed
Exponential.subExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#57-61) is never used and should be removed
Exponential.sub_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#264-266) is never used and should be removed
Exponential.sub_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#260-262) is never used and should be removed
SToken._addReservesFresh(uint256) (contracts/SToken.sol#1271-1308) is never used and should be removed
SToken._addReservesInternal(uint256) (contracts/SToken.sol#1253-1263) is never used and should be removed
SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799) is never used and should be removed
SToken.borrowInternal(uint256) (contracts/SToken.sol#714-722) is never used and should be removed
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) is never used and should be removed
SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941) is never used and should be removed
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) is never used and should be removed
SToken.mintInternal(uint256) (contracts/SToken.sol#470-478) is never used and should be removed
SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707) is never used and should be removed
SToken.redeemInternal(uint256) (contracts/SToken.sol#569-577) is never used and should be removed
SToken.redeemUnderlyingInternal(uint256) (contracts/SToken.sol#585-593) is never used and should be removed
SToken.repayBorrowBehalfInternal(address,uint256) (contracts/SToken.sol#822-830) is never used and should be removed
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) is never used and should be removed
SToken.repayBorrowInternal(uint256) (contracts/SToken.sol#806-814) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Constant ComptrollerInterface.isComptroller (contracts/ComptrollerInterface.sol#5) is not in UPPER_CASE_WITH_UNDERSCORES
Variable ComptrollerV2Storage._mintGuardianPaused (contracts/ComptrollerStorage.sol#89) is not in mixedCase
Variable ComptrollerV2Storage._borrowGuardianPaused (contracts/ComptrollerStorage.sol#90) is not in mixedCase
Constant Exponential.expScale (contracts/Exponential.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.doubleScale (contracts/Exponential.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.halfExpScale (contracts/Exponential.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.mantissaOne (contracts/Exponential.sol#16) 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
Function SToken._setPendingAdmin(address) (contracts/SToken.sol#1135-1151) is not in mixedCase
Function SToken._acceptAdmin() (contracts/SToken.sol#1158-1178) is not in mixedCase
Function SToken._setComptroller(ComptrollerInterface) (contracts/SToken.sol#1185-1202) is not in mixedCase
Function SToken._setReserveFactor(uint256) (contracts/SToken.sol#1209-1217) is not in mixedCase
Function SToken._reduceReserves(uint256) (contracts/SToken.sol#1316-1324) is not in mixedCase
Function SToken._transferReserves(uint256) (contracts/SToken.sol#1380-1388) is not in mixedCase
Function SToken._setInterestRateModel(InterestRateModel) (contracts/SToken.sol#1445-1453) is not in mixedCase
Variable SToken.Storage._notEntered (contracts/STokenInterfaces.sol#10) is not in mixedCase
Constant STokenStorage.borrowRateMaxMantissa (contracts/STokenInterfaces.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.reserveFactorMaxMantissa (contracts/STokenInterfaces.sol#36) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.protocolSeizeShareMantissa (contracts/STokenInterfaces.sol#121) is not in UPPER_CASE_WITH_UNDERSCORES
Function STokenInterface._setPendingAdmin(address) (contracts/STokenInterfaces.sol#246) is not in mixedCase
Function STokenInterface._acceptAdmin() (contracts/STokenInterfaces.sol#247) is not in mixedCase
Function STokenInterface._setComptroller(ComptrollerInterface) (contracts/STokenInterfaces.sol#248) is not in mixedCase
Function STokenInterface._setReserveFactor(uint256) (contracts/STokenInterfaces.sol#249) is not in mixedCase
Function STokenInterface._reduceReserves(uint256) (contracts/STokenInterfaces.sol#250) is not in mixedCase
Function STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251) is not in mixedCase
Constant STokenInterface.isSToken (contracts/STokenInterfaces.sol#129) is not in UPPER_CASE_WITH_UNDERSCORES
Function SErc20Interface._addReserves(uint256) (contracts/STokenInterfaces.sol#276) is not in mixedCase
Function SDelegatorInterface._setImplementation(address,bool,bytes) (contracts/STokenInterfaces.sol#298) is not in mixedCase
Function SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307) is not in mixedCase
Function SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface).seizeTokens (contracts/SToken.sol#996) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable STokenInterface.seize(address,address,uint256).seizeTokens (contracts/STokenInterfaces.sol#241) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable SToken.seize(address,address,uint256).seizeTokens (contracts/SToken.sol#1031) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable SToken.seizeInternal(address,address,address,uint256).seizeTokens (contracts/SToken.sol#1057) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

SToken (contracts/SToken.sol#16-1525) does not implement functions:
        - SToken.doTransferIn(address,uint256) (contracts/SToken.sol#1504)
        - SToken.doTransferOut(address,uint256) (contracts/SToken.sol#1511)
        - SToken.getCashPrior() (contracts/SToken.sol#1498)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

ComptrollerV1Storage.closeFactorMantissa (contracts/ComptrollerStorage.sol#38) should be constant
ComptrollerV1Storage.liquidationIncentiveMantissa (contracts/ComptrollerStorage.sol#43) should be constant
ComptrollerV1Storage.maxAssets (contracts/ComptrollerStorage.sol#48) should be constant
ComptrollerV2Storage._borrowGuardianPaused (contracts/ComptrollerStorage.sol#90) should be constant
ComptrollerV2Storage._mintGuardianPaused (contracts/ComptrollerStorage.sol#89) should be constant
ComptrollerV2Storage.pauseGuardian (contracts/ComptrollerStorage.sol#88) should be constant
ComptrollerV2Storage.seizeGuardianPaused (contracts/ComptrollerStorage.sol#92) should be constant
ComptrollerV2Storage.transferGuardianPaused (contracts/ComptrollerStorage.sol#91) should be constant
ComptrollerV3Storage.strikeRate (contracts/ComptrollerStorage.sol#110) should be constant
ComptrollerV6Storage.reserveAddress (contracts/ComptrollerStorage.sol#152) should be constant
ComptrollerV6Storage.reserveGuardian (contracts/ComptrollerStorage.sol#149) should be constant
ComptrollerV6Storage.xtr5staking (contracts/ComptrollerStorage.sol#155) should be constant
ComptrollerV7Storage.marketCapGuardian (contracts/ComptrollerStorage.sol#160) should be constant
ComptrollerV7Storage.protocolPaused (contracts/ComptrollerStorage.sol#171) should be constant
SDelegationStorage.implementation (contracts/STokenInterfaces.sol#283) should be constant
SErc20Storage.underlying (contracts/STokenInterfaces.sol#258) should be constant
UnitrollerAdminStorage.admin (contracts/ComptrollerStorage.sol#10) should be constant
UnitrollerAdminStorage.comptrollerImplementation (contracts/ComptrollerStorage.sol#20) should be constant
UnitrollerAdminStorage.pendingAdmin (contracts/ComptrollerStorage.sol#15) should be constant
UnitrollerAdminStorage.pendingComptrollerImplementation (contracts/ComptrollerStorage.sol#25) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57)
_setInterestRateModel(InterestRateModel) should be declared external:
        - SToken._setInterestRateModel(InterestRateModel) (contracts/SToken.sol#1445-1453)
        - STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (contracts/STokenInterfaces.sol#298)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

DAIInterestRateModelV3.sol

```
DAIInterestRateModelV3.dsrPerBlock() (contracts/DAIInterestRateModelV3.sol#82-87) performs a multiplication on the result of a division:
        -pot.dsr().sub(1e27).div(1e9).mul(15) (contracts/DAIInterestRateModelV3.sol#83-86)
DAIInterestRateModelV3.poke() (contracts/DAIInterestRateModelV3.sol#92-107) performs a multiplication on the result of a division:
        -stabilityFeePerBlock = duty.add(jug.base()).sub(1e27).mul(1e18).div(1e27).mul(15) (contracts/DAIInterestRateModelV3.sol#94)
JumpRateModelV2.getSupplyRate(uint256,uint256,uint256,uint256) (contracts/JumpRateModelV2.sol#116-121) performs a multiplication on the result of a division:
        -rateToPool = borrowRate.mul(oneMinusReserveFactor).div(1e18) (contracts/JumpRateModelV2.sol#119)
        -utilizationRate(cash,borrows,reserves).mul(rateToPool).div(1e18) (contracts/JumpRateModelV2.sol#120)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

DAIInterestRateModelV3.updateJumpRateModel(uint256,uint256,uint256,uint256) (contracts/DAIInterestRateModelV3.sol#51-56) should emit an event for:
        - gapPerBlock = gapPerYear / blocksPerYear (contracts/DAIInterestRateModelV3.sol#53)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

JumpRateModelV2.constructor(uint256,uint256,uint256,uint256,address).owner_ (contracts/JumpRateModelV2.sol#54) lacks a zero-check on :
        - owner = owner_ (contracts/JumpRateModelV2.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

SafeMath.add(uint256,uint256,string) (contracts/SafeMath.sol#43-48) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/SafeMath.sol#167-169) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/SafeMath.sol#182-185) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (contracts/SafeMath.sol#107-119) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Constant DAIInterestRateModelV3.assumedOneMinusReserveFactorMantissa (contracts/DAIInterestRateModelV3.sol#24) is not in UPPER_CASE_WITH_UNDERSCORES
Constant InterestRateModel.isInterestRateModel (contracts/InterestRateModel.sol#9) is not in UPPER_CASE_WITH_UNDERSCORES
Constant JumpRateModelV2.blocksPerYear (contracts/JumpRateModelV2.sol#24) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Juglike.base (contracts/DAIInterestRateModelV3.sol#131) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

EIP20Interface.sol

No issues were found by Slither.

EIP20NonStandardInterface.sol

```
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#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
```

ErrorReporter.sol

```
ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) is never used and should be removed
ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) is never used and should be removed
TokenErrorReporter.fail(TokenErrorReporter.Error,TokenErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#199-203) is never used and should be removed
TokenErrorReporter.failOpaque(TokenErrorReporter.Error,TokenErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#208-212) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

Exponential.sol

```
Exponential.divScalarByExpTruncate(uint256,Exponential.Exp).fraction (contracts/Exponential.sol#135) shadows:
        - Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

CarefulMath.addThenSubUInt(uint256,uint256,uint256) (contracts/CarefulMath.sol#76-84) is never used and should be removed
CarefulMath.addUInt(uint256,uint256) (contracts/CarefulMath.sol#63-71) is never used and should be removed
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) is never used and should be removed
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) is never used and should be removed
CarefulMath.subUInt(uint256,uint256) (contracts/CarefulMath.sol#52-58) is never used and should be removed
Exponential.addExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#48-52) is never used and should be removed
Exponential.add_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#206-208) is never used and should be removed
Exponential.add_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#242-244) is never used and should be removed
Exponential.add_(uint256,uint256) (contracts/Exponential.sol#250-252) is never used and should be removed
Exponential.add_(uint256,uint256,string) (contracts/Exponential.sol#254-258) is never used and should be removed
Exponential.divExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#191-193) is never used and should be removed
Exponential.divScalar(Exponential.Exp,uint256) (contracts/Exponential.sol#102-109) is never used and should be removed
Exponential.divScalarByExp(uint256,Exponential.Exp) (contracts/Exponential.sol#114-129) is never used and should be removed
Exponential.divScalarByExpTruncate(uint256,Exponential.Exp) (contracts/Exponential.sol#134-141) is never used and should be removed
Exponential.div_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#326-328) is never used and should be removed
Exponential.div_(Exponential.Double,uint256) (contracts/Exponential.sol#330-332) is never used and should be removed
Exponential.div_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#314-316) is never used and should be removed
Exponential.div_(Exponential.Exp,uint256) (contracts/Exponential.sol#318-320) is never used and should be removed
Exponential.div_(uint256,Exponential.Double) (contracts/Exponential.sol#334-336) is never used and should be removed
Exponential.div_(uint256,Exponential.Exp) (contracts/Exponential.sol#322-324) is never used and should be removed
Exponential.div_(uint256,uint256) (contracts/Exponential.sol#338-340) is never used and should be removed
Exponential.div_(uint256,uint256,string) (contracts/Exponential.sol#342-345) is never used and should be removed
Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) is never used and should be removed
Exponential.getExp(uint256,uint256) (contracts/Exponential.sol#31-41) is never used and should be removed
Exponential.greaterThanExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#221-223) is never used and should be removed
Exponential.isZeroExp(Exponential.Exp) (contracts/Exponential.sol#228-230) is never used and should be removed
Exponential.lessThanExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#207-209) is never used and should be removed
Exponential.lessThanOrEqualExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#214-216) is never used and should be removed
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) is never used and should be removed
Exponential.mulExp(uint256,uint256) (contracts/Exponential.sol#171-173) is never used and should be removed
Exponential.mulExp3(Exponential.Exp,Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#178-184) is never used and should be removed
Exponential.mulScalar(Exponential.Exp,uint256) (contracts/Exponential.sol#66-73) is never used and should be removed
Exponential.mulScalarTruncate(Exponential.Exp,uint256) (contracts/Exponential.sol#78-85) is never used and should be removed
Exponential.mulScalarTruncateAddUInt(Exponential.Exp,uint256,uint256) (contracts/Exponential.sol#90-97) is never used and should be removed
Exponential.mul_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#289-291) is never used and should be removed
Exponential.mul_(Exponential.Double,uint256) (contracts/Exponential.sol#293-295) is never used and should be removed
Exponential.mul_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#277-279) is never used and should be removed
Exponential.mul_(Exponential.Exp,uint256) (contracts/Exponential.sol#281-283) is never used and should be removed
Exponential.mul_(uint256,Exponential.Double) (contracts/Exponential.sol#297-299) is never used and should be removed
Exponential.mul_(uint256,Exponential.Exp) (contracts/Exponential.sol#285-287) is never used and should be removed
Exponential.mul_(uint256,uint256) (contracts/Exponential.sol#301-303) is never used and should be removed
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) is never used and should be removed
Exponential.safe224(uint256,string) (contracts/Exponential.sol#232-235) is never used and should be removed
Exponential.safe32(uint256,string) (contracts/Exponential.sol#237-240) is never used and should be removed
Exponential.subExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#57-61) is never used and should be removed
Exponential.sub_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#264-266) is never used and should be removed
Exponential.sub_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#260-262) is never used and should be removed
Exponential.sub_(uint256,uint256) (contracts/Exponential.sol#268-270) is never used and should be removed
Exponential.sub_(uint256,uint256,string) (contracts/Exponential.sol#272-275) is never used and should be removed
Exponential.truncate(Exponential.Exp) (contracts/Exponential.sol#199-202) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Constant Exponential.expScale (contracts/Exponential.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.doubleScale (contracts/Exponential.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.halfExpScale (contracts/Exponential.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.mantissaOne (contracts/Exponential.sol#16) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Exponential.mantissaOne (contracts/Exponential.sol#16) is never used in Exponential (contracts/Exponential.sol#12-350)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
```

InterestRateModel.sol

```
Constant InterestRateModel.isInterestRateModel (contracts/InterestRateModel.sol#9) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

JumpRateModel.sol

```
JumpRateModel.getSupplyRate(uint256,uint256,uint256,uint256) (contracts/JumpRateModel.sol#99-104) performs a multiplication on the result of a division:
        -rateToPool = borrowRate.mul(oneMinusReserveFactor).div(1e18) (contracts/JumpRateModel.sol#102)
        -utilizationRate(cash,borrows,reserves).mul(rateToPool).div(1e18) (contracts/JumpRateModel.sol#103)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

SafeMath.add(uint256,uint256,string) (contracts/SafeMath.sol#43-48) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/SafeMath.sol#167-169) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/SafeMath.sol#182-185) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (contracts/SafeMath.sol#107-119) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Constant InterestRateModel.isInterestRateModel (contracts/InterestRateModel.sol#9) is not in UPPER_CASE_WITH_UNDERSCORES
Constant JumpRateModel.blocksPerYear (contracts/JumpRateModel.sol#18) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

getSupplyRate(uint256,uint256,uint256,uint256) should be declared external:
        - JumpRateModel.getSupplyRate(uint256,uint256,uint256,uint256) (contracts/JumpRateModel.sol#99-104)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

JumpRateModelV2.sol

```
JumpRateModelV2.getSupplyRate(uint256,uint256,uint256,uint256) (contracts/JumpRateModelV2.sol#116-121) performs a multiplication on the result of a division:
        -rateToPool = borrowRate.mul(oneMinusReserveFactor).div(1e18) (contracts/JumpRateModelV2.sol#119)
        -utilizationRate(cash,borrows,reserves).mul(rateToPool).div(1e18) (contracts/JumpRateModelV2.sol#120)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

JumpRateModelV2.constructor(uint256,uint256,uint256,address).owner_ (contracts/JumpRateModelV2.sol#54) lacks a zero-check on :
                - owner = owner_ (contracts/JumpRateModelV2.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

SafeMath.add(uint256,uint256,string) (contracts/SafeMath.sol#43-48) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/SafeMath.sol#167-169) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/SafeMath.sol#182-185) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (contracts/SafeMath.sol#107-119) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Constant InterestRateModel.isInterestRateModel (contracts/InterestRateModel.sol#9) is not in UPPER_CASE_WITH_UNDERSCORES
Constant JumpRateModelV2.blocksPerYear (contracts/JumpRateModelV2.sol#24) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

getSupplyRate(uint256,uint256,uint256,uint256) should be declared external:
        - JumpRateModelV2.getSupplyRate(uint256,uint256,uint256,uint256) (contracts/JumpRateModelV2.sol#116-121)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Maximillion.sol

```
Maximillion.repayBehalfExplicit(address,SEther) (contracts/Maximillion.sol#37-46) sends eth to arbitrary user
        Dangerous calls:
        - sEther._repayBorrowBehalf.value(borrows)(borrower) (contracts/Maximillion.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

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#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/SToken.sol#390)
SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - require(bool,string)(mathErr == MathError.NO_ERROR,could not calculate block delta) (contracts/SToken.sol#406)
SToken.balanceOfUnderlying(address) (contracts/SToken.sol#190-195) uses a dangerous strict equality:
        - require(bool,string)(mErr == MathError.NO_ERROR,balance could not be calculated) (contracts/SToken.sol#193)
SToken.borrowBalanceStored(address) (contracts/SToken.sol#271-275) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,borrowBalanceStored: borrowBalanceStoredInternal failed) (contracts/SToken.sol#273)
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) uses a dangerous strict equality:
        - b == 0 (contracts/CarefulMath.sol#42)
SEther.doTransferIn(address,uint256) (contracts/SEther.sol#136-141) uses a dangerous strict equality:
        - require(bool,string)(msg.value == amount,value mismatch) (contracts/SEther.sol#139)
SToken.exchangeRateStored() (contracts/SToken.sol#328-332) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,exchangeRateStored: exchangeRateStoredInternal failed) (contracts/SToken.sol#330)
SToken.exchangeRateStoredInternal() (contracts/SToken.sol#339-369) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/SToken.sol#341)
SEther.getCashPrior() (contracts/SEther.sol#124-128) uses a dangerous strict equality:
        - require(bool)(err == MathError.NO_ERROR) (contracts/SEther.sol#126)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/SToken.sol#33)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting comptroller failed) (contracts/SToken.sol#41)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting interest rate model failed) (contracts/SToken.sol#49)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == uint256(Error.NO_ERROR),LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/SToken.sol#997)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(seizeError == uint256(Error.NO_ERROR),token seizure failed) (contracts/SToken.sol#1011)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_EXCHANGE_CALCULATION_FAILED) (contracts/SToken.sol#536)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_TOTAL_SUPPLY_CALCULATION_FAILED) (contracts/SToken.sol#544)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#547)
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) uses a dangerous strict equality:
        - assert(bool)(err2 == MathError.NO_ERROR) (contracts/Exponential.sol#163)
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) uses a dangerous strict equality:
        - a == 0 (contracts/CarefulMath.sol#25)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - a == 0 || b == 0 (contracts/Exponential.sol#306)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - require(bool,string)(c / a == b,errorMessage) (contracts/Exponential.sol#310)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#899)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#902)
SEther.requireNoError(uint256,string) (contracts/SEther.sol#148-167) uses a dangerous strict equality:
        - errCode == uint256(Error.NO_ERROR) (contracts/SEther.sol#149)
SEther.requireNoError(uint256,string) (contracts/SEther.sol#148-167) uses a dangerous strict equality:
        - require(bool,string)(errCode == uint256(Error.NO_ERROR),string(fullMessage)) (contracts/SEther.sol#166)
SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,exchange rate math error) (contracts/SToken.sol#1089)
SToken.transfer(address,uint256) (contracts/SToken.sol#135-137) uses a dangerous strict equality:
        - transferTokens(msg.sender,msg.sender,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#136)
SToken.transferFrom(address,address,uint256) (contracts/SToken.sol#146-148) uses a dangerous strict equality:
        - transferTokens(msg.sender,src,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1087)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#696)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

SToken._addReservesFresh(uint256).actualAddAmount (contracts/SToken.sol#1274) is a local variable never initialized
SToken.borrowFresh(address,uint256).vars (contracts/SToken.sol#753) is a local variable never initialized
SToken.redeemFresh(address,uint256,uint256).vars (contracts/SToken.sol#616) is a local variable never initialized
SToken.repayBorrowFresh(address,address,uint256).vars (contracts/SToken.sol#862) is a local variable never initialized
SToken.mintFresh(address,uint256).vars (contracts/SToken.sol#509) is a local variable never initialized
SToken.seizeInternal(address,address,address,uint256).vars (contracts/SToken.sol#1073) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Exponential.divScalarByExpTruncate(uint256,Exponential.Exp).fraction (contracts/Exponential.sol#135) shadows:
        - Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

SToken._setPendingAdmin(address).newPendingAdmin (contracts/SToken.sol#1135) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (contracts/SToken.sol#1145)
SEther.constructor(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8,address).admin_ (contracts/SEther.sol#27) lacks a zero-check on :
                - admin = admin_ (contracts/SEther.sol#34)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#788)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#789)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#790)
Reentrancy in SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/SToken.sol#499)
        State variables written after the call(s):
        - accountTokens[minter] = vars.accountTokensNew (contracts/SToken.sol#551)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#550)
```

PriceOracle.sol

```
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#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/SToken.sol#390)
SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - require(bool,string)(mathErr == MathError.NO_ERROR,could not calculate block delta) (contracts/SToken.sol#406)
SToken.balanceOfUnderlying(address) (contracts/SToken.sol#190-195) uses a dangerous strict equality:
        - require(bool,string)(mErr == MathError.NO_ERROR,balance could not be calculated) (contracts/SToken.sol#193)
SToken.borrowBalanceStored(address) (contracts/SToken.sol#271-275) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,borrowBalanceStored: borrowBalanceStoredInternal failed) (contracts/SToken.sol#273)
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) uses a dangerous strict equality:
        - b == 0 (contracts/CarefulMath.sol#42)
SToken.exchangeRateStored() (contracts/SToken.sol#328-332) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,exchangeRateStored: exchangeRateStoredInternal failed) (contracts/SToken.sol#330)
SToken.exchangeRateStoredInternal() (contracts/SToken.sol#339-369) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/SToken.sol#341)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/SToken.sol#33)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting comptroller failed) (contracts/SToken.sol#41)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting interest rate model failed) (contracts/SToken.sol#49)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == uint256(Error.NO_ERROR),LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/SToken.sol#997)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(seizeError == uint256(Error.NO_ERROR),token seizure failed) (contracts/SToken.sol#1011)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_EXCHANGE_CALCULATION_FAILED) (contracts/SToken.sol#536)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_TOTAL_SUPPLY_CALCULATION_FAILED) (contracts/SToken.sol#544)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#547)
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) uses a dangerous strict equality:
        - assert(bool)(err2 == MathError.NO_ERROR) (contracts/Exponential.sol#163)
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) uses a dangerous strict equality:
        - a == 0 (contracts/CarefulMath.sol#25)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - a == 0 || b == 0 (contracts/Exponential.sol#306)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - require(bool,string)(c / a == b,errorMessage) (contracts/Exponential.sol#310)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#899)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#902)
SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,exchange rate math error) (contracts/SToken.sol#1089)
SToken.transfer(address,uint256) (contracts/SToken.sol#135-137) uses a dangerous strict equality:
        - transferTokens(msg.sender,msg.sender,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#136)
SToken.transferFrom(address,address,uint256) (contracts/SToken.sol#146-148) uses a dangerous strict equality:
        - transferTokens(msg.sender,src,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#696)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

SToken.repayBorrowFresh(address,address,uint256).vars (contracts/SToken.sol#862) is a local variable never initialized
SToken.redeemFresh(address,uint256,uint256).vars (contracts/SToken.sol#616) is a local variable never initialized
SToken.mintFresh(address,uint256).vars (contracts/SToken.sol#509) is a local variable never initialized
SToken.borrowFresh(address,uint256).vars (contracts/SToken.sol#753) is a local variable never initialized
SToken._addReservesFresh(uint256).actualAddAmount (contracts/SToken.sol#1274) is a local variable never initialized
SToken.seizeInternal(address,address,address,uint256).vars (contracts/SToken.sol#1073) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Exponential.divScalarByExpTruncate(uint256,Exponential.Exp).fraction (contracts/Exponential.sol#135) shadows:
        - Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) should emit an event for:
        - initialExchangeRateMantissa = initialExchangeRateMantissa_ (contracts/SToken.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

SToken._setPendingAdmin(address).newPendingAdmin (contracts/SToken.sol#1135) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (contracts/SToken.sol#1145)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#788)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#789)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#790)
Reentrancy in SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/SToken.sol#499)
        State variables written after the call(s):
        - accountTokens[minter] = vars.accountTokensNew (contracts/SToken.sol#551)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#550)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - accountTokens[redeemer] = vars.accountTokensNew (contracts/SToken.sol#697)
Reentrancy in SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#905)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#906)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
Reentrancy in SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
        State variables written after the call(s):
        - accountTokens[borrower] = vars.borrowerTokensNew (contracts/SToken.sol#1112)
        - accountTokens[liquidator] = vars.liquidatorTokensNew (contracts/SToken.sol#1113)
        - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#1111)
Reentrancy in SToken.transferTokens(address,address,address,uint256) (contracts/SToken.sol#68-127):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/SToken.sol#70)
        State variables written after the call(s):
        - accountTokens[src] = srsTokensNew (contracts/SToken.sol#113)
        - accountTokens[dst] = dstTokensNew (contracts/SToken.sol#114)
        - transferAllowances[src][spender] = allowanceNew (contracts/SToken.sol#118)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        Event emitted after the call(s):
        - Borrow(borrower,borrowAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#793)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#772)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.BORROW_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#740)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.BORROW_CASH_NOT_AVAILABLE) (contracts/SToken.sol#750)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.BORROW_FRESHNESS_CHECK) (contracts/SToken.sol#745)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#767)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#762)
```

```
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.LIQUIDATE_FRESHNESS_CHECK),0) (contracts/SToken.sol#961)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.LIQUIDATE_COLLATERAL_FRESHNESS_CHECK),0) (contracts/SToken.sol#966)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED,FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_ZERO),0) (contracts/SToken.sol#976)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_ACCOUNT_PAIR,FailureInfo.LIQUIDATE_LIQUIDATOR_IS_BORROWER),0) (contracts/SToken.sol#971)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED,FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_UINT_MAX),0) (contracts/SToken.sol#981)
        - Failure(uint(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.COMPTROLLER_REJECTION,FailureInfo.LIQUIDATE_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#956)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(TokenErrorReporter.Error(repayBorrowError),FailureInfo.LIQUIDATE_REPAY_BORROW_FRESH_FAILED),0) (contracts/SToken.sol#988)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
        - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(sTokenCollateral),seizeTokens) (contracts/SToken.sol#1014)
Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(TokenErrorReporter.Error(error),FailureInfo.LIQUIDATE_ACCRUE_COLLATERAL_INTEREST_FAILED),0) (contracts/SToken.sol#936)
Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(sTokenCollateral),seizeTokens) (contracts/SToken.sol#1014)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
Reentrancy in SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/SToken.sol#499)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.MINT_FRESHNESS_CHECK),0) (contracts/SToken.sol#506)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.MINT_EXCHANGE_RATE_READ_FAILED,uint256(vars.mathErr)),0) (contracts/SToken.sol#513)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.COMPTROLLER_REJECTION,FailureInfo.MINT_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#501)
        - Mint(minter,vars.actualMintAmount,vars.mintTokens) (contracts/SToken.sol#554)
        - Transfer(address(this),minter,vars.mintTokens) (contracts/SToken.sol#555)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.REDEEM_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#675)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.REDEEM_TRANSFER_OUT_NOT_POSSIBLE) (contracts/SToken.sol#680)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.REDEEM_FRESHNESS_CHECK) (contracts/SToken.sol#660)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.REDEEM_NEW_TOTAL_SUPPLY_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#670)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.COMPTROLLER_REJECTION,FailureInfo.REDEEM_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#655)
        - Redeem(redeemer,vars.redeemAmount,vars.redeemTokens) (contracts/SToken.sol#701)
        - Transfer(redeemer,address(this),vars.redeemTokens) (contracts/SToken.sol#700)
Reentrancy in SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.REPAY_BORROW_FRESHNESS_CHECK),0) (contracts/SToken.sol#859)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.COMPTROLLER_REJECTION,FailureInfo.REPAY_BORROW_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#854)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.REPAY_BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)),0) (contracts/SToken.sol#870)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
Reentrancy in SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_DECREMENT_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#1093)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_ACCOUNT_PAIR,FailureInfo.LIQUIDATE_SEIZE_LIQUIDATOR_IS_BORROWER) (contracts/SToken.sol#1066)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.COMPTROLLER_REJECTION,FailureInfo.LIQUIDATE_SEIZE_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#1061)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_DECREMENT_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#1082)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_INCREMENT_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#1102)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
Reentrancy in SToken.transferTokens(address,address,address,uint256) (contracts/SToken.sol#68-127):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/SToken.sol#70)
        Event emitted after the call(s):
```

```
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.BAD_INPUT,FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/SToken.sol#77)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/SToken.sol#96)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_NOT_ENOUGH) (contracts/SToken.sol#101)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_TOO_MUCH) (contracts/SToken.sol#106)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.TRANSFER_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#72)
        - Transfer(src,dst,tokens) (contracts/SToken.sol#122)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) is never used and should be removed
ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) is never used and should be removed
Exponential.addExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#48-52) is never used and should be removed
Exponential.add_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#246-248) is never used and should be removed
Exponential.add_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#242-244) is never used and should be removed
Exponential.divExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#191-193) is never used and should be removed
Exponential.divScalar(Exponential.Exp,uint256) (contracts/Exponential.sol#102-109) is never used and should be removed
Exponential.divScalarByExp(uint256,Exponential.Exp) (contracts/Exponential.sol#114-129) is never used and should be removed
Exponential.divScalarByExpTruncate(uint256,Exponential.Exp) (contracts/Exponential.sol#134-141) is never used and should be removed
Exponential.div_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#326-328) is never used and should be removed
Exponential.div_(Exponential.Double,uint256) (contracts/Exponential.sol#330-332) is never used and should be removed
Exponential.div_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#314-316) is never used and should be removed
Exponential.div_(Exponential.Exp,uint256) (contracts/Exponential.sol#318-320) is never used and should be removed
Exponential.div_(uint256,Exponential.Double) (contracts/Exponential.sol#334-336) is never used and should be removed
Exponential.div_(uint256,Exponential.Exp) (contracts/Exponential.sol#322-324) is never used and should be removed
Exponential.div_(uint256,uint256) (contracts/Exponential.sol#338-340) is never used and should be removed
Exponential.div_(uint256,uint256,string) (contracts/Exponential.sol#342-345) is never used and should be removed
Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) is never used and should be removed
Exponential.greaterThanExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#221-223) is never used and should be removed
Exponential.isZeroExp(Exponential.Exp) (contracts/Exponential.sol#228-230) is never used and should be removed
Exponential.lessThanExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#207-209) is never used and should be removed
Exponential.lessThanOrEqualExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#214-216) is never used and should be removed
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) is never used and should be removed
Exponential.mulExp(uint256,uint256) (contracts/Exponential.sol#171-173) is never used and should be removed
Exponential.mulExp3(Exponential.Exp,Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#178-184) is never used and should be removed
Exponential.mul_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#289-291) is never used and should be removed
Exponential.mul_(Exponential.Double,uint256) (contracts/Exponential.sol#293-295) is never used and should be removed
Exponential.mul_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#277-279) is never used and should be removed
Exponential.mul_(Exponential.Exp,uint256) (contracts/Exponential.sol#281-283) is never used and should be removed
Exponential.mul_(uint256,Exponential.Double) (contracts/Exponential.sol#297-299) is never used and should be removed
Exponential.safe224(uint256,string) (contracts/Exponential.sol#232-235) is never used and should be removed
Exponential.safe32(uint256,string) (contracts/Exponential.sol#237-240) is never used and should be removed
Exponential.subExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#57-61) is never used and should be removed
Exponential.sub_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#264-266) is never used and should be removed
Exponential.sub_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#260-262) is never used and should be removed
SToken._addReservesFresh(uint256) (contracts/SToken.sol#1271-1308) is never used and should be removed
SToken._addReservesInternal(uint256) (contracts/SToken.sol#1253-1263) is never used and should be removed
SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799) is never used and should be removed
SToken.borrowInternal(uint256) (contracts/SToken.sol#714-722) is never used and should be removed
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) is never used and should be removed
SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941) is never used and should be removed
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) is never used and should be removed
SToken.mintInternal(uint256) (contracts/SToken.sol#470-478) is never used and should be removed
SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707) is never used and should be removed
SToken.redeemInternal(uint256) (contracts/SToken.sol#569-577) is never used and should be removed
SToken.redeemUnderlyingInternal(uint256) (contracts/SToken.sol#585-593) is never used and should be removed
SToken.repayBorrowBehalfInternal(address,uint256) (contracts/SToken.sol#822-830) is never used and should be removed
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) is never used and should be removed
SToken.repayBorrowInternal(uint256) (contracts/SToken.sol#806-814) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Constant ComptrollerInterface.isComptroller (contracts/ComptrollerInterface.sol#5) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.expScale (contracts/Exponential.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.doubleScale (contracts/Exponential.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.halfExpScale (contracts/Exponential.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.mantissaOne (contracts/Exponential.sol#16) 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
Function SToken._setPendingAdmin(address) (contracts/SToken.sol#1135-1151) is not in mixedCase
Function SToken._acceptAdmin() (contracts/SToken.sol#1158-1178) is not in mixedCase
Function SToken._setComptroller(ComptrollerInterface) (contracts/SToken.sol#1185-1202) is not in mixedCase
Function SToken._setReserveFactor(uint256) (contracts/SToken.sol#1209-1217) is not in mixedCase
Function SToken._reduceReserves(uint256) (contracts/SToken.sol#1316-1324) is not in mixedCase
Function SToken._transferReserves(uint256) (contracts/SToken.sol#1308-1308) is not in mixedCase
Function SToken._setInterestRateModel(InterestRateModel) (contracts/SToken.sol#1445-1453) is not in mixedCase
Variable STokenStorage._notEntered (contracts/STokenInterfaces.sol#10) is not in mixedCase
Constant STokenStorage.borrowRateMaxMantissa (contracts/STokenInterfaces.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.reserveFactorMaxMantissa (contracts/STokenInterfaces.sol#36) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.protocolSeizeShareMantissa (contracts/STokenInterfaces.sol#121) is not in UPPER_CASE_WITH_UNDERSCORES
Function STokenInterface._setPendingAdmin(address) (contracts/STokenInterfaces.sol#246) is not in mixedCase
Function STokenInterface._acceptAdmin() (contracts/STokenInterfaces.sol#247) is not in mixedCase
Function STokenInterface._setComptroller(ComptrollerInterface) (contracts/STokenInterfaces.sol#248) is not in mixedCase
Function STokenInterface._setReserveFactor(uint256) (contracts/STokenInterfaces.sol#249) is not in mixedCase
Function STokenInterface._reduceReserves(uint256) (contracts/STokenInterfaces.sol#250) is not in mixedCase
Function STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251) is not in mixedCase
Constant STokenInterface.isSToken (contracts/STokenInterfaces.sol#129) is not in UPPER_CASE_WITH_UNDERSCORES
Function SErc20Interface._addReserves(uint256) (contracts/STokenInterfaces.sol#276) is not in mixedCase
Function SDelegatorInterface._setImplementation(address,bool,bytes) (contracts/STokenInterfaces.sol#298) is not in mixedCase
Function SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307) is not in mixedCase
Function SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable SToken.seizeInternal(address,address,address,uint256).seizeTokens (contracts/SToken.sol#1057) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface).seizeTokens (contracts/SToken.sol#996) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable SToken.seize(address,address,uint256).seizeTokens (contracts/SToken.sol#1031) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable STokenInterface.seize(address,address,uint256).seizeTokens (contracts/STokenInterfaces.sol#241) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

SToken (contracts/SToken.sol#16-1525) does not implement functions:
        - SToken.doTransferIn(address,uint256) (contracts/SToken.sol#1504)
        - SToken.doTransferOut(address,uint256) (contracts/SToken.sol#1511)
        - SToken.getCashPrior() (contracts/SToken.sol#1498)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

SDelegationStorage.implementation (contracts/STokenInterfaces.sol#283) should be constant
SErc20Storage.underlying (contracts/STokenInterfaces.sol#258) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57)
_setInterestRateModel(InterestRateModel) should be declared external:
        - SToken._setInterestRateModel(InterestRateModel) (contracts/SToken.sol#1445-1453)
        - STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (contracts/STokenInterfaces.sol#298)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Reservoir.sol

```
Reservoir.drip() (contracts/Reservoir.sol#45-66) ignores return value by token_.transfer(target_,toDrip_) (contracts/Reservoir.sol#63)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

Reservoir.mul(uint256,uint256,string) (contracts/Reservoir.sol#82-89) uses a dangerous strict equality:
        - require(bool,string)(c / a == b,errorMessage) (contracts/Reservoir.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reservoir.constructor(uint256,EIP20Interface,address).target_ (contracts/Reservoir.sol#32) lacks a zero-check on :
        - target = target_ (contracts/Reservoir.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

drip() should be declared external:
        - Reservoir.drip() (contracts/Reservoir.sol#45-66)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

SafeMath.sol

```
SafeMath.add(uint256,uint256) (contracts/SafeMath.sol#28-33) is never used and should be removed
SafeMath.add(uint256,uint256,string) (contracts/SafeMath.sol#43-48) is never used and should be removed
SafeMath.div(uint256,uint256) (contracts/SafeMath.sol#132-134) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/SafeMath.sol#147-154) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/SafeMath.sol#167-169) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/SafeMath.sol#182-185) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/SafeMath.sol#85-97) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (contracts/SafeMath.sol#107-119) is never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/SafeMath.sol#58-60) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (contracts/SafeMath.sol#70-75) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

SDaiDelegate.sol

```
SErc20Delegate._becomeImplementation(bytes) (contracts/SErc20Delegate.sol#20-30) uses a Boolean constant improperly:
        -false (contracts/SErc20Delegate.sol#25)
SErc20Delegate._resignImplementation() (contracts/SErc20Delegate.sol#35-42) uses a Boolean constant improperly:
        -false (contracts/SErc20Delegate.sol#37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#misuse-of-a-boolean-constant

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#48)
GemLike (contracts/SDaiDelegate.sol#192-196) has incorrect ERC20 function interface:GemLike.approve(address,uint256) (contracts/SDaiDelegate.sol#193)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/SToken.sol#390)
SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - require(bool,string)(mathErr == MathError.NO_ERROR,could not calculate block delta) (contracts/SToken.sol#406)
SToken.balanceOfUnderlying(address) (contracts/SToken.sol#190-195) uses a dangerous strict equality:
        - require(bool,string)(mErr == MathError.NO_ERROR,balance could not be calculated) (contracts/SToken.sol#193)
SToken.borrowBalanceStored(address) (contracts/SToken.sol#271-276) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,borrowBalanceStored: borrowBalanceStoredInternal failed) (contracts/SToken.sol#273)
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) uses a dangerous strict equality:
        - b == 0 (contracts/CarefulMath.sol#42)
SToken.exchangeRateStored() (contracts/SToken.sol#328-332) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,exchangeRateStored: exchangeRateStoredInternal failed) (contracts/SToken.sol#330)
SToken.exchangeRateStoredInternal() (contracts/SToken.sol#339-369) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/SToken.sol#341)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/SToken.sol#33)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting comptroller failed) (contracts/SToken.sol#41)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting interest rate model failed) (contracts/SToken.sol#49)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == uint256(Error.NO_ERROR),LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/SToken.sol#997)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(seizeError == uint256(Error.NO_ERROR),token seizure failed) (contracts/SToken.sol#1011)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_EXCHANGE_CALCULATION_FAILED) (contracts/SToken.sol#536)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_TOTAL_SUPPLY_CALCULATION_FAILED) (contracts/SToken.sol#544)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#547)
SDaiDelegate.mul(uint256,uint256) (contracts/SDaiDelegate.sol#177-179) uses a dangerous strict equality:
        - require(bool,string)(y == 0 || (z = x * y) / y == x,mul-overflow) (contracts/SDaiDelegate.sol#178)
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) uses a dangerous strict equality:
        - assert(bool)(err2 == MathError.NO_ERROR) (contracts/Exponential.sol#163)
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) uses a dangerous strict equality:
        - a == 0 (contracts/CarefulMath.sol#25)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - a == 0 || b == 0 (contracts/Exponential.sol#306)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - require(bool,string)(c / a == b,errorMessage) (contracts/Exponential.sol#310)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#899)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#902)
SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,exchange rate math error) (contracts/SToken.sol#1089)
SToken.transfer(address,uint256) (contracts/SToken.sol#135-137) uses a dangerous strict equality:
        - transferTokens(msg.sender,msg.sender,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#136)
SToken.transferFrom(address,address,uint256) (contracts/SToken.sol#146-148) uses a dangerous strict equality:
        - transferTokens(msg.sender,src,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#696)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

SToken.repayBorrowFresh(address,address,uint256).vars (contracts/SToken.sol#862) is a local variable never initialized
SToken.seizeInternal(address,address,address,uint256).vars (contracts/SToken.sol#1073) is a local variable never initialized
SToken._addReservesFresh(uint256).actualAddAmount (contracts/SToken.sol#1274) is a local variable never initialized
SToken.borrowFresh(address,uint256).vars (contracts/SToken.sol#753) is a local variable never initialized
SToken.mintFresh(address,uint256).vars (contracts/SToken.sol#509) is a local variable never initialized
SToken.redeemFresh(address,uint256,uint256).vars (contracts/SToken.sol#616) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

SDaiDelegate._becomeImplementation(address,address) (contracts/SDaiDelegate.sol#42-67) ignores return value by pot.drip() (contracts/SDaiDelegate.sol#63)
SDaiDelegate._resignImplementation() (contracts/SDaiDelegate.sol#72-92) ignores return value by pot.drip() (contracts/SDaiDelegate.sol#81)
SDaiDelegate.accrueInterest() (contracts/SDaiDelegate.sol#101-107) ignores return value by PotLike(potAddress).drip() (contracts/SDaiDelegate.sol#103)
SErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SErc20.sol#21-34) ignores return value by EIP20Interface(underlying).totalSupply() (contracts/SErc20.sol#33)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Exponential.divScalarByExpTruncate(uint256,Exponential.Exp).fraction (contracts/Exponential.sol#135) shadows:
        - Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

SErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8).underlying_ (contracts/SErc20.sol#21) lacks a zero-check on :
                - underlying = underlying_ (contracts/SErc20.sol#32)
SToken._setPendingAdmin(address).newPendingAdmin (contracts/SToken.sol#1135) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (contracts/SToken.sol#1145)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in SDaiDelegate._becomeImplementation(address,address) (contracts/SDaiDelegate.sol#42-67):
        External calls:
        - dai = daiJoin.dai() (contracts/SDaiDelegate.sol#46)
        - vat = daiJoin.vat() (contracts/SDaiDelegate.sol#47)
        State variables written after the call(s):
        - daiJoinAddress = daiJoinAddress_ (contracts/SDaiDelegate.sol#51)
        - potAddress = potAddress_ (contracts/SDaiDelegate.sol#52)
        - vatAddress = address(vat) (contracts/SDaiDelegate.sol#53)
Reentrancy in SDaiDelegate.accrueInterest() (contracts/SDaiDelegate.sol#101-107):
        External calls:
        - PotLike(potAddress).drip() (contracts/SDaiDelegate.sol#103)
        State variables written after the call(s):
        - super.accrueInterest() (contracts/SDaiDelegate.sol#106)
                - accrualBlockNumber = currentBlockNumber (contracts/SToken.sol#453)
        - super.accrueInterest() (contracts/SDaiDelegate.sol#106)
                - borrowIndex = borrowIndexNew (contracts/SToken.sol#454)
        - super.accrueInterest() (contracts/SDaiDelegate.sol#106)
                - totalBorrows = totalBorrowsNew (contracts/SToken.sol#455)
        - super.accrueInterest() (contracts/SDaiDelegate.sol#106)
                - totalReserves = totalReservesNew (contracts/SToken.sol#456)
Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#788)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#789)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#790)
```

```
Reentrancy in SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/SToken.sol#499)
        State variables written after the call(s):
        - accountTokens[minter] = vars.accountTokensNew (contracts/SToken.sol#551)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#550)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - accountTokens[redeemer] = vars.accountTokensNew (contracts/SToken.sol#697)
Reentrancy in SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#905)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#906)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
Reentrancy in SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
        State variables written after the call(s):
        - accountTokens[borrower] = vars.borrowerTokensNew (contracts/SToken.sol#1112)
        - accountTokens[liquidator] = vars.liquidatorTokensNew (contracts/SToken.sol#1113)
        - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#1111)
Reentrancy in SToken.transferTokens(address,address,address,uint256) (contracts/SToken.sol#68-127):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/SToken.sol#70)
        State variables written after the call(s):
        - accountTokens[src] = srcTokensNew (contracts/SToken.sol#113)
        - accountTokens[dst] = dstTokensNew (contracts/SToken.sol#114)
        - transferAllowances[src][spender] = allowanceNew (contracts/SToken.sol#118)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in SDaiDelegate.accrueInterest() (contracts/SDaiDelegate.sol#101-107):
        External calls:
        - PotLike(potAddress).drip() (contracts/SDaiDelegate.sol#103)
        Event emitted after the call(s):
        - AccrueInterest(cashPrior,interestAccumulated,borrowIndexNew,totalBorrowsNew) (contracts/SToken.sol#459)
                - super.accrueInterest() (contracts/SDaiDelegate.sol#106)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - super.accrueInterest() (contracts/SDaiDelegate.sol#106)
Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        Event emitted after the call(s):
        - Borrow(borrower,borrowAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#793)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.BORROW_CASH_NOT_AVAILABLE) (contracts/SToken.sol#750)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#762)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#767)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#772)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.COMPTROLLER_REJECTION,FailureInfo.BORROW_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#740)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.BORROW_FRESHNESS_CHECK) (contracts/SToken.sol#745)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.COMPTROLLER_REJECTION,FailureInfo.LIQUIDATE_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#956)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.LIQUIDATE_FRESHNESS_CHECK),0) (contracts/SToken.sol#961)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED,FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_ZERO),0) (contracts/SToken.sol#976)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.LIQUIDATE_COLLATERAL_FRESHNESS_CHECK),0) (contracts/SToken.sol#966)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED,FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_UINT_MAX),0) (contracts/SToken.sol#981)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_ACCOUNT_PAIR,FailureInfo.LIQUIDATE_LIQUIDATOR_IS_BORROWER),0) (contracts/SToken.sol#971)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - (fail(TokenErrorReporter.Error(repayBorrowError),FailureInfo.LIQUIDATE_REPAY_BORROW_FRESH_FAILED),0) (contracts/SToken.sol#988)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - TransferBorrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
        - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(sTokenCollateral),seizeTokens) (contracts/SToken.sol#1014)
Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(TokenErrorReporter.Error(error),FailureInfo.LIQUIDATE_ACCRUE_COLLATERAL_INTEREST_FAILED),0) (contracts/SToken.sol#936)
Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(sTokenCollateral),seizeTokens) (contracts/SToken.sol#1014)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
```

```
Reentrancy in SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/SToken.sol#499)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail1Opaque(Error.COMPTROLLER_REJECTION,FailureInfo.MINT_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#501)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.MINT_FRESHNESS_CHECK),0) (contracts/SToken.sol#506)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail1Opaque(Error.MATH_ERROR,FailureInfo.MINT_EXCHANGE_RATE_READ_FAILED,uint256(vars.mathErr)),0) (contracts/SToken.sol#513)
        - Mint(minter,vars.actualMintAmount,vars.mintTokens) (contracts/SToken.sol#554)
        - Transfer(address(this),minter,vars.mintTokens) (contracts/SToken.sol#555)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.REDEEM_FRESHNESS_CHECK) (contracts/SToken.sol#660)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail1Opaque(Error.COMPTROLLER_REJECTION,FailureInfo.REDEEM_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#655)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail1Opaque(Error.MATH_ERROR,FailureInfo.REDEEM_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#675)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.REDEEM_TRANSFER_OUT_NOT_POSSIBLE) (contracts/SToken.sol#680)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail1Opaque(Error.MATH_ERROR,FailureInfo.REDEEM_NEW_TOTAL_SUPPLY_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#670)
        - Redeem(redeemer,vars.redeemAmount,vars.redeemTokens) (contracts/SToken.sol#701)
        - Transfer(redeemer,address(this),vars.redeemTokens) (contracts/SToken.sol#700)
Reentrancy in SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail1Opaque(Error.MATH_ERROR,FailureInfo.REPAY_BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)),0) (contracts/SToken.sol#870)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail1Opaque(Error.COMPTROLLER_REJECTION,FailureInfo.REPAY_BORROW_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#854)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail1(Error.MARKET_NOT_FRESH,FailureInfo.REPAY_BORROW_FRESHNESS_CHECK),0) (contracts/SToken.sol#859)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
Reentrancy in SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail1(Error.INVALID_ACCOUNT_PAIR,FailureInfo.LIQUIDATE_SEIZE_LIQUIDATOR_IS_BORROWER) (contracts/SToken.sol#1066)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail1Opaque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_INCREMENT_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#1102)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail1Opaque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_DECREMENT_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#1093)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail1Opaque(Error.COMPTROLLER_REJECTION,FailureInfo.LIQUIDATE_SEIZE_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#1061)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail1Opaque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_DECREMENT_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#1082)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
Reentrancy in SToken.transferTokens(address,address,address,uint256) (contracts/SToken.sol#68-127):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/SToken.sol#70)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.BAD_INPUT,FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/SToken.sol#77)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/SToken.sol#96)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail1Opaque(Error.COMPTROLLER_REJECTION,FailureInfo.TRANSFER_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#72)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_NOT_ENOUGH) (contracts/SToken.sol#101)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_TOO_MUCH) (contracts/SToken.sol#106)
        - Transfer(src,dst,tokens) (contracts/SToken.sol#122)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

SErc20.doTransferIn(address,uint256) (contracts/SErc20.sol#142-167) uses assembly
        - INLINE ASM (contracts/SErc20.sol#148-160)
SErc20.doTransferOut(address,uint256) (contracts/SErc20.sol#178-197) uses assembly
        - INLINE ASM (contracts/SErc20.sol#183-195)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) is never used and should be removed
ComptrollerErrorReporter.fail1Opaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) is never used and should be removed
Exponential.addExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#408-52) is never used and should be removed
Exponential.add_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#246-248) is never used and should be removed
Exponential.add_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#242-244) is never used and should be removed
Exponential.divExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#191-193) is never used and should be removed
Exponential.divScalar(Exponential.Exp,uint256) (contracts/Exponential.sol#102-109) is never used and should be removed
Exponential.div_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#326-328) is never used and should be removed
Exponential.div_(Exponential.Double,uint256) (contracts/Exponential.sol#330-332) is never used and should be removed
Exponential.div_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#314-316) is never used and should be removed
Exponential.div_(Exponential.Exp,uint256) (contracts/Exponential.sol#318-320) is never used and should be removed
Exponential.div_(uint256,Exponential.Double) (contracts/Exponential.sol#334-336) is never used and should be removed
Exponential.div_(uint256,Exponential.Exp) (contracts/Exponential.sol#322-324) is never used and should be removed
Exponential.div_(uint256,uint256) (contracts/Exponential.sol#338-340) is never used and should be removed
Exponential.div_(uint256,uint256,string) (contracts/Exponential.sol#342-345) is never used and should be removed
Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) is never used and should be removed
Exponential.greaterThanExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#221-223) is never used and should be removed
Exponential.isZeroExp(Exponential.Exp) (contracts/Exponential.sol#228-230) is never used and should be removed
Exponential.lessThanExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#207-209) is never used and should be removed
Exponential.lessThanOrEqualExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#210-216) is never used and should be removed
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) is never used and should be removed
Exponential.mulExp(uint256,uint256) (contracts/Exponential.sol#171-173) is never used and should be removed
Exponential.mulExp3(Exponential.Exp,Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#178-184) is never used and should be removed
Exponential.mul_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#289-291) is never used and should be removed
Exponential.mul_(Exponential.Double,uint256) (contracts/Exponential.sol#293-295) is never used and should be removed
Exponential.mul_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#277-279) is never used and should be removed
Exponential.mul_(Exponential.Exp,uint256) (contracts/Exponential.sol#281-283) is never used and should be removed
Exponential.mul_(uint256,Exponential.Double) (contracts/Exponential.sol#297-299) is never used and should be removed
Exponential.safe224(uint256,string) (contracts/Exponential.sol#232-235) is never used and should be removed
Exponential.safe32(uint256,string) (contracts/Exponential.sol#237-240) is never used and should be removed
Exponential.subExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#57-61) is never used and should be removed
Exponential.sub_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#264-266) is never used and should be removed
Exponential.sub_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#260-262) is never used and should be removed
SErc20.doTransferIn(address,uint256) (contracts/SErc20.sol#142-167) is never used and should be removed
SErc20.doTransferOut(address,uint256) (contracts/SErc20.sol#178-197) is never used and should be removed
SErc20.getCashPrior() (contracts/SErc20.sol#128-131) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Constant ComptrollerInterface.isComptroller (contracts/ComptrollerInterface.sol#5) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.expScale (contracts/Exponential.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.doubleScale (contracts/Exponential.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.halfExpScale (contracts/Exponential.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.mantissaOne (contracts/Exponential.sol#16) is not in UPPER_CASE_WITH_UNDERSCORES
Constant InterestRateModel.isInterestRateModel (contracts/InterestRateModel.sol#9) is not in UPPER_CASE_WITH_UNDERSCORES
Function SDaiDelegate._becomeImplementation(bytes) (contracts/SDaiDelegate.sol#30-35) is not in mixedCase
Function SDaiDelegate._resignImplementation() (contracts/SDaiDelegate.sol#72-92) is not in mixedCase
Function SErc20._addReserves(uint256) (contracts/SErc20.sol#117-119) is not in mixedCase
Function SErc20Delegate._becomeImplementation(bytes) (contracts/SErc20Delegate.sol#20-30) is not in mixedCase
Function SErc20Delegate._resignImplementation() (contracts/SErc20Delegate.sol#35-42) is not in mixedCase
Function SToken._setPendingAdmin(address) (contracts/SToken.sol#1135-1151) is not in mixedCase
Function SToken._acceptAdmin() (contracts/SToken.sol#1158-1178) is not in mixedCase
Function SToken._setComptroller(ComptrollerInterface) (contracts/SToken.sol#1185-1202) is not in mixedCase
Function SToken._setReserveFactor(uint256) (contracts/SToken.sol#1209-1217) is not in mixedCase
Function SToken._reduceReserves(uint256) (contracts/SToken.sol#1316-1324) is not in mixedCase
Function SToken._transferReserves(uint256) (contracts/SToken.sol#1380-1388) is not in mixedCase
Function SToken._setInterestRateModel(InterestRateModel) (contracts/SToken.sol#1445-1453) is not in mixedCase
Variable STokenStorage._notEntered (contracts/STokenInterfaces.sol#10) is not in mixedCase
Constant STokenStorage.borrowRateMaxMantissa (contracts/STokenInterfaces.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.reserveFactorMaxMantissa (contracts/STokenInterfaces.sol#36) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.protocolSeizeShareMantissa (contracts/STokenInterfaces.sol#121) is not in UPPER_CASE_WITH_UNDERSCORES
Function STokenInterface._setPendingAdmin(address) (contracts/STokenInterfaces.sol#246) is not in mixedCase
Function STokenInterface._acceptAdmin() (contracts/STokenInterfaces.sol#247) is not in mixedCase
Function STokenInterface._setComptroller(ComptrollerInterface) (contracts/STokenInterfaces.sol#248) is not in mixedCase
Function STokenInterface._setReserveFactor(uint256) (contracts/STokenInterfaces.sol#249) is not in mixedCase
Function STokenInterface._reduceReserves(uint256) (contracts/STokenInterfaces.sol#250) is not in mixedCase
Function STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251) is not in mixedCase
Constant STokenInterface.isSToken (contracts/STokenInterfaces.sol#129) is not in UPPER_CASE_WITH_UNDERSCORES
Function SErc20Interface._addReserves(uint256) (contracts/STokenInterfaces.sol#276) is not in mixedCase
Function SDelegatorInterface._setImplementation(address,bool,bytes) (contracts/STokenInterfaces.sol#298) is not in mixedCase
Function SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307) is not in mixedCase
Function SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "data (contracts/SErc20Delegate.sol#22)" inSErc20Delegate (contracts/SErc20Delegate.sol#10-43)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
```

```
Variable SToken.seize(address,address,uint256).seizeTokens (contracts/SToken.sol#1031) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable STokenInterface.seize(address,address,uint256).seizeTokens (contracts/STokenInterfaces.sol#241) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface).seizeTokens (contracts/SToken.sol#996) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable SToken.seizeInternal(address,address,address,uint256).seizeTokens (contracts/SToken.sol#1057) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

_becomeImplementation(bytes) should be declared external:
        - SDaiDelegate._becomeImplementation(bytes) (contracts/SDaiDelegate.sol#30-35)
        - SErc20Delegate._becomeImplementation(bytes) (contracts/SErc20Delegate.sol#20-30)
_resignImplementation() should be declared external:
        - SDaiDelegate._resignImplementation() (contracts/SDaiDelegate.sol#72-92)
        - SErc20Delegate._resignImplementation() (contracts/SErc20Delegate.sol#35-42)
initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - SErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SErc20.sol#21-34)
_setInterestRateModel(InterestRateModel) should be declared external:
        - SToken._setInterestRateModel(InterestRateModel) (contracts/SToken.sol#1445-1453)
        - STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (contracts/STokenInterfaces.sol#298)
_becomeImplementation(bytes) should be declared external:
        - SDaiDelegate._becomeImplementation(bytes) (contracts/SDaiDelegate.sol#30-35)
        - SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307)
        - SErc20Delegate._becomeImplementation(bytes) (contracts/SErc20Delegate.sol#20-30)
_resignImplementation() should be declared external:
        - SDaiDelegate._resignImplementation() (contracts/SDaiDelegate.sol#72-92)
        - SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312)
        - SErc20Delegate._resignImplementation() (contracts/SErc20Delegate.sol#35-42)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

SErc20.sol

```
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#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/SToken.sol#390)
SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - require(bool,string)(mathErr == MathError.NO_ERROR,could not calculate block delta) (contracts/SToken.sol#406)
SToken.balanceOfUnderlying(address) (contracts/SToken.sol#190-195) uses a dangerous strict equality:
        - require(bool,string)(mErr == MathError.NO_ERROR,balance could not be calculated) (contracts/SToken.sol#193)
SToken.borrowBalanceStored(address) (contracts/SToken.sol#271-275) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,borrowBalanceStored: borrowBalanceStoredInternal failed) (contracts/SToken.sol#273)
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) uses a dangerous strict equality:
        - b == 0 (contracts/CarefulMath.sol#42)
SToken.exchangeRateStored() (contracts/SToken.sol#328-332) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,exchangeRateStored: exchangeRateStoredInternal failed) (contracts/SToken.sol#330)
SToken.exchangeRateStoredInternal() (contracts/SToken.sol#339-369) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/SToken.sol#341)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/SToken.sol#33)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting comptroller failed) (contracts/SToken.sol#41)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting interest rate model failed) (contracts/SToken.sol#49)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == uint256(Error.NO_ERROR),LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/SToken.sol#997)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(seizeError == uint256(Error.NO_ERROR),token seizure failed) (contracts/SToken.sol#1011)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_EXCHANGE_CALCULATION_FAILED) (contracts/SToken.sol#536)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_TOTAL_SUPPLY_CALCULATION_FAILED) (contracts/SToken.sol#544)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#547)
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) uses a dangerous strict equality:
        - assert(bool)(err2 == MathError.NO_ERROR) (contracts/Exponential.sol#163)
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) uses a dangerous strict equality:
        - a == 0 (contracts/CarefulMath.sol#25)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - a == 0 || b == 0 (contracts/Exponential.sol#306)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - require(bool,string)(c / a == b,errorMessage) (contracts/Exponential.sol#310)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#858-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#899)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#858-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#902)
SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,exchange rate math error) (contracts/SToken.sol#1089)
SToken.transfer(address,uint256) (contracts/SToken.sol#135-137) uses a dangerous strict equality:
        - transferTokens(msg.sender,msg.sender,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#136)
SToken.transferFrom(address,address,uint256) (contracts/SToken.sol#146-148) uses a dangerous strict equality:
        - transferTokens(msg.sender,src,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#696)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

SToken.mintFresh(address,uint256).vars (contracts/SToken.sol#509) is a local variable never initialized
SToken.redeemFresh(address,uint256,uint256).vars (contracts/SToken.sol#616) is a local variable never initialized
SToken.seizeInternal(address,address,address,uint256).vars (contracts/SToken.sol#1073) is a local variable never initialized
SToken.repayBorrowFresh(address,address,uint256).vars (contracts/SToken.sol#862) is a local variable never initialized
SToken.borrowFresh(address,uint256).vars (contracts/SToken.sol#753) is a local variable never initialized
SToken._addReservesFresh(uint256).actualAddAmount (contracts/SToken.sol#1274) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

SErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SErc20.sol#21-34) ignores return value by EIP20Interface(underlying).totalSupply() (contracts/SErc20.sol#33)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Exponential.divScalarByExpTruncate(uint256,Exponential.Exp).fraction (contracts/Exponential.sol#135) shadows:
        - Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

SToken._setPendingAdmin(address).newPendingAdmin (contracts/SToken.sol#1135) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (contracts/SToken.sol#1145)
SErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8).underlying_ (contracts/SErc20.sol#21) lacks a zero-check on :
                - underlying = underlying_ (contracts/SErc20.sol#32)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#788)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#789)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#790)
Reentrancy in SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/SToken.sol#499)
        State variables written after the call(s):
        - accountTokens[minter] = vars.accountTokensNew (contracts/SToken.sol#551)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#550)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
```

SErc20Delegate.sol

```
SErc20Delegate._becomeImplementation(bytes) (contracts/SErc20Delegate.sol#20-30) uses a Boolean constant improperly:
        -false (contracts/SErc20Delegate.sol#25)
SErc20Delegate._resignImplementation() (contracts/SErc20Delegate.sol#35-42) uses a Boolean constant improperly:
        -false (contracts/SErc20Delegate.sol#37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#misuse-of-a-boolean-constant

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#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/SToken.sol#390)
SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - require(bool,string)(mathErr == MathError.NO_ERROR,could not calculate block delta) (contracts/SToken.sol#406)
SToken.balanceOfUnderlying(address) (contracts/SToken.sol#190-195) uses a dangerous strict equality:
        - require(bool,string)(mErr == MathError.NO_ERROR,balance could not be calculated) (contracts/SToken.sol#193)
SToken.borrowBalanceStored(address) (contracts/SToken.sol#271-275) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,borrowBalanceStored: borrowBalanceStoredInternal failed) (contracts/SToken.sol#273)
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) uses a dangerous strict equality:
        - b == 0 (contracts/CarefulMath.sol#42)
SToken.exchangeRateStored() (contracts/SToken.sol#328-332) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,exchangeRateStored: exchangeRateStoredInternal failed) (contracts/SToken.sol#330)
SToken.exchangeRateStoredInternal() (contracts/SToken.sol#339-369) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/SToken.sol#341)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/SToken.sol#33)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting comptroller failed) (contracts/SToken.sol#41)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting interest rate model failed) (contracts/SToken.sol#49)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == uint256(Error.NO_ERROR),LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/SToken.sol#997)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(seizeError == uint256(Error.NO_ERROR),token seizure failed) (contracts/SToken.sol#1011)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_EXCHANGE_CALCULATION_FAILED) (contracts/SToken.sol#536)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_TOTAL_SUPPLY_CALCULATION_FAILED) (contracts/SToken.sol#544)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#547)
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) uses a dangerous strict equality:
        - assert(bool)(err2 == MathError.NO_ERROR) (contracts/Exponential.sol#163)
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) uses a dangerous strict equality:
        - a == 0 (contracts/CarefulMath.sol#25)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - a == 0 || b == 0 (contracts/Exponential.sol#306)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - require(bool,string)(c / a == b,errorMessage) (contracts/Exponential.sol#310)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#899)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#902)
SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,exchange rate math error) (contracts/SToken.sol#1089)
SToken.transfer(address,uint256) (contracts/SToken.sol#135-137) uses a dangerous strict equality:
        - transferTokens(msg.sender,msg.sender,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#136)
SToken.transferFrom(address,address,uint256) (contracts/SToken.sol#146-148) uses a dangerous strict equality:
        - transferTokens(msg.sender,src,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#696)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

SToken.borrowFresh(address,uint256).vars (contracts/SToken.sol#753) is a local variable never initialized
SToken.repayBorrowFresh(address,address,uint256).vars (contracts/SToken.sol#862) is a local variable never initialized
SToken.seizeInternal(address,address,address,uint256).vars (contracts/SToken.sol#1073) is a local variable never initialized
SToken.redeemFresh(address,uint256,uint256).vars (contracts/SToken.sol#616) is a local variable never initialized
SToken.mintFresh(address,uint256).vars (contracts/SToken.sol#509) is a local variable never initialized
SToken._addReservesFresh(uint256).actualAddAmount (contracts/SToken.sol#1274) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

SErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SErc20.sol#21-34) ignores return value by EIP20Interface(underlying).totalSupply() (contracts/SErc20.sol#33)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Exponential.divScalarByExpTruncate(uint256,Exponential.Exp).fraction (contracts/Exponential.sol#135) shadows:
        - Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

SErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8).underlying_ (contracts/SErc20.sol#21) lacks a zero-check on :
                - underlying = underlying_ (contracts/SErc20.sol#32)
SToken._setPendingAdmin(address).newPendingAdmin (contracts/SToken.sol#1135) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (contracts/SToken.sol#1145)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#788)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#789)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#790)
Reentrancy in SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/SToken.sol#499)
        State variables written after the call(s):
        - accountTokens[minter] = vars.accountTokensNew (contracts/SToken.sol#551)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#550)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - accountTokens[redeemer] = vars.accountTokensNew (contracts/SToken.sol#697)
Reentrancy in SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#905)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#906)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
Reentrancy in SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
        State variables written after the call(s):
        - accountTokens[borrower] = vars.borrowerTokensNew (contracts/SToken.sol#1112)
        - accountTokens[liquidator] = vars.liquidatorTokensNew (contracts/SToken.sol#1113)
        - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#1111)
Reentrancy in SToken.transferTokens(address,address,address,uint256) (contracts/SToken.sol#68-127):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/SToken.sol#70)
        State variables written after the call(s):
        - accountTokens[src] = srsTokensNew (contracts/SToken.sol#113)
        - accountTokens[dst] = dstTokensNew (contracts/SToken.sol#114)
        - transferAllowances[src][spender] = allowanceNew (contracts/SToken.sol#118)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        Event emitted after the call(s):
```

```
        - Borrow(borrower,borrowAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/STokon.sol#793)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.BORROW_COMPTROLLER_REJECTION,allowed) (contracts/STokon.sol#740)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.BORROW_CASH_NOT_AVAILABLE) (contracts/STokon.sol#750)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.BORROW_FRESHNESS_CHECK) (contracts/STokon.sol#745)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/STokon.sol#767)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/STokon.sol#762)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/STokon.sol#772)
Reentrancy in STokon.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/STokon.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/STokon.sol#954)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.LIQUIDATE_COMPTROLLER_REJECTION,allowed),0) (contracts/STokon.sol#956)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.LIQUIDATE_COLLATERAL_FRESHNESS_CHECK),0) (contracts/STokon.sol#966)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED,FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_UINT_MAX),0) (contracts/STokon.sol#981)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_ACCOUNT_PAIR,FailureInfo.LIQUIDATE_LIQUIDATOR_IS_BORROWER),0) (contracts/STokon.sol#971)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED,FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_ZERO),0) (contracts/STokon.sol#976)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.LIQUIDATE_FRESHNESS_CHECK),0) (contracts/STokon.sol#961)
Reentrancy in STokon.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/STokon.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/STokon.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/STokon.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/STokon.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/STokon.sol#913)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/STokon.sol#986)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(TokenErrorReporter.Error(repayBorrowError),FailureInfo.LIQUIDATE_REPAY_BORROW_FRESH_FAILED),0) (contracts/STokon.sol#988)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/STokon.sol#986)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/STokon.sol#910)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/STokon.sol#986)
Reentrancy in STokon.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/STokon.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/STokon.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/STokon.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/STokon.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/STokon.sol#913)
        - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/STokon.sol#1005)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/STokon.sol#1059)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/STokon.sol#1121)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/STokon.sol#1005)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/STokon.sol#1005)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/STokon.sol#1118)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/STokon.sol#1005)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/STokon.sol#1116)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/STokon.sol#1005)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/STokon.sol#1117)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/STokon.sol#1005)
Reentrancy in STokon.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/STokon.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/STokon.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/STokon.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/STokon.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/STokon.sol#913)
        - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/STokon.sol#1005)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/STokon.sol#1059)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/STokon.sol#1121)
        - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/STokon.sol#1007)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(sTokenCollateral),seizeTokens) (contracts/STokon.sol#1014)
Reentrancy in STokon.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/STokon.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/STokon.sol#933)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(TokenErrorReporter.Error(error),FailureInfo.LIQUIDATE_ACCRUE_COLLATERAL_INTEREST_FAILED),0) (contracts/STokon.sol#936)
Reentrancy in STokon.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/STokon.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/STokon.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/STokon.sol#940)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/STokon.sol#852)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/STokon.sol#1059)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/STokon.sol#954)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/STokon.sol#913)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/STokon.sol#1007)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/STokon.sol#1121)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/STokon.sol#1017)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/STokon.sol#940)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/STokon.sol#940)
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(sTokenCollateral),seizeTokens) (contracts/STokon.sol#1014)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/STokon.sol#940)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/STokon.sol#910)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/STokon.sol#940)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/STokon.sol#1118)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/STokon.sol#940)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/STokon.sol#1116)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/STokon.sol#940)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/STokon.sol#1117)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/STokon.sol#940)
Reentrancy in STokon.mintFresh(address,uint256) (contracts/STokon.sol#497-561):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/STokon.sol#499)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.MINT_COMPTROLLER_REJECTION,allowed),0) (contracts/STokon.sol#501)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.MINT_EXCHANGE_RATE_READ_FAILED,uint256(vars.mathErr)),0) (contracts/STokon.sol#513)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.MINT_FRESHNESS_CHECK),0) (contracts/STokon.sol#506)
        - Mint(minter,vars.actualMintAmount,vars.mintTokens) (contracts/STokon.sol#554)
        - Transfer(address(this),minter,vars.mintTokens) (contracts/STokon.sol#555)
Reentrancy in STokon.redeemFresh(address,uint256,uint256) (contracts/STokon.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/STokon.sol#653)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.REDEEM_NEW_TOTAL_SUPPLY_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/STokon.sol#670)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.REDEEM_FRESHNESS_CHECK) (contracts/STokon.sol#660)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.REDEEM_TRANSFER_OUT_NOT_POSSIBLE) (contracts/STokon.sol#680)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.REDEEM_COMPTROLLER_REJECTION,allowed) (contracts/STokon.sol#655)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.REDEEM_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/STokon.sol#675)
        - Redeem(redeemer,vars.redeemAmount,vars.redeemTokens) (contracts/STokon.sol#701)
        - Transfer(redeemer,address(this),vars.redeemTokens) (contracts/STokon.sol#700)
Reentrancy in STokon.repayBorrowFresh(address,address,uint256) (contracts/STokon.sol#850-916):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/STokon.sol#852)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.REPAY_BORROW_FRESHNESS_CHECK),0) (contracts/STokon.sol#859)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.REPAY_BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)),0) (contracts/STokon.sol#870)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.REPAY_BORROW_COMPTROLLER_REJECTION,allowed),0) (contracts/STokon.sol#854)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/STokon.sol#910)
Reentrancy in STokon.seizeInternal(address,address,address,uint256) (contracts/STokon.sol#1057-1124):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/STokon.sol#1059)
        Event emitted after the call(s):
```

```
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_DECREMENT_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#1082)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_INCREMENT_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#1102)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_ACCOUNT_PAIR,FailureInfo.LIQUIDATE_SEIZE_LIQUIDATOR_IS_BORROWER) (contracts/SToken.sol#1066)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.LIQUIDATE_SEIZE_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#1061)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_DECREMENT_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#1093)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
Reentrancy in SToken.transferTokens(address,address,address,uint256) (contracts/SToken.sol#68-127):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/SToken.sol#70)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_NOT_ENOUGH) (contracts/SToken.sol#101)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.BAD_INPUT,FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/SToken.sol#77)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/SToken.sol#96)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_TOO_MUCH) (contracts/SToken.sol#106)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.TRANSFER_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#72)
        - Transfer(src,dst,tokens) (contracts/SToken.sol#122)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

SErc20.doTransferIn(address,uint256) (contracts/SErc20.sol#142-167) uses assembly
        - INLINE ASM (contracts/SErc20.sol#148-160)
SErc20.doTransferOut(address,uint256) (contracts/SErc20.sol#178-197) uses assembly
        - INLINE ASM (contracts/SErc20.sol#183-195)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) is never used and should be removed
ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) is never used and should be removed
Exponential.addExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#48-52) is never used and should be removed
Exponential.add_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#246-248) is never used and should be removed
Exponential.add_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#242-244) is never used and should be removed
Exponential.divExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#191-193) is never used and should be removed
Exponential.divScalar(Exponential.Exp,uint256) (contracts/Exponential.sol#102-109) is never used and should be removed
Exponential.div_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#326-328) is never used and should be removed
Exponential.div_(Exponential.Double,uint256) (contracts/Exponential.sol#330-332) is never used and should be removed
Exponential.div_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#314-316) is never used and should be removed
Exponential.div_(Exponential.Exp,uint256) (contracts/Exponential.sol#318-320) is never used and should be removed
Exponential.div_(uint256,Exponential.Double) (contracts/Exponential.sol#334-336) is never used and should be removed
Exponential.div_(uint256,Exponential.Exp) (contracts/Exponential.sol#322-324) is never used and should be removed
Exponential.div_(uint256,uint256) (contracts/Exponential.sol#338-340) is never used and should be removed
Exponential.div_(uint256,uint256,string) (contracts/Exponential.sol#342-345) is never used and should be removed
Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) is never used and should be removed
Exponential.greaterThanExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#221-223) is never used and should be removed
Exponential.isZeroExp(Exponential.Exp) (contracts/Exponential.sol#228-230) is never used and should be removed
Exponential.lessThanExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#207-209) is never used and should be removed
Exponential.lessThanOrEqualExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#214-216) is never used and should be removed
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) is never used and should be removed
Exponential.mulExp(uint256,uint256) (contracts/Exponential.sol#171-173) is never used and should be removed
Exponential.mulExp3(Exponential.Exp,Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#178-184) is never used and should be removed
Exponential.mul_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#289-291) is never used and should be removed
Exponential.mul_(Exponential.Double,uint256) (contracts/Exponential.sol#293-295) is never used and should be removed
Exponential.mul_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#277-279) is never used and should be removed
Exponential.mul_(Exponential.Exp,uint256) (contracts/Exponential.sol#281-283) is never used and should be removed
Exponential.mul_(uint256,Exponential.Double) (contracts/Exponential.sol#297-299) is never used and should be removed
Exponential.safe224(uint256,string) (contracts/Exponential.sol#232-235) is never used and should be removed
Exponential.safe32(uint256,string) (contracts/Exponential.sol#237-240) is never used and should be removed
Exponential.subExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#57-61) is never used and should be removed
Exponential.sub_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#264-266) is never used and should be removed
Exponential.sub_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#260-262) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Constant ComptrollerInterface.isComptroller (contracts/ComptrollerInterface.sol#5) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.expScale (contracts/Exponential.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.doubleScale (contracts/Exponential.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.halfExpScale (contracts/Exponential.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.mantissaOne (contracts/Exponential.sol#16) is not in UPPER_CASE_WITH_UNDERSCORES
Constant InterestRateModel.isInterestRateModel (contracts/InterestRateModel.sol#9) is not in UPPER_CASE_WITH_UNDERSCORES
Function SErc20._addReserves(uint256) (contracts/SErc20.sol#117-119) is not in mixedCase
Function SErc20Delegate._becomeImplementation(bytes) (contracts/SErc20Delegate.sol#20-30) is not in mixedCase
Function SErc20Delegate._resignImplementation() (contracts/SErc20Delegate.sol#35-42) is not in mixedCase
Function SToken._setPendingAdmin(address) (contracts/SToken.sol#1135-1151) is not in mixedCase
Function SToken._acceptAdmin() (contracts/SToken.sol#1158-1178) is not in mixedCase
Function SToken._setComptroller(ComptrollerInterface) (contracts/SToken.sol#1185-1202) is not in mixedCase
Function SToken._setReserveFactor(uint256) (contracts/SToken.sol#1209-1217) is not in mixedCase
Function SToken._reduceReserves(uint256) (contracts/SToken.sol#1316-1324) is not in mixedCase
Function SToken._transferReserves(uint256) (contracts/SToken.sol#1380-1388) is not in mixedCase
Function SToken._setInterestRateModel(InterestRateModel) (contracts/SToken.sol#1445-1453) is not in mixedCase
Variable STokenStorage._notEntered (contracts/STokenInterfaces.sol#10) is not in mixedCase
Constant STokenStorage.borrowRateMaxMantissa (contracts/STokenInterfaces.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.reserveFactorMaxMantissa (contracts/STokenInterfaces.sol#36) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.protocolSeizeShareMantissa (contracts/STokenInterfaces.sol#121) is not in UPPER_CASE_WITH_UNDERSCORES
Function STokenInterface._setPendingAdmin(address) (contracts/STokenInterfaces.sol#246) is not in mixedCase
Function STokenInterface._acceptAdmin() (contracts/STokenInterfaces.sol#247) is not in mixedCase
Function STokenInterface._setComptroller(ComptrollerInterface) (contracts/STokenInterfaces.sol#248) is not in mixedCase
Function STokenInterface._setReserveFactor(uint256) (contracts/STokenInterfaces.sol#249) is not in mixedCase
Function STokenInterface._reduceReserves(uint256) (contracts/STokenInterfaces.sol#250) is not in mixedCase
Function STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251) is not in mixedCase
Constant STokenInterface.isSToken (contracts/STokenInterfaces.sol#129) is not in UPPER_CASE_WITH_UNDERSCORES
Function SErc20Interface._addReserves(uint256) (contracts/STokenInterfaces.sol#276) is not in mixedCase
Function SDelegatorInterface._setImplementation(address,bool,bytes) (contracts/STokenInterfaces.sol#298) is not in mixedCase
Function SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307) is not in mixedCase
Function SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "data (contracts/SErc20Delegate.sol#22)" inSErc20Delegate (contracts/SErc20Delegate.sol#10-43)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable SToken.seize(address,address,uint256).seizeTokens (contracts/SToken.sol#1031) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface).seizeTokens (contracts/SToken.sol#996) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable SToken.seizeInternal(address,address,address,uint256).seizeTokens (contracts/SToken.sol#1057) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable STokenInterface.seize(address,address,uint256).seizeTokens (contracts/STokenInterfaces.sol#241) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - SErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SErc20.sol#21-34)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307)
        - SErc20Delegate._becomeImplementation(bytes) (contracts/SErc20Delegate.sol#20-30)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312)
        - SErc20Delegate._resignImplementation() (contracts/SErc20Delegate.sol#35-42)
_setInterestRateModel(InterestRateModel) should be declared external:
        - SToken._setInterestRateModel(InterestRateModel) (contracts/SToken.sol#1445-1453)
        - STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (contracts/STokenInterfaces.sol#298)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

SErc20Delegator.sol

```
SErc20Delegator.delegateTo(address,bytes) (contracts/SErc20Delegator.sol#421-429) uses delegatecall to a input-controlled function id
        - (success,returnData) = callee.delegatecall(data) (contracts/SErc20Delegator.sol#422)
SErc20Delegator.fallback() (contracts/SErc20Delegator.sol#462-476) uses delegatecall to a input-controlled function id
        - (success) = implementation.delegatecall(msg.data) (contracts/SErc20Delegator.sol#466)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-delegatecall

Reentrancy in SErc20Delegator._setImplementation(address,bool,bytes) (contracts/SErc20Delegator.sol#60-73):
        External calls:
        - delegateToImplementation(abi.encodeWithSignature(_resignImplementation())) (contracts/SErc20Delegator.sol#64)
                - (success,returnData) = callee.delegatecall(data) (contracts/SErc20Delegator.sol#422)
        State variables written after the call(s):
        - implementation = implementation_ (contracts/SErc20Delegator.sol#68)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

SErc20Delegator.constructor(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8,address,address,bytes).admin_ (contracts/SErc20Delegator.sol#31) lacks a zero-check on :
                - admin = admin_ (contracts/SErc20Delegator.sol#51)
SErc20Delegator._setImplementation(address,bool,bytes).implementation_ (contracts/SErc20Delegator.sol#60) lacks a zero-check on :
                - implementation = implementation_ (contracts/SErc20Delegator.sol#68)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Reentrancy in SErc20Delegator._setImplementation(address,bool,bytes) (contracts/SErc20Delegator.sol#60-73):
        External calls:
        - delegateToImplementation(abi.encodeWithSignature(_resignImplementation())) (contracts/SErc20Delegator.sol#64)
            - (success,returnData) = callee.delegatecall(data) (contracts/SErc20Delegator.sol#422)
        - delegateToImplementation(abi.encodeWithSignature(_becomeImplementation(bytes),becomeImplementationData)) (contracts/SErc20Delegator.sol#70)
            - (success,returnData) = callee.delegatecall(data) (contracts/SErc20Delegator.sol#422)
        Event emitted after the call(s):
        - NewImplementation(oldImplementation,implementation) (contracts/SErc20Delegator.sol#72)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

SErc20Delegator.delegateTo(address,bytes) (contracts/SErc20Delegator.sol#421-429) uses assembly
        - INLINE ASM (contracts/SErc20Delegator.sol#423-427)
SErc20Delegator.delegateToViewImplementation(bytes) (contracts/SErc20Delegator.sol#448-456) uses assembly
        - INLINE ASM (contracts/SErc20Delegator.sol#450-454)
SErc20Delegator.fallback() (contracts/SErc20Delegator.sol#462-476) uses assembly
        - INLINE ASM (contracts/SErc20Delegator.sol#468-475)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Low level call in SErc20Delegator.delegateTo(address,bytes) (contracts/SErc20Delegator.sol#421-429):
        - (success,returnData) = callee.delegatecall(data) (contracts/SErc20Delegator.sol#422)
Low level call in SErc20Delegator.delegateToViewImplementation(bytes) (contracts/SErc20Delegator.sol#448-456):
        - (success,returnData) = address(this).staticcall(abi.encodeWithSignature(delegateToImplementation(bytes),data)) (contracts/SErc20Delegator.sol#449)
Low level call in SErc20Delegator.fallback() (contracts/SErc20Delegator.sol#462-476):
        - (success) = implementation.delegatecall(msg.data) (contracts/SErc20Delegator.sol#466)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Constant ComptrollerInterface.isComptroller (contracts/ComptrollerInterface.sol#5) is not in UPPER_CASE_WITH_UNDERSCORES
Constant InterestRateModel.isInterestRateModel (contracts/InterestRateModel.sol#9) is not in UPPER_CASE_WITH_UNDERSCORES
Function SErc20Delegator._setImplementation(address,bool,bytes) (contracts/SErc20Delegator.sol#60-73) is not in mixedCase
Function SErc20Delegator._setPendingAdmin(address) (contracts/SErc20Delegator.sol#338-341) is not in mixedCase
Function SErc20Delegator._setComptroller(ComptrollerInterface) (contracts/SErc20Delegator.sol#348-351) is not in mixedCase
Function SErc20Delegator._setReserveFactor(uint256) (contracts/SErc20Delegator.sol#358-361) is not in mixedCase
Function SErc20Delegator._acceptAdmin() (contracts/SErc20Delegator.sol#368-371) is not in mixedCase
Function SErc20Delegator._addReserves(uint256) (contracts/SErc20Delegator.sol#378-381) is not in mixedCase
Function SErc20Delegator._reduceReserves(uint256) (contracts/SErc20Delegator.sol#388-391) is not in mixedCase
Function SErc20Delegator._transferReserves(uint256) (contracts/SErc20Delegator.sol#398-401) is not in mixedCase
Function SErc20Delegator._setInterestRateModel(InterestRateModel) (contracts/SErc20Delegator.sol#409-412) is not in mixedCase
Variable STokenStorage._notEntered (contracts/STokenInterfaces.sol#10) is not in mixedCase
Constant STokenStorage.borrowRateMaxMantissa (contracts/STokenInterfaces.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.reserveFactorMaxMantissa (contracts/STokenInterfaces.sol#36) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.protocolSeizeShareMantissa (contracts/STokenInterfaces.sol#121) is not in UPPER_CASE_WITH_UNDERSCORES
Function STokenInterface._setPendingAdmin(address) (contracts/STokenInterfaces.sol#246) is not in mixedCase
Function STokenInterface._acceptAdmin() (contracts/STokenInterfaces.sol#247) is not in mixedCase
Function STokenInterface._setComptroller(ComptrollerInterface) (contracts/STokenInterfaces.sol#248) is not in mixedCase
Function STokenInterface._setReserveFactor(uint256) (contracts/STokenInterfaces.sol#249) is not in mixedCase
Function STokenInterface._reduceReserves(uint256) (contracts/STokenInterfaces.sol#250) is not in mixedCase
Function STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251) is not in mixedCase
Constant STokenInterface.isSToken (contracts/STokenInterfaces.sol#129) is not in UPPER_CASE_WITH_UNDERSCORES
Function SErc20Interface._addReserves(uint256) (contracts/STokenInterfaces.sol#276) is not in mixedCase
Function SDelegatorInterface._setImplementation(address,bool,bytes) (contracts/STokenInterfaces.sol#298) is not in mixedCase
Function SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307) is not in mixedCase
Function SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

STokenStorage._notEntered (contracts/STokenInterfaces.sol#10) is never used in SErc20Delegator (contracts/SErc20Delegator.sol#10-477)
STokenStorage.borrowRateMaxMantissa (contracts/STokenInterfaces.sol#31) is never used in SErc20Delegator (contracts/SErc20Delegator.sol#10-477)
STokenStorage.reserveFactorMaxMantissa (contracts/STokenInterfaces.sol#36) is never used in SErc20Delegator (contracts/SErc20Delegator.sol#10-477)
STokenStorage.initialExchangeRateMantissa (contracts/STokenInterfaces.sol#61) is never used in SErc20Delegator (contracts/SErc20Delegator.sol#10-477)
STokenStorage.accountTokens (contracts/STokenInterfaces.sol#96) is never used in SErc20Delegator (contracts/SErc20Delegator.sol#10-477)
STokenStorage.transferAllowances (contracts/STokenInterfaces.sol#101) is never used in SErc20Delegator (contracts/SErc20Delegator.sol#10-477)
STokenStorage.accountBorrows (contracts/STokenInterfaces.sol#116) is never used in SErc20Delegator (contracts/SErc20Delegator.sol#10-477)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

SErc20Storage.underlying (contracts/STokenInterfaces.sol#258) should be constant
STokenStorage._notEntered (contracts/STokenInterfaces.sol#10) should be constant
STokenStorage.accrualBlockNumber (contracts/STokenInterfaces.sol#71) should be constant
STokenStorage.borrowIndex (contracts/STokenInterfaces.sol#76) should be constant
STokenStorage.decimals (contracts/STokenInterfaces.sol#25) should be constant
STokenStorage.initialExchangeRateMantissa (contracts/STokenInterfaces.sol#61) should be constant
STokenStorage.name (contracts/STokenInterfaces.sol#15) should be constant
STokenStorage.pendingAdmin (contracts/STokenInterfaces.sol#46) should be constant
STokenStorage.reserveFactorMantissa (contracts/STokenInterfaces.sol#66) should be constant
STokenStorage.symbol (contracts/STokenInterfaces.sol#20) should be constant
STokenStorage.totalBorrows (contracts/STokenInterfaces.sol#81) should be constant
STokenStorage.totalReserves (contracts/STokenInterfaces.sol#86) should be constant
STokenStorage.totalSupply (contracts/STokenInterfaces.sol#91) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

borrowBalanceStored(address) should be declared external:
        - SErc20Delegator.borrowBalanceStored(address) (contracts/SErc20Delegator.sol#273-276)
        - STokenInterface.borrowBalanceStored(address) (contracts/STokenInterfaces.sol#236)
exchangeRateCurrent() should be declared external:
        - SErc20Delegator.exchangeRateCurrent() (contracts/SErc20Delegator.sol#282-285)
        - STokenInterface.exchangeRateCurrent() (contracts/STokenInterfaces.sol#237)
exchangeRateStored() should be declared external:
        - SErc20Delegator.exchangeRateStored() (contracts/SErc20Delegator.sol#292-295)
        - STokenInterface.exchangeRateStored() (contracts/STokenInterfaces.sol#238)
accrueInterest() should be declared external:
        - SErc20Delegator.accrueInterest() (contracts/SErc20Delegator.sol#311-314)
        - STokenInterface.accrueInterest() (contracts/STokenInterfaces.sol#240)
_setComptroller(ComptrollerInterface) should be declared external:
        - SErc20Delegator._setComptroller(ComptrollerInterface) (contracts/SErc20Delegator.sol#348-351)
        - STokenInterface._setComptroller(ComptrollerInterface) (contracts/STokenInterfaces.sol#248)
_setInterestRateModel(InterestRateModel) should be declared external:
        - SErc20Delegator._setInterestRateModel(InterestRateModel) (contracts/SErc20Delegator.sol#409-412)
        - STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

SErc20Immutable.sol

```
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#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/SToken.sol#390)
SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - require(bool,string)(mathErr == MathError.NO_ERROR,could not calculate block delta) (contracts/SToken.sol#406)
SToken.balanceOfUnderlying(address) (contracts/SToken.sol#190-195) uses a dangerous strict equality:
        - require(bool,string)(mErr == MathError.NO_ERROR,balance could not be calculated) (contracts/SToken.sol#193)
SToken.borrowBalanceStored(address) (contracts/SToken.sol#271-275) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,borrowBalanceStored: borrowBalanceStoredInternal failed) (contracts/SToken.sol#273)
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) uses a dangerous strict equality:
        - b == 0 (contracts/CarefulMath.sol#42)
SToken.exchangeRateStored() (contracts/SToken.sol#328-332) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,exchangeRateStored: exchangeRateStoredInternal failed) (contracts/SToken.sol#330)
SToken.exchangeRateStoredInternal() (contracts/SToken.sol#339-369) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/SToken.sol#341)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/SToken.sol#33)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting comptroller failed) (contracts/SToken.sol#41)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting interest rate model failed) (contracts/SToken.sol#49)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == uint256(Error.NO_ERROR),LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/SToken.sol#997)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(seizeError == uint256(Error.NO_ERROR),token seizure failed) (contracts/SToken.sol#1011)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_EXCHANGE_CALCULATION_FAILED) (contracts/SToken.sol#536)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_TOTAL_SUPPLY_CALCULATION_FAILED) (contracts/SToken.sol#544)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#547)
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) uses a dangerous strict equality:
        - assert(bool)(err2 == MathError.NO_ERROR) (contracts/Exponential.sol#163)
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) uses a dangerous strict equality:
        - a == 0 (contracts/CarefulMath.sol#25)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - a == 0 || b == 0 (contracts/Exponential.sol#306)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - require(bool,string)(c / a == b,errorMessage) (contracts/Exponential.sol#310)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#899)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#902)
SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,exchange rate math error) (contracts/SToken.sol#1089)
```

```
SToken.transfer(address,uint256) (contracts/SToken.sol#135-137) uses a dangerous strict equality:
        - transferTokens(msg.sender,msg.sender,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#136)
SToken.transferFrom(address,address,uint256) (contracts/SToken.sol#146-148) uses a dangerous strict equality:
        - transferTokens(msg.sender,src,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#696)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

SToken.redeemFresh(address,uint256,uint256).vars (contracts/SToken.sol#616) is a local variable never initialized
SToken._addReservesFresh(uint256).actualAddAmount (contracts/SToken.sol#1274) is a local variable never initialized
SToken.mintFresh(address,uint256).vars (contracts/SToken.sol#509) is a local variable never initialized
SToken.borrowFresh(address,uint256).vars (contracts/SToken.sol#753) is a local variable never initialized
SToken.repayBorrowFresh(address,address,uint256).vars (contracts/SToken.sol#862) is a local variable never initialized
SToken.seizeInternal(address,address,address,uint256).vars (contracts/SToken.sol#1073) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

SErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SErc20.sol#21-34) ignores return value by EIP20Interface(underlying).totalSupply() (contracts/SErc20.sol#33)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Exponential.divScalarByExpTruncate(uint256,Exponential.Exp).fraction (contracts/Exponential.sol#135) shadows:
        - Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

SErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8).underlying_ (contracts/SErc20.sol#21) lacks a zero-check on :
        - underlying = underlying_ (contracts/SErc20.sol#32)
SToken._setPendingAdmin(address).newPendingAdmin (contracts/SToken.sol#1135) lacks a zero-check on :
        - pendingAdmin = newPendingAdmin (contracts/SToken.sol#1145)
SErc20Immutable.constructor(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8,address).admin_ (contracts/SErc20Immutable.sol#29) lacks a zero-check on :
        - admin = admin_ (contracts/SErc20Immutable.sol#37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#788)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#789)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#790)
Reentrancy in SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/SToken.sol#499)
        State variables written after the call(s):
        - accountTokens[minter] = vars.accountTokensNew (contracts/SToken.sol#551)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#550)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - accountTokens[redeemer] = vars.accountTokensNew (contracts/SToken.sol#697)
Reentrancy in SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#905)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#906)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
Reentrancy in SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
        State variables written after the call(s):
        - accountTokens[borrower] = vars.borrowerTokensNew (contracts/SToken.sol#1112)
        - accountTokens[liquidator] = vars.liquidatorTokensNew (contracts/SToken.sol#1113)
        - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#1111)
Reentrancy in SToken.transferTokens(address,address,address,uint256) (contracts/SToken.sol#68-127):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/SToken.sol#70)
        State variables written after the call(s):
        - accountTokens[src] = srcTokensNew (contracts/SToken.sol#113)
        - accountTokens[dst] = dstTokensNew (contracts/SToken.sol#114)
        - transferAllowances[src][spender] = allowanceNew (contracts/SToken.sol#118)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        Event emitted after the call(s):
        - Borrow(borrower,borrowAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#793)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#762)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.BORROW_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#740)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#772)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.BORROW_FRESHNESS_CHECK) (contracts/SToken.sol#745)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#767)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.BORROW_CASH_NOT_AVAILABLE) (contracts/SToken.sol#750)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.LIQUIDATE_FRESHNESS_CHECK),0) (contracts/SToken.sol#961)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED,FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_ZERO),0) (contracts/SToken.sol#976)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.LIQUIDATE_COLLATERAL_FRESHNESS_CHECK),0) (contracts/SToken.sol#966)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.LIQUIDATE_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#956)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - (fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED,FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_UINT_MAX),0) (contracts/SToken.sol#981)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_ACCOUNT_PAIR,FailureInfo.LIQUIDATE_LIQUIDATOR_IS_BORROWER),0) (contracts/SToken.sol#971)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - (fail(TokenErrorReporter.Error(repayBorrowError),FailureInfo.LIQUIDATE_REPAY_BORROW_FRESH_FAILED),0) (contracts/SToken.sol#988)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
```

```
            - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
            - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
            - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
            - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
            - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
            - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
            - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
        - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(sTokenCollateral),seizeTokens) (contracts/SToken.sol#1014)
Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
            - (fail(TokenErrorReporter.Error(error),FailureInfo.LIQUIDATE_ACCRUE_COLLATERAL_INTEREST_FAILED),0) (contracts/SToken.sol#936)
Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
            - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
            - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
            - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
            - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
            - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
            - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
            - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(sTokenCollateral),seizeTokens) (contracts/SToken.sol#1014)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
Reentrancy in SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/SToken.sol#499)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
            - (fail(Error.MARKET_NOT_FRESH,FailureInfo.MINT_FRESHNESS_CHECK),0) (contracts/SToken.sol#506)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
```

SEther.sol

```
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#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/SToken.sol#390)
SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - require(bool,string)(mathErr == MathError.NO_ERROR,could not calculate block delta) (contracts/SToken.sol#406)
SToken.balanceOfUnderlying(address) (contracts/SToken.sol#190-195) uses a dangerous strict equality:
        - require(bool,string)(mErr == MathError.NO_ERROR,balance could not be calculated) (contracts/SToken.sol#193)
SToken.borrowBalanceStored(address) (contracts/SToken.sol#271-275) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,borrowBalanceStored: borrowBalanceStoredInternal failed) (contracts/SToken.sol#273)
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) uses a dangerous strict equality:
        - b == 0 (contracts/CarefulMath.sol#42)
SEther.doTransferIn(address,uint256) (contracts/SEther.sol#136-141) uses a dangerous strict equality:
        - require(bool,string)(msg.value == amount,value mismatch) (contracts/SEther.sol#139)
SToken.exchangeRateStored() (contracts/SToken.sol#328-332) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,exchangeRateStored: exchangeRateStoredInternal failed) (contracts/SToken.sol#330)
SToken.exchangeRateStoredInternal() (contracts/SToken.sol#339-369) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/SToken.sol#341)
SEther.getCashPrior() (contracts/SEther.sol#124-128) uses a dangerous strict equality:
        - require(bool)(err == MathError.NO_ERROR) (contracts/SEther.sol#126)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/SToken.sol#33)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting comptroller failed) (contracts/SToken.sol#41)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting interest rate model failed) (contracts/SToken.sol#49)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == uint256(Error.NO_ERROR),LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/SToken.sol#997)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(seizeError == uint256(Error.NO_ERROR),token seizure failed) (contracts/SToken.sol#1011)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_EXCHANGE_CALCULATION_FAILED) (contracts/SToken.sol#536)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_TOTAL_SUPPLY_CALCULATION_FAILED) (contracts/SToken.sol#544)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#547)
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) uses a dangerous strict equality:
        - assert(bool)(err2 == MathError.NO_ERROR) (contracts/Exponential.sol#163)
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) uses a dangerous strict equality:
        - a == 0 (contracts/CarefulMath.sol#25)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - a == 0 || b == 0 (contracts/Exponential.sol#306)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - require(bool,string)(c / a == b,errorMessage) (contracts/Exponential.sol#310)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#899)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#902)
SEther.requireNoError(uint256,string) (contracts/SEther.sol#148-167) uses a dangerous strict equality:
        - errCode == uint256(Error.NO_ERROR) (contracts/SEther.sol#149)
SEther.requireNoError(uint256,string) (contracts/SEther.sol#148-167) uses a dangerous strict equality:
        - require(bool,string)(errCode == uint256(Error.NO_ERROR),string(fullMessage)) (contracts/SEther.sol#166)
SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,exchange rate math error) (contracts/SToken.sol#1089)
SToken.transfer(address,uint256) (contracts/SToken.sol#135-137) uses a dangerous strict equality:
        - transferTokens(msg.sender,msg.sender,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#136)
SToken.transferFrom(address,address,uint256) (contracts/SToken.sol#146-148) uses a dangerous strict equality:
        - transferTokens(msg.sender,src,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
            - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
            - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
            - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
            - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
            - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
            - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
            - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
            - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
            - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#696)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

SToken.repayBorrowFresh(address,address,uint256).vars (contracts/SToken.sol#862) is a local variable never initialized
SToken.seizeInternal(address,address,address,uint256).vars (contracts/SToken.sol#1073) is a local variable never initialized
SToken.borrowFresh(address,uint256).vars (contracts/SToken.sol#753) is a local variable never initialized
SToken.redeemFresh(address,uint256,uint256).vars (contracts/SToken.sol#616) is a local variable never initialized
SToken.mintFresh(address,uint256).vars (contracts/SToken.sol#509) is a local variable never initialized
SToken._addReservesFresh(uint256).actualAddAmount (contracts/SToken.sol#1274) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Exponential.divScalarByExpTruncate(uint256,Exponential.Exp).fraction (contracts/Exponential.sol#135) shadows:
        - Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
```

SimplePriceOracle.sol

```
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#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/SToken.sol#390)
SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - require(bool,string)(mathErr == MathError.NO_ERROR,could not calculate block delta) (contracts/SToken.sol#406)
SToken.balanceOfUnderlying(address) (contracts/SToken.sol#190-195) uses a dangerous strict equality:
        - require(bool,string)(mErr == MathError.NO_ERROR,balance could not be calculated) (contracts/SToken.sol#193)
SToken.borrowBalanceStored(address) (contracts/SToken.sol#271-275) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,borrowBalanceStored: borrowBalanceStoredInternal failed) (contracts/SToken.sol#273)
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) uses a dangerous strict equality:
        - b == 0 (contracts/CarefulMath.sol#42)
SToken.exchangeRateStored() (contracts/SToken.sol#328-332) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,exchangeRateStored: exchangeRateStoredInternal failed) (contracts/SToken.sol#330)
SToken.exchangeRateStoredInternal() (contracts/SToken.sol#339-369) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/SToken.sol#341)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/SToken.sol#33)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting comptroller failed) (contracts/SToken.sol#41)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting interest rate model failed) (contracts/SToken.sol#49)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == uint256(Error.NO_ERROR),LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/SToken.sol#997)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(seizeError == uint256(Error.NO_ERROR),token seizure failed) (contracts/SToken.sol#1011)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_EXCHANGE_CALCULATION_FAILED) (contracts/SToken.sol#536)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_TOTAL_SUPPLY_CALCULATION_FAILED) (contracts/SToken.sol#544)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#547)
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) uses a dangerous strict equality:
        - assert(bool)(err2 == MathError.NO_ERROR) (contracts/Exponential.sol#163)
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) uses a dangerous strict equality:
        - a == 0 (contracts/CarefulMath.sol#25)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - a == 0 || b == 0 (contracts/Exponential.sol#306)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - require(bool,string)(c / a == b,errorMessage) (contracts/Exponential.sol#310)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#899)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#902)
SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,exchange rate math error) (contracts/SToken.sol#1089)
SToken.transfer(address,uint256) (contracts/SToken.sol#135-137) uses a dangerous strict equality:
        - transferTokens(msg.sender,msg.sender,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#136)
SToken.transferFrom(address,address,uint256) (contracts/SToken.sol#146-148) uses a dangerous strict equality:
        - transferTokens(msg.sender,src,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#696)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

SToken.redeemFresh(address,uint256,uint256).vars (contracts/SToken.sol#616) is a local variable never initialized
SToken.borrowFresh(address,uint256).vars (contracts/SToken.sol#753) is a local variable never initialized
SToken._addReservesFresh(uint256).actualAddAmount (contracts/SToken.sol#1274) is a local variable never initialized
SToken.seizeInternal(address,address,address,uint256).vars (contracts/SToken.sol#1073) is a local variable never initialized
SToken.mintFresh(address,uint256).vars (contracts/SToken.sol#509) is a local variable never initialized
SToken.repayBorrowFresh(address,address,uint256).vars (contracts/SToken.sol#862) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

SErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SErc20.sol#21-34) ignores return value by EIP20Interface(underlying).totalSupply() (contracts/SErc20.sol#33)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Exponential.divScalarByExpTruncate(uint256,Exponential.Exp).fraction (contracts/Exponential.sol#135) shadows:
        - Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

SToken._setPendingAdmin(address).newPendingAdmin (contracts/SToken.sol#1135) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (contracts/SToken.sol#1145)
SErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8).underlying_ (contracts/SErc20.sol#21) lacks a zero-check on :
                - underlying = underlying_ (contracts/SErc20.sol#32)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#788)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#789)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#790)
Reentrancy in SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/SToken.sol#499)
        State variables written after the call(s):
        - accountTokens[minter] = vars.accountTokensNew (contracts/SToken.sol#551)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#550)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - accountTokens[redeemer] = vars.accountTokensNew (contracts/SToken.sol#697)
Reentrancy in SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#905)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#906)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
Reentrancy in SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
        State variables written after the call(s):
        - accountTokens[borrower] = vars.borrowerTokensNew (contracts/SToken.sol#1112)
        - accountTokens[liquidator] = vars.liquidatorTokensNew (contracts/SToken.sol#1113)
        - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#1111)
Reentrancy in SToken.transferTokens(address,address,address,uint256) (contracts/SToken.sol#68-127):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/SToken.sol#70)
        State variables written after the call(s):
        - accountTokens[src] = srsTokensNew (contracts/SToken.sol#113)
        - accountTokens[dst] = dstTokensNew (contracts/SToken.sol#114)
        - transferAllowances[src][spender] = allowanceNew (contracts/SToken.sol#118)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        Event emitted after the call(s):
        - Borrow(borrower,borrowAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#793)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#762)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.BORROW_FRESHNESS_CHECK) (contracts/SToken.sol#745)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.BORROW_CASH_NOT_AVAILABLE) (contracts/SToken.sol#750)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#767)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.COMPTROLLER_REJECTION,FailureInfo.BORROW_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#740)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#772)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_ACCOUNT_PAIR,FailureInfo.LIQUIDATE_LIQUIDATOR_IS_BORROWER),0) (contracts/SToken.sol#971)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.COMPTROLLER_REJECTION,FailureInfo.LIQUIDATE_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#956)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED,FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_UINT_MAX),0) (contracts/SToken.sol#981)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.LIQUIDATE_FRESHNESS_CHECK),0) (contracts/SToken.sol#961)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.LIQUIDATE_COLLATERAL_FRESHNESS_CHECK),0) (contracts/SToken.sol#966)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED,FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_ZERO),0) (contracts/SToken.sol#976)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(TokenErrorReporter.Error(repayBorrowError),FailureInfo.LIQUIDATE_REPAY_BORROW_FRESH_FAILED),0) (contracts/SToken.sol#988)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
        - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(sTokenCollateral),seizeTokens) (contracts/SToken.sol#1014)
Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(TokenErrorReporter.Error(error),FailureInfo.LIQUIDATE_ACCRUE_COLLATERAL_INTEREST_FAILED),0) (contracts/SToken.sol#936)
Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(sTokenCollateral),seizeTokens) (contracts/SToken.sol#1014)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
Reentrancy in SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/SToken.sol#499)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.COMPTROLLER_REJECTION,FailureInfo.MINT_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#501)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.MINT_FRESHNESS_CHECK),0) (contracts/SToken.sol#506)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.MINT_EXCHANGE_RATE_READ_FAILED,uint256(vars.mathErr)),0) (contracts/SToken.sol#513)
        - Mint(minter,vars.actualMintAmount,vars.mintTokens) (contracts/SToken.sol#554)
        - Transfer(address(this),minter,vars.mintTokens) (contracts/SToken.sol#555)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.COMPTROLLER_REJECTION,FailureInfo.REDEEM_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#655)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.REDEEM_FRESHNESS_CHECK) (contracts/SToken.sol#660)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.REDEEM_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#675)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.REDEEM_NEW_TOTAL_SUPPLY_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#670)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.REDEEM_TRANSFER_OUT_NOT_POSSIBLE) (contracts/SToken.sol#680)
        - Redeem(redeemer,vars.redeemAmount,vars.redeemTokens) (contracts/SToken.sol#701)
        - Transfer(redeemer,address(this),vars.redeemTokens) (contracts/SToken.sol#700)
Reentrancy in SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.REPAY_BORROW_FRESHNESS_CHECK),0) (contracts/SToken.sol#859)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.COMPTROLLER_REJECTION,FailureInfo.REPAY_BORROW_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#854)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.REPAY_BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)),0) (contracts/SToken.sol#870)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
```

```
Reentrancy in SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_ACCOUNT_PAIR,FailureInfo.LIQUIDATE_SEIZE_LIQUIDATOR_IS_BORROWER) (contracts/SToken.sol#1066)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_INCREMENT_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#1102)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_DECREMENT_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#1082)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.LIQUIDATE_SEIZE_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#1061)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_DECREMENT_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#1093)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
Reentrancy in SToken.transferTokens(address,address,address,uint256) (contracts/SToken.sol#68-127):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/SToken.sol#70)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.TRANSFER_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#72)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/SToken.sol#96)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.BAD_INPUT,FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/SToken.sol#77)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_NOT_ENOUGH) (contracts/SToken.sol#101)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_TOO_MUCH) (contracts/SToken.sol#106)
        - Transfer(src,dst,tokens) (contracts/SToken.sol#122)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

SErc20.doTransferIn(address,uint256) (contracts/SErc20.sol#142-167) uses assembly
        - INLINE ASM (contracts/SErc20.sol#148-160)
SErc20.doTransferOut(address,uint256) (contracts/SErc20.sol#178-197) uses assembly
        - INLINE ASM (contracts/SErc20.sol#183-195)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) is never used and should be removed
ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) is never used and should be removed
Exponential.addExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#48-52) is never used and should be removed
Exponential.add_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#246-248) is never used and should be removed
Exponential.add_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#242-244) is never used and should be removed
Exponential.divExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#191-193) is never used and should be removed
Exponential.divScalar(Exponential.Exp,uint256) (contracts/Exponential.sol#102-109) is never used and should be removed
Exponential.div_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#326-328) is never used and should be removed
Exponential.div_(Exponential.Double,uint256) (contracts/Exponential.sol#330-332) is never used and should be removed
Exponential.div_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#314-316) is never used and should be removed
Exponential.div_(Exponential.Exp,uint256) (contracts/Exponential.sol#318-320) is never used and should be removed
Exponential.div_(uint256,Exponential.Double) (contracts/Exponential.sol#334-336) is never used and should be removed
Exponential.div_(uint256,Exponential.Exp) (contracts/Exponential.sol#322-324) is never used and should be removed
Exponential.div_(uint256,uint256) (contracts/Exponential.sol#338-340) is never used and should be removed
Exponential.div_(uint256,uint256,string) (contracts/Exponential.sol#342-345) is never used and should be removed
Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) is never used and should be removed
Exponential.greaterThanExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#221-223) is never used and should be removed
Exponential.isZeroExp(Exponential.Exp) (contracts/Exponential.sol#228-230) is never used and should be removed
Exponential.lessThanExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#207-209) is never used and should be removed
Exponential.lessThanOrEqualExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#214-216) is never used and should be removed
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) is never used and should be removed
Exponential.mulExp(uint256,uint256) (contracts/Exponential.sol#171-173) is never used and should be removed
Exponential.mulExp3(Exponential.Exp,Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#178-184) is never used and should be removed
Exponential.mul_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#289-291) is never used and should be removed
Exponential.mul_(Exponential.Double,uint256) (contracts/Exponential.sol#293-295) is never used and should be removed
Exponential.mul_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#277-279) is never used and should be removed
Exponential.mul_(Exponential.Exp,uint256) (contracts/Exponential.sol#281-283) is never used and should be removed
Exponential.mul_(uint256,Exponential.Double) (contracts/Exponential.sol#297-299) is never used and should be removed
Exponential.safe224(uint256,string) (contracts/Exponential.sol#232-235) is never used and should be removed
Exponential.safe32(uint256,string) (contracts/Exponential.sol#237-240) is never used and should be removed
Exponential.subExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#57-61) is never used and should be removed
Exponential.sub_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#264-266) is never used and should be removed
Exponential.sub_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#260-262) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Constant ComptrollerInterface.isComptroller (contracts/ComptrollerInterface.sol#5) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.expScale (contracts/Exponential.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.doubleScale (contracts/Exponential.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.halfExpScale (contracts/Exponential.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.mantissaOne (contracts/Exponential.sol#16) 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
Function SErc20._addReserves(uint256) (contracts/SErc20.sol#117-119) is not in mixedCase
Function SToken._setPendingAdmin(address) (contracts/SToken.sol#1135-1151) is not in mixedCase
Function SToken._acceptAdmin() (contracts/SToken.sol#1158-1178) is not in mixedCase
Function SToken._setComptroller(ComptrollerInterface) (contracts/SToken.sol#1185-1202) is not in mixedCase
Function SToken._setReserveFactor(uint256) (contracts/SToken.sol#1209-1217) is not in mixedCase
Function SToken._reduceReserves(uint256) (contracts/SToken.sol#1316-1324) is not in mixedCase
Function SToken._transferReserves(uint256) (contracts/SToken.sol#1380-1388) is not in mixedCase
Function SToken._setInterestRateModel(InterestRateModel) (contracts/SToken.sol#1445-1453) is not in mixedCase
Variable STokenStorage._notEntered (contracts/STokenInterfaces.sol#10) is not in mixedCase
Constant STokenStorage.borrowRateMaxMantissa (contracts/STokenInterfaces.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.reserveFactorMaxMantissa (contracts/STokenInterfaces.sol#36) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.protocolSeizeShareMantissa (contracts/STokenInterfaces.sol#121) is not in UPPER_CASE_WITH_UNDERSCORES
Function STokenInterface._setPendingAdmin(address) (contracts/STokenInterfaces.sol#206) is not in mixedCase
Function STokenInterface._acceptAdmin() (contracts/STokenInterfaces.sol#247) is not in mixedCase
Function STokenInterface._setComptroller(ComptrollerInterface) (contracts/STokenInterfaces.sol#248) is not in mixedCase
Function STokenInterface._setReserveFactor(uint256) (contracts/STokenInterfaces.sol#249) is not in mixedCase
Function STokenInterface._reduceReserves(uint256) (contracts/STokenInterfaces.sol#250) is not in mixedCase
Function STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251) is not in mixedCase
Constant STokenInterface.isSToken (contracts/STokenInterfaces.sol#129) is not in UPPER_CASE_WITH_UNDERSCORES
Function SErc20Interface._addReserves(uint256) (contracts/STokenInterfaces.sol#276) is not in mixedCase
Function SDelegatorInterface._setImplementation(address,bool,bytes) (contracts/STokenInterfaces.sol#298) is not in mixedCase
Function SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307) is not in mixedCase
Function SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable SToken.seizeInternal(address,address,address,uint256).seizeTokens (contracts/SToken.sol#1057) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable SToken.seize(address,address,uint256).seizeTokens (contracts/SToken.sol#1031) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface).seizeTokens (contracts/SToken.sol#996) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable STokenInterface.seize(address,address,uint256).seizeTokens (contracts/STokenInterfaces.sol#241) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

SDelegationStorage.implementation (contracts/STokenInterfaces.sol#283) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - SErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SErc20.sol#21-34)
_setInterestRateModel(InterestRateModel) should be declared external:
        - SToken._setInterestRateModel(InterestRateModel) (contracts/SToken.sol#1445-1453)
        - STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (contracts/STokenInterfaces.sol#298)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312)
getUnderlyingPrice(SToken) should be declared external:
        - SimplePriceOracle.getUnderlyingPrice(SToken) (contracts/SimplePriceOracle.sol#10-16)
setUnderlyingPrice(SToken,uint256) should be declared external:
        - SimplePriceOracle.setUnderlyingPrice(SToken,uint256) (contracts/SimplePriceOracle.sol#18-22)
setDirectPrice(address,uint256) should be declared external:
        - SimplePriceOracle.setDirectPrice(address,uint256) (contracts/SimplePriceOracle.sol#24-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

SStrkLikeDelegate.sol

```
SErc20Delegate._becomeImplementation(bytes) (contracts/SErc20Delegate.sol#20-30) uses a Boolean constant improperly:
        -false (contracts/SErc20Delegate.sol#25)
SErc20Delegate._resignImplementation() (contracts/SErc20Delegate.sol#35-42) uses a Boolean constant improperly:
        -false (contracts/SErc20Delegate.sol#37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#misuse-of-a-boolean-constant

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#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface
```

```
SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/SToken.sol#390)
SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - require(bool,string)(mathErr == MathError.NO_ERROR,could not calculate block delta) (contracts/SToken.sol#406)
SToken.balanceOfUnderlying(address) (contracts/SToken.sol#190-195) uses a dangerous strict equality:
        - require(bool,string)(mErr == MathError.NO_ERROR,balance could not be calculated) (contracts/SToken.sol#193)
SToken.borrowBalanceStored(address) (contracts/SToken.sol#271-275) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,borrowBalanceStored: borrowBalanceStoredInternal failed) (contracts/SToken.sol#273)
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) uses a dangerous strict equality:
        - b == 0 (contracts/CarefulMath.sol#42)
SToken.exchangeRateStored() (contracts/SToken.sol#328-332) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,exchangeRateStored: exchangeRateStoredInternal failed) (contracts/SToken.sol#330)
SToken.exchangeRateStoredInternal() (contracts/SToken.sol#339-369) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/SToken.sol#341)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/SToken.sol#33)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting comptroller failed) (contracts/SToken.sol#41)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting interest rate model failed) (contracts/SToken.sol#49)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == uint256(Error.NO_ERROR),LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/SToken.sol#997)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(seizeError == uint256(Error.NO_ERROR),token seizure failed) (contracts/SToken.sol#1011)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_EXCHANGE_CALCULATION_FAILED) (contracts/SToken.sol#536)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_TOTAL_SUPPLY_CALCULATION_FAILED) (contracts/SToken.sol#544)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#547)
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) uses a dangerous strict equality:
        - assert(bool)(err2 == MathError.NO_ERROR) (contracts/Exponential.sol#163)
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) uses a dangerous strict equality:
        - a == 0 (contracts/CarefulMath.sol#25)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - a == 0 || b == 0 (contracts/Exponential.sol#306)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - require(bool,string)(c / a == b,errorMessage) (contracts/Exponential.sol#310)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#899)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#902)
SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,exchange rate math error) (contracts/SToken.sol#1089)
SToken.transfer(address,uint256) (contracts/SToken.sol#135-137) uses a dangerous strict equality:
        - transferTokens(msg.sender,msg.sender,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#136)
SToken.transferFrom(address,address,uint256) (contracts/SToken.sol#146-148) uses a dangerous strict equality:
        - transferTokens(msg.sender,src,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#696)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

SToken._addReservesFresh(uint256).actualAddAmount (contracts/SToken.sol#1274) is a local variable never initialized
SToken.mintFresh(address,uint256).vars (contracts/SToken.sol#509) is a local variable never initialized
SToken.borrowFresh(address,uint256).vars (contracts/SToken.sol#753) is a local variable never initialized
SToken.repayBorrowFresh(address,address,uint256).vars (contracts/SToken.sol#862) is a local variable never initialized
SToken.redeemFresh(address,uint256,uint256).vars (contracts/SToken.sol#616) is a local variable never initialized
SToken.seizeInternal(address,address,address,uint256).vars (contracts/SToken.sol#1073) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

SErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SErc20.sol#21-34) ignores return value by EIP20Interface(underlying).totalSupply() (contracts/SErc20.sol#33)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Exponential.divScalarByExpTruncate(uint256,Exponential.Exp).fraction (contracts/Exponential.sol#135) shadows:
        - Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

SErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8).underlying_ (contracts/SErc20.sol#21) lacks a zero-check on :
        - underlying = underlying_ (contracts/SErc20.sol#32)
SToken._setPendingAdmin(address).newPendingAdmin (contracts/SToken.sol#1135) lacks a zero-check on :
        - pendingAdmin = newPendingAdmin (contracts/SToken.sol#1145)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#788)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#789)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#790)
Reentrancy in SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/SToken.sol#499)
        State variables written after the call(s):
        - accountTokens[minter] = vars.accountTokensNew (contracts/SToken.sol#551)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#550)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
```

SToken.sol

```
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#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/SToken.sol#390)
SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - require(bool,string)(mathErr == MathError.NO_ERROR,could not calculate block delta) (contracts/SToken.sol#406)
SToken.balanceOfUnderlying(address) (contracts/SToken.sol#190-195) uses a dangerous strict equality:
        - require(bool,string)(mErr == MathError.NO_ERROR,balance could not be calculated) (contracts/SToken.sol#193)
SToken.borrowBalanceStored(address) (contracts/SToken.sol#271-275) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,borrowBalanceStored: borrowBalanceStoredInternal failed) (contracts/SToken.sol#273)
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) uses a dangerous strict equality:
        - b == 0 (contracts/CarefulMath.sol#42)
SToken.exchangeRateStored() (contracts/SToken.sol#328-332) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,exchangeRateStored: exchangeRateStoredInternal failed) (contracts/SToken.sol#330)
SToken.exchangeRateStoredInternal() (contracts/SToken.sol#339-369) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/SToken.sol#341)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/SToken.sol#33)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting comptroller failed) (contracts/SToken.sol#41)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting interest rate model failed) (contracts/SToken.sol#49)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == uint256(Error.NO_ERROR),LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/SToken.sol#997)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(seizeError == uint256(Error.NO_ERROR),token seizure failed) (contracts/SToken.sol#1011)
```

```
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_EXCHANGE_CALCULATION_FAILED) (contracts/SToken.sol#536)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_TOTAL_SUPPLY_CALCULATION_FAILED) (contracts/SToken.sol#544)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#547)
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) uses a dangerous strict equality:
        - assert(bool)(err2 == MathError.NO_ERROR) (contracts/Exponential.sol#163)
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) uses a dangerous strict equality:
        - a == 0 (contracts/CarefulMath.sol#25)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - a == 0 || b == 0 (contracts/Exponential.sol#306)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - require(bool,string)(c / a == b,errorMessage) (contracts/Exponential.sol#310)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#899)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#902)
SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,exchange rate math error) (contracts/SToken.sol#1089)
SToken.transfer(address,uint256) (contracts/SToken.sol#135-137) uses a dangerous strict equality:
        - transferTokens(msg.sender,msg.sender,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#136)
SToken.transferFrom(address,address,uint256) (contracts/SToken.sol#146-148) uses a dangerous strict equality:
        - transferTokens(msg.sender,src,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1087)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#696)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

SToken.mintFresh(address,uint256).vars (contracts/SToken.sol#509) is a local variable never initialized
SToken._addReservesFresh(uint256).actualAddAmount (contracts/SToken.sol#1274) is a local variable never initialized
SToken.repayBorrowFresh(address,address,uint256).vars (contracts/SToken.sol#862) is a local variable never initialized
SToken.borrowFresh(address,uint256).vars (contracts/SToken.sol#753) is a local variable never initialized
SToken.redeemFresh(address,uint256,uint256).vars (contracts/SToken.sol#616) is a local variable never initialized
SToken.seizeInternal(address,address,address,uint256).vars (contracts/SToken.sol#1073) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Exponential.divScalarByExpTruncate(uint256,Exponential.Exp).fraction (contracts/Exponential.sol#135) shadows:
        - Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) should emit an event for:
        - initialExchangeRateMantissa = initialExchangeRateMantissa_ (contracts/SToken.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

SToken._setPendingAdmin(address).newPendingAdmin (contracts/SToken.sol#1135) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (contracts/SToken.sol#1145)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#788)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#789)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#790)
Reentrancy in SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/SToken.sol#499)
        State variables written after the call(s):
        - accountTokens[minter] = vars.accountTokensNew (contracts/SToken.sol#551)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#550)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - accountTokens[redeemer] = vars.accountTokensNew (contracts/SToken.sol#697)
Reentrancy in SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#905)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#906)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
Reentrancy in SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
        State variables written after the call(s):
        - accountTokens[borrower] = vars.borrowerTokensNew (contracts/SToken.sol#1112)
        - accountTokens[liquidator] = vars.liquidatorTokensNew (contracts/SToken.sol#1113)
        - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#1111)
Reentrancy in SToken.transferTokens(address,address,address,uint256) (contracts/SToken.sol#68-127):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/SToken.sol#70)
        State variables written after the call(s):
        - accountTokens[src] = srcTokensNew (contracts/SToken.sol#113)
        - accountTokens[dst] = dstTokensNew (contracts/SToken.sol#114)
        - transferAllowances[src][spender] = allowanceNew (contracts/SToken.sol#118)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        Event emitted after the call(s):
        - Borrow(borrower,borrowAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#793)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.BORROW_FRESHNESS_CHECK) (contracts/SToken.sol#745)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#762)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#767)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#772)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.BORROW_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#740)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.BORROW_CASH_NOT_AVAILABLE) (contracts/SToken.sol#750)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_ACCOUNT_PAIR,FailureInfo.LIQUIDATE_LIQUIDATOR_IS_BORROWER),0) (contracts/SToken.sol#971)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED,FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_ZERO),0) (contracts/SToken.sol#976)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.LIQUIDATE_COLLATERAL_FRESHNESS_CHECK),0) (contracts/SToken.sol#966)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.LIQUIDATE_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#956)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED,FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_UINT_MAX),0) (contracts/SToken.sol#981)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.LIQUIDATE_FRESHNESS_CHECK),0) (contracts/SToken.sol#961)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(TokenErrorReporter.Error(repayBorrowError),FailureInfo.LIQUIDATE_REPAY_BORROW_FRESH_FAILED),0) (contracts/SToken.sol#988)
```

```
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
        - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(sTokenCollateral),seizeTokens) (contracts/SToken.sol#1014)
Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - (fail(TokenErrorReporter.Error(error),FailureInfo.LIQUIDATE_ACCRUE_COLLATERAL_INTEREST_FAILED),0) (contracts/SToken.sol#936)
Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(sTokenCollateral),seizeTokens) (contracts/SToken.sol#1014)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
Reentrancy in SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/SToken.sol#499)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - (fail0paque(Error.MATH_ERROR,FailureInfo.MINT_EXCHANGE_RATE_READ_FAILED,uint256(vars.mathErr)),0) (contracts/SToken.sol#513)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - (fail0paque(Error.COMPTROLLER_REJECTION,FailureInfo.MINT_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#501)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - (fail(Error.MARKET_NOT_FRESH,FailureInfo.MINT_FRESHNESS_CHECK),0) (contracts/SToken.sol#506)
        - Mint(minter,vars.actualMintAmount,vars.mintTokens) (contracts/SToken.sol#554)
        - Transfer(address(this),minter,vars.mintTokens) (contracts/SToken.sol#555)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.COMPTROLLER_REJECTION,FailureInfo.REDEEM_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#655)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.REDEEM_TRANSFER_OUT_NOT_POSSIBLE) (contracts/SToken.sol#680)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.REDEEM_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#675)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.REDEEM_NEW_TOTAL_SUPPLY_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#670)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.REDEEM_FRESHNESS_CHECK) (contracts/SToken.sol#660)
        - Redeem(redeemer,vars.redeemAmount,vars.redeemTokens) (contracts/SToken.sol#701)
        - Transfer(redeemer,address(this),vars.redeemTokens) (contracts/SToken.sol#700)
Reentrancy in SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - (fail0paque(Error.MATH_ERROR,FailureInfo.REPAY_BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)),0) (contracts/SToken.sol#870)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - (fail(Error.MARKET_NOT_FRESH,FailureInfo.REPAY_BORROW_FRESHNESS_CHECK),0) (contracts/SToken.sol#859)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - (fail0paque(Error.COMPTROLLER_REJECTION,FailureInfo.REPAY_BORROW_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#854)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
Reentrancy in SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_DECREMENT_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#1093)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_INCREMENT_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#1102)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_ACCOUNT_PAIR,FailureInfo.LIQUIDATE_SEIZE_LIQUIDATOR_IS_BORROWER) (contracts/SToken.sol#1066)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_DECREMENT_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#1082)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.COMPTROLLER_REJECTION,FailureInfo.LIQUIDATE_SEIZE_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#1061)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
Reentrancy in SToken.transferTokens(address,address,address,uint256) (contracts/SToken.sol#68-127):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/SToken.sol#70)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_NOT_ENOUGH) (contracts/SToken.sol#101)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/SToken.sol#96)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - fail0paque(Error.COMPTROLLER_REJECTION,FailureInfo.TRANSFER_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#72)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_TOO_MUCH) (contracts/SToken.sol#106)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.BAD_INPUT,FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/SToken.sol#77)
        - Transfer(src,dst,tokens) (contracts/SToken.sol#122)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) is never used and should be removed
ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) is never used and should be removed
Exponential.addExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#48-52) is never used and should be removed
Exponential.add_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#246-248) is never used and should be removed
Exponential.add_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#242-244) is never used and should be removed
Exponential.divExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#191-193) is never used and should be removed
Exponential.divScalar(Exponential.Exp,uint256) (contracts/Exponential.sol#102-109) is never used and should be removed
Exponential.divScalarByExp(uint256,Exponential.Exp) (contracts/Exponential.sol#114-129) is never used and should be removed
Exponential.divScalarByExpTruncate(uint256,Exponential.Exp) (contracts/Exponential.sol#134-141) is never used and should be removed
Exponential.div_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#326-328) is never used and should be removed
Exponential.div_(Exponential.Double,uint256) (contracts/Exponential.sol#330-332) is never used and should be removed
Exponential.div_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#314-316) is never used and should be removed
```

```
Exponential.div_(Exponential.Exp,uint256) (contracts/Exponential.sol#318-320) is never used and should be removed
Exponential.div_(uint256,Exponential.Double) (contracts/Exponential.sol#334-336) is never used and should be removed
Exponential.div_(uint256,Exponential.Exp) (contracts/Exponential.sol#322-324) is never used and should be removed
Exponential.div_(uint256,uint256) (contracts/Exponential.sol#338-340) is never used and should be removed
Exponential.div_(uint256,uint256,string) (contracts/Exponential.sol#342-345) is never used and should be removed
Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) is never used and should be removed
Exponential.greaterThanExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#221-223) is never used and should be removed
Exponential.isZeroExp(Exponential.Exp) (contracts/Exponential.sol#228-230) is never used and should be removed
Exponential.lessThanExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#207-209) is never used and should be removed
Exponential.lessThanOrEqualExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#214-216) is never used and should be removed
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) is never used and should be removed
Exponential.mulExp(uint256,uint256) (contracts/Exponential.sol#171-173) is never used and should be removed
Exponential.mulExp3(Exponential.Exp,Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#178-184) is never used and should be removed
Exponential.mul_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#289-291) is never used and should be removed
Exponential.mul_(Exponential.Double,uint256) (contracts/Exponential.sol#293-295) is never used and should be removed
Exponential.mul_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#277-279) is never used and should be removed
Exponential.mul_(Exponential.Exp,uint256) (contracts/Exponential.sol#281-283) is never used and should be removed
Exponential.mul_(uint256,Exponential.Double) (contracts/Exponential.sol#297-299) is never used and should be removed
Exponential.safe224(uint256,string) (contracts/Exponential.sol#232-235) is never used and should be removed
Exponential.safe32(uint256,string) (contracts/Exponential.sol#237-240) is never used and should be removed
Exponential.subExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#57-61) is never used and should be removed
Exponential.sub_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#264-266) is never used and should be removed
Exponential.sub_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#260-262) is never used and should be removed
SToken._addReservesFresh(uint256) (contracts/SToken.sol#1271-1308) is never used and should be removed
SToken._addReservesInternal(uint256) (contracts/SToken.sol#1253-1263) is never used and should be removed
SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799) is never used and should be removed
SToken.borrowInternal(uint256) (contracts/SToken.sol#714-722) is never used and should be removed
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#1092-1020) is never used and should be removed
SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941) is never used and should be removed
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) is never used and should be removed
SToken.mintInternal(uint256) (contracts/SToken.sol#470-478) is never used and should be removed
SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707) is never used and should be removed
SToken.redeemInternal(uint256) (contracts/SToken.sol#569-577) is never used and should be removed
SToken.redeemUnderlyingInternal(uint256) (contracts/SToken.sol#585-593) is never used and should be removed
SToken.repayBorrowBehalfInternal(address,uint256) (contracts/SToken.sol#822-830) is never used and should be removed
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) is never used and should be removed
SToken.repayBorrowInternal(uint256) (contracts/SToken.sol#806-814) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Constant ComptrollerInterface.isComptroller (contracts/ComptrollerInterface.sol#5) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.expScale (contracts/Exponential.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.doubleScale (contracts/Exponential.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.halfExpScale (contracts/Exponential.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.mantissaOne (contracts/Exponential.sol#16) is not in UPPER_CASE_WITH_UNDERSCORES
Constant InterestRateModel.isInterestRateModel (contracts/InterestRateModel.sol#9) is not in UPPER_CASE_WITH_UNDERSCORES
Function SToken._setPendingAdmin(address) (contracts/SToken.sol#1135-1151) is not in mixedCase
Function SToken._acceptAdmin() (contracts/SToken.sol#1158-1178) is not in mixedCase
Function SToken._setComptroller(ComptrollerInterface) (contracts/SToken.sol#1185-1202) is not in mixedCase
Function SToken._setReserveFactor(uint256) (contracts/SToken.sol#1209-1217) is not in mixedCase
Function SToken._reduceReserves(uint256) (contracts/SToken.sol#1316-1324) is not in mixedCase
Function SToken._transferReserves(uint256) (contracts/SToken.sol#1380-1388) is not in mixedCase
Function SToken._setInterestRateModel(InterestRateModel) (contracts/SToken.sol#1445-1453) is not in mixedCase
Variable STokenStorage._notEntered (contracts/STokenInterfaces.sol#10) is not in mixedCase
Constant STokenStorage.borrowRateMaxMantissa (contracts/STokenInterfaces.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.reserveFactorMaxMantissa (contracts/STokenInterfaces.sol#36) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.protocolSeizeShareMantissa (contracts/STokenInterfaces.sol#121) is not in UPPER_CASE_WITH_UNDERSCORES
Function STokenInterface._setPendingAdmin(address) (contracts/STokenInterfaces.sol#246) is not in mixedCase
Function STokenInterface._acceptAdmin() (contracts/STokenInterfaces.sol#247) is not in mixedCase
Function STokenInterface._setComptroller(ComptrollerInterface) (contracts/STokenInterfaces.sol#248) is not in mixedCase
Function STokenInterface._setReserveFactor(uint256) (contracts/STokenInterfaces.sol#249) is not in mixedCase
Function STokenInterface._reduceReserves(uint256) (contracts/STokenInterfaces.sol#250) is not in mixedCase
Function STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251) is not in mixedCase
Constant STokenInterface.isSToken (contracts/STokenInterfaces.sol#129) is not in UPPER_CASE_WITH_UNDERSCORES
Function SErc20Interface._addReserves(uint256) (contracts/STokenInterfaces.sol#276) is not in mixedCase
Function SDelegatorInterface._setImplementation(address,bool,bytes) (contracts/STokenInterfaces.sol#298) is not in mixedCase
Function SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307) is not in mixedCase
Function SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable SToken.seizeInternal(address,address,address,uint256).seizeTokens (contracts/SToken.sol#1057) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable SToken.seize(address,address,uint256).seizeTokens (contracts/SToken.sol#1031) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface).seizeTokens (contracts/SToken.sol#990) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable SToken.seize(address,address,uint256).seizeTokens (contracts/SToken.sol#241) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

SToken (contracts/SToken.sol#16-1525) does not implement functions:
        - SToken.doTransferIn(address,uint256) (contracts/SToken.sol#1504)
        - SToken.doTransferOut(address,uint256) (contracts/SToken.sol#1511)
        - SToken.getCashPrior() (contracts/SToken.sol#1498)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

SDelegationStorage.implementation (contracts/STokenInterfaces.sol#283) should be constant
SErc20Storage.underlying (contracts/STokenInterfaces.sol#258) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57)
_setInterestRateModel(InterestRateModel) should be declared external:
        - SToken._setInterestRateModel(InterestRateModel) (contracts/SToken.sol#1445-1453)
        - STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (contracts/STokenInterfaces.sol#298)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

STokenInterfaces.sol

```
Constant ComptrollerInterface.isComptroller (contracts/ComptrollerInterface.sol#5) is not in UPPER_CASE_WITH_UNDERSCORES
Constant InterestRateModel.isInterestRateModel (contracts/InterestRateModel.sol#9) is not in UPPER_CASE_WITH_UNDERSCORES
Variable STokenStorage._notEntered (contracts/STokenInterfaces.sol#10) is not in mixedCase
Constant STokenStorage.borrowRateMaxMantissa (contracts/STokenInterfaces.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.reserveFactorMaxMantissa (contracts/STokenInterfaces.sol#36) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.protocolSeizeShareMantissa (contracts/STokenInterfaces.sol#121) is not in UPPER_CASE_WITH_UNDERSCORES
Function STokenInterface._setPendingAdmin(address) (contracts/STokenInterfaces.sol#246) is not in mixedCase
Function STokenInterface._acceptAdmin() (contracts/STokenInterfaces.sol#247) is not in mixedCase
Function STokenInterface._setComptroller(ComptrollerInterface) (contracts/STokenInterfaces.sol#248) is not in mixedCase
Function STokenInterface._setReserveFactor(uint256) (contracts/STokenInterfaces.sol#249) is not in mixedCase
Function STokenInterface._reduceReserves(uint256) (contracts/STokenInterfaces.sol#250) is not in mixedCase
Function STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251) is not in mixedCase
Constant STokenInterface.isSToken (contracts/STokenInterfaces.sol#129) is not in UPPER_CASE_WITH_UNDERSCORES
Function SErc20Interface._addReserves(uint256) (contracts/STokenInterfaces.sol#276) is not in mixedCase
Function SDelegatorInterface._setImplementation(address,bool,bytes) (contracts/STokenInterfaces.sol#298) is not in mixedCase
Function SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307) is not in mixedCase
Function SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

SDelegationStorage.implementation (contracts/STokenInterfaces.sol#283) should be constant
SErc20Storage.underlying (contracts/STokenInterfaces.sol#258) should be constant
STokenStorage._notEntered (contracts/STokenInterfaces.sol#10) should be constant
STokenStorage.accrualBlockNumber (contracts/STokenInterfaces.sol#71) should be constant
STokenStorage.admin (contracts/STokenInterfaces.sol#61) should be constant
STokenStorage.borrowIndex (contracts/STokenInterfaces.sol#76) should be constant
STokenStorage.decimals (contracts/STokenInterfaces.sol#25) should be constant
STokenStorage.initialExchangeRateMantissa (contracts/STokenInterfaces.sol#61) should be constant
STokenStorage.name (contracts/STokenInterfaces.sol#15) should be constant
STokenStorage.pendingAdmin (contracts/STokenInterfaces.sol#66) should be constant
STokenStorage.reserveFactorMantissa (contracts/STokenInterfaces.sol#66) should be constant
STokenStorage.symbol (contracts/STokenInterfaces.sol#20) should be constant
STokenStorage.totalBorrows (contracts/STokenInterfaces.sol#81) should be constant
STokenStorage.totalReserves (contracts/STokenInterfaces.sol#86) should be constant
STokenStorage.totalSupply (contracts/STokenInterfaces.sol#91) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

borrowBalanceStored(address) should be declared external:
        - STokenInterface.borrowBalanceStored(address) (contracts/STokenInterfaces.sol#236)
exchangeRateCurrent() should be declared external:
        - STokenInterface.exchangeRateCurrent() (contracts/STokenInterfaces.sol#237)
exchangeRateStored() should be declared external:
        - STokenInterface.exchangeRateStored() (contracts/STokenInterfaces.sol#238)
accrueInterest() should be declared external:
        - STokenInterface.accrueInterest() (contracts/STokenInterfaces.sol#240)
_setComptroller(ComptrollerInterface) should be declared external:
        - STokenInterface._setComptroller(ComptrollerInterface) (contracts/STokenInterfaces.sol#248)
_setInterestRateModel(InterestRateModel) should be declared external:
        - STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (contracts/STokenInterfaces.sol#298)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

StrikeAggregatorPriceOracle.sol

```
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#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/SToken.sol#390)
SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - require(bool,string)(mathErr == MathError.NO_ERROR,could not calculate block delta) (contracts/SToken.sol#406)
SToken.balanceOfUnderlying(address) (contracts/SToken.sol#190-195) uses a dangerous strict equality:
        - require(bool,string)(mErr == MathError.NO_ERROR,balance could not be calculated) (contracts/SToken.sol#193)
SToken.borrowBalanceStored(address) (contracts/SToken.sol#271-275) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,borrowBalanceStored: borrowBalanceStoredInternal failed) (contracts/SToken.sol#273)
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) uses a dangerous strict equality:
        - b == 0 (contracts/CarefulMath.sol#42)
SToken.exchangeRateStored() (contracts/SToken.sol#328-332) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,exchangeRateStored: exchangeRateStoredInternal failed) (contracts/SToken.sol#330)
SToken.exchangeRateStoredInternal() (contracts/SToken.sol#339-369) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/SToken.sol#341)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/SToken.sol#33)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting comptroller failed) (contracts/SToken.sol#41)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting interest rate model failed) (contracts/SToken.sol#49)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == uint256(Error.NO_ERROR),LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/SToken.sol#997)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(seizeError == uint256(Error.NO_ERROR),token seizure failed) (contracts/SToken.sol#1011)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_EXCHANGE_CALCULATION_FAILED) (contracts/SToken.sol#536)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_TOTAL_SUPPLY_CALCULATION_FAILED) (contracts/SToken.sol#544)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#547)
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) uses a dangerous strict equality:
        - assert(bool)(err2 == MathError.NO_ERROR) (contracts/Exponential.sol#163)
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) uses a dangerous strict equality:
        - a == 0 (contracts/CarefulMath.sol#25)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - a == 0 || b == 0 (contracts/Exponential.sol#306)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - require(bool,string)(c / a == b,errorMessage) (contracts/Exponential.sol#310)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#899)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#902)
SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,exchange rate math error) (contracts/SToken.sol#1089)
SToken.transfer(address,uint256) (contracts/SToken.sol#135-137) uses a dangerous strict equality:
        - transferTokens(msg.sender,msg.sender,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#136)
SToken.transferFrom(address,address,uint256) (contracts/SToken.sol#146-148) uses a dangerous strict equality:
        - transferTokens(msg.sender,src,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#696)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

SToken.seizeInternal(address,address,address,uint256).vars (contracts/SToken.sol#1073) is a local variable never initialized
SToken.borrowFresh(address,uint256).vars (contracts/SToken.sol#753) is a local variable never initialized
SToken.redeemFresh(address,uint256,uint256).vars (contracts/SToken.sol#616) is a local variable never initialized
SToken.mintFresh(address,uint256).vars (contracts/SToken.sol#500) is a local variable never initialized
SToken._addReservesFresh(uint256).actualAddAmount (contracts/SToken.sol#1274) is a local variable never initialized
SToken.repayBorrowFresh(address,address,uint256).vars (contracts/SToken.sol#862) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

SErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SErc20.sol#21-34) ignores return value by EIP20Interface(underlying).totalSupply() (contracts/SErc20.sol#33)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Exponential.divScalarByExpTruncate(uint256,Exponential.Exp).fraction (contracts/Exponential.sol#135) shadows:
        - Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

SToken._setPendingAdmin(address).newPendingAdmin (contracts/SToken.sol#1135) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (contracts/SToken.sol#1145)
SErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8).underlying_ (contracts/SErc20.sol#21) lacks a zero-check on :
                - underlying = underlying_ (contracts/SErc20.sol#32)
StrikeAggregatorPriceOracle.setAdmin(address).newAdmin (contracts/StrikeAggregatorPriceOracle.sol#128) lacks a zero-check on :
                - admin = newAdmin (contracts/StrikeAggregatorPriceOracle.sol#131)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#788)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#789)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#790)
Reentrancy in SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/SToken.sol#499)
        State variables written after the call(s):
        - accountTokens[minter] = vars.accountTokensNew (contracts/SToken.sol#551)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#550)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - accountTokens[redeemer] = vars.accountTokensNew (contracts/SToken.sol#697)
Reentrancy in SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#905)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#906)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
Reentrancy in SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
        State variables written after the call(s):
        - accountTokens[borrower] = vars.borrowerTokensNew (contracts/SToken.sol#1112)
        - accountTokens[liquidator] = vars.liquidatorTokensNew (contracts/SToken.sol#1113)
        - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#1111)
Reentrancy in SToken.transferTokens(address,address,address,uint256) (contracts/SToken.sol#68-127):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/SToken.sol#70)
        State variables written after the call(s):
        - accountTokens[src] = srcTokensNew (contracts/SToken.sol#113)
        - accountTokens[dst] = dstTokensNew (contracts/SToken.sol#114)
        - transferAllowances[src][spender] = allowanceNew (contracts/SToken.sol#118)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        Event emitted after the call(s):
        - Borrow(borrower,borrowAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#793)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.BORROW_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#740)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#208)
                - fail(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.BORROW_CASH_NOT_AVAILABLE) (contracts/SToken.sol#750)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#762)
```

StrikePriceOracle.sol

```
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#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/SToken.sol#390)
SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - require(bool,string)(mathErr == MathError.NO_ERROR,could not calculate block delta) (contracts/SToken.sol#406)
SToken.balanceOfUnderlying(address) (contracts/SToken.sol#190-195) uses a dangerous strict equality:
        - require(bool,string)(mErr == MathError.NO_ERROR,balance could not be calculated) (contracts/SToken.sol#193)
SToken.borrowBalanceStored(address) (contracts/SToken.sol#271-275) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,borrowBalanceStored: borrowBalanceStoredInternal failed) (contracts/SToken.sol#273)
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) uses a dangerous strict equality:
        - b == 0 (contracts/CarefulMath.sol#42)
SToken.exchangeRateStored() (contracts/SToken.sol#328-332) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,exchangeRateStored: exchangeRateStoredInternal failed) (contracts/SToken.sol#330)
SToken.exchangeRateStoredInternal() (contracts/SToken.sol#339-369) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/SToken.sol#341)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/SToken.sol#33)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting comptroller failed) (contracts/SToken.sol#41)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting interest rate model failed) (contracts/SToken.sol#49)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == uint256(Error.NO_ERROR),LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/SToken.sol#997)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(seizeError == uint256(Error.NO_ERROR),token seizure failed) (contracts/SToken.sol#1011)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_EXCHANGE_CALCULATION_FAILED) (contracts/SToken.sol#536)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_TOTAL_SUPPLY_CALCULATION_FAILED) (contracts/SToken.sol#544)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#547)
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) uses a dangerous strict equality:
        - assert(bool)(err2 == MathError.NO_ERROR) (contracts/Exponential.sol#163)
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) uses a dangerous strict equality:
        - a == 0 (contracts/CarefulMath.sol#25)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - a == 0 || b == 0 (contracts/Exponential.sol#306)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - require(bool,string)(c / a == b,errorMessage) (contracts/Exponential.sol#310)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#899)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#902)
SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,exchange rate math error) (contracts/SToken.sol#1089)
SToken.transfer(address,uint256) (contracts/SToken.sol#135-137) uses a dangerous strict equality:
        - transferTokens(msg.sender,msg.sender,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#136)
SToken.transferFrom(address,address,uint256) (contracts/SToken.sol#146-148) uses a dangerous strict equality:
        - transferTokens(msg.sender,src,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#696)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

SToken.repayBorrowFresh(address,address,uint256).vars (contracts/SToken.sol#862) is a local variable never initialized
SToken._addReservesFresh(uint256).actualAddAmount (contracts/SToken.sol#1274) is a local variable never initialized
SToken.borrowFresh(address,uint256).vars (contracts/SToken.sol#731) is a local variable never initialized
SToken.mintFresh(address,uint256).vars (contracts/SToken.sol#509) is a local variable never initialized
SToken.seizeInternal(address,address,address,uint256).vars (contracts/SToken.sol#1073) is a local variable never initialized
SToken.redeemFresh(address,uint256,uint256).vars (contracts/SToken.sol#616) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

SErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SErc20.sol#21-34) ignores return value by EIP20Interface(underlying).totalSupply() (contracts/SErc20.sol#33)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Exponential.divScalarByExpTruncate(uint256,Exponential.Exp).fraction (contracts/Exponential.sol#135) shadows:
        - Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

SToken._setPendingAdmin(address).newPendingAdmin (contracts/SToken.sol#1135) lacks a zero-check on :
                - pendingAdmin = newPendingAdmin (contracts/SToken.sol#1145)
SErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8).underlying_ (contracts/SErc20.sol#21) lacks a zero-check on :
                - underlying = underlying_ (contracts/SErc20.sol#32)
StrikePriceOracle.setAdmin(address).newAdmin (contracts/StrikePriceOracle.sol#90) lacks a zero-check on :
                - admin = newAdmin (contracts/StrikePriceOracle.sol#93)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#788)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#789)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#790)
Reentrancy in SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/SToken.sol#499)
        State variables written after the call(s):
        - accountTokens[minter] = vars.accountTokensNew (contracts/SToken.sol#551)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#550)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - accountTokens[redeemer] = vars.accountTokensNew (contracts/SToken.sol#697)
Reentrancy in SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
        State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (contracts/SToken.sol#905)
        - accountBorrows[borrower].interestIndex = borrowIndex (contracts/SToken.sol#906)
        - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
Reentrancy in SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
        State variables written after the call(s):
        - accountTokens[borrower] = vars.borrowerTokensNew (contracts/SToken.sol#1112)
        - accountTokens[liquidator] = vars.liquidatorTokensNew (contracts/SToken.sol#1113)
        - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#1111)
Reentrancy in SToken.transferTokens(address,address,address,uint256) (contracts/SToken.sol#68-127):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/SToken.sol#70)
        State variables written after the call(s):
        - accountTokens[src] = srsTokensNew (contracts/SToken.sol#113)
        - accountTokens[dst] = dstTokensNew (contracts/SToken.sol#114)
        - transferAllowances[src][spender] = allowanceNew (contracts/SToken.sol#118)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in SToken.borrowFresh(address,uint256) (contracts/SToken.sol#736-799):
        External calls:
        - allowed = comptroller.borrowAllowed(address(this),borrower,borrowAmount) (contracts/SToken.sol#738)
        Event emitted after the call(s):
        - Borrow(borrower,borrowAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#793)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.BORROW_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#740)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.BORROW_CASH_NOT_AVAILABLE) (contracts/SToken.sol#750)
```

```
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#762)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#767)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#772)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.BORROW_FRESHNESS_CHECK) (contracts/SToken.sol#745)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED,FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_UINT_MAX),0) (contracts/SToken.sol#981)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.LIQUIDATE_COLLATERAL_FRESHNESS_CHECK),0) (contracts/SToken.sol#966)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.LIQUIDATE_FRESHNESS_CHECK),0) (contracts/SToken.sol#961)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_CLOSE_AMOUNT_REQUESTED,FailureInfo.LIQUIDATE_CLOSE_AMOUNT_IS_ZERO),0) (contracts/SToken.sol#976)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.LIQUIDATE_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#956)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_ACCOUNT_PAIR,FailureInfo.LIQUIDATE_LIQUIDATOR_IS_BORROWER),0) (contracts/SToken.sol#971)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#988)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - (fail(TokenErrorReporter.Error(repayBorrowError),FailureInfo.LIQUIDATE_REPAY_BORROW_FRESH_FAILED),0) (contracts/SToken.sol#988)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
                - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
                - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
Reentrancy in SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020):
        External calls:
        - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
        - (repayBorrowError,actualRepayAmount) = repayBorrowFresh(liquidator,borrower,repayAmount) (contracts/SToken.sol#986)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
        - seizeError = seizeInternal(address(this),liquidator,borrower,seizeTokens) (contracts/SToken.sol#1005)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
        - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
        Event emitted after the call(s):
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(sTokenCollateral),seizeTokens) (contracts/SToken.sol#1014)
Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - (fail(TokenErrorReporter.Error(error),FailureInfo.LIQUIDATE_ACCRUE_COLLATERAL_INTEREST_FAILED),0) (contracts/SToken.sol#936)
Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - LiquidateBorrow(liquidator,borrower,actualRepayAmount,address(sTokenCollateral),seizeTokens) (contracts/SToken.sol#1014)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
                - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
Reentrancy in SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-563):
        External calls:
        - allowed = comptroller.mintAllowed(address(this),minter,mintAmount) (contracts/SToken.sol#499)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.MINT_EXCHANGE_RATE_READ_FAILED,uint256(vars.mathErr)),0) (contracts/SToken.sol#513)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.MINT_FRESHNESS_CHECK),0) (contracts/SToken.sol#506)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.MINT_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#501)
        - Mint(minter,vars.actualMintAmount,vars.mintTokens) (contracts/SToken.sol#554)
        - Transfer(address(this),minter,vars.mintTokens) (contracts/SToken.sol#555)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.REDEEM_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#655)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.REDEEM_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#675)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.TOKEN_INSUFFICIENT_CASH,FailureInfo.REDEEM_TRANSFER_OUT_NOT_POSSIBLE) (contracts/SToken.sol#680)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.REDEEM_NEW_TOTAL_SUPPLY_CALCULATION_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#670)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.REDEEM_FRESHNESS_CHECK) (contracts/SToken.sol#660)
        - Redeem(redeemer,vars.redeemAmount,vars.redeemTokens) (contracts/SToken.sol#701)
        - Transfer(redeemer,address(this),vars.redeemTokens) (contracts/SToken.sol#700)
Reentrancy in SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916):
        External calls:
        - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.REPAY_BORROW_COMPTROLLER_REJECTION,allowed),0) (contracts/SToken.sol#854)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.REPAY_BORROW_ACCUMULATED_BALANCE_CALCULATION_FAILED,uint256(vars.mathErr)),0) (contracts/SToken.sol#870)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.MARKET_NOT_FRESH,FailureInfo.REPAY_BORROW_FRESHNESS_CHECK),0) (contracts/SToken.sol#859)
        - RepayBorrow(payer,borrower,vars.actualRepayAmount,vars.accountBorrowsNew,vars.totalBorrowsNew) (contracts/SToken.sol#910)
Reentrancy in SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124):
        External calls:
        - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_DECREMENT_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#1093)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.LIQUIDATE_SEIZE_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#1061)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
                - failOpaque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_INCREMENT_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#1102)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
                - fail(Error.INVALID_ACCOUNT_PAIR,FailureInfo.LIQUIDATE_SEIZE_LIQUIDATOR_IS_BORROWER) (contracts/SToken.sol#1066)
```

```
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
              - failOpaque(Error.MATH_ERROR,FailureInfo.LIQUIDATE_SEIZE_BALANCE_DECREMENT_FAILED,uint256(vars.mathErr)) (contracts/SToken.sol#1082)
        - ReservesAdded(address(this),vars.protocolSeizeAmount,vars.totalReservesNew) (contracts/SToken.sol#1118)
        - Transfer(borrower,liquidator,vars.liquidatorSeizeTokens) (contracts/SToken.sol#1116)
        - Transfer(borrower,address(this),vars.protocolSeizeTokens) (contracts/SToken.sol#1117)
Reentrancy in SToken.transferTokens(address,address,address,uint256) (contracts/SToken.sol#68-127):
        External calls:
        - allowed = comptroller.transferAllowed(address(this),src,dst,tokens) (contracts/SToken.sol#70)
        Event emitted after the call(s):
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
              - fail(Error.BAD_INPUT,FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/SToken.sol#77)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
              - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_NOT_ENOUGH) (contracts/SToken.sol#101)
        - Failure(uint256(err),uint256(info),opaqueError) (contracts/ErrorReporter.sol#209)
              - failOpaque(Error.COMPTROLLER_REJECTION,FailureInfo.TRANSFER_COMPTROLLER_REJECTION,allowed) (contracts/SToken.sol#72)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
              - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_NOT_ALLOWED) (contracts/SToken.sol#96)
        - Failure(uint256(err),uint256(info),0) (contracts/ErrorReporter.sol#200)
              - fail(Error.MATH_ERROR,FailureInfo.TRANSFER_TOO_MUCH) (contracts/SToken.sol#106)
        - Transfer(src,dst,tokens) (contracts/SToken.sol#122)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

SErc20.doTransferIn(address,uint256) (contracts/SErc20.sol#142-167) uses assembly
        - INLINE ASM (contracts/SErc20.sol#148-160)
SErc20.doTransferOut(address,uint256) (contracts/SErc20.sol#178-197) uses assembly
        - INLINE ASM (contracts/SErc20.sol#183-195)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

ComptrollerErrorReporter.fail(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo) (contracts/ErrorReporter.sol#58-62) is never used and should be removed
ComptrollerErrorReporter.failOpaque(ComptrollerErrorReporter.Error,ComptrollerErrorReporter.FailureInfo,uint256) (contracts/ErrorReporter.sol#67-71) is never used and should be removed
Exponential.addExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#48-52) is never used and should be removed
Exponential.add_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#246-248) is never used and should be removed
Exponential.add_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#242-244) is never used and should be removed
Exponential.divExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#191-193) is never used and should be removed
Exponential.divScalar(Exponential.Exp,uint256) (contracts/Exponential.sol#102-109) is never used and should be removed
Exponential.div_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#326-328) is never used and should be removed
Exponential.div_(Exponential.Double,uint256) (contracts/Exponential.sol#330-332) is never used and should be removed
Exponential.div_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#314-316) is never used and should be removed
Exponential.div_(Exponential.Exp,uint256) (contracts/Exponential.sol#318-320) is never used and should be removed
Exponential.div_(uint256,Exponential.Double) (contracts/Exponential.sol#334-336) is never used and should be removed
Exponential.div_(uint256,Exponential.Exp) (contracts/Exponential.sol#322-324) is never used and should be removed
Exponential.div_(uint256,uint256) (contracts/Exponential.sol#338-340) is never used and should be removed
Exponential.div_(uint256,uint256,string) (contracts/Exponential.sol#342-345) is never used and should be removed
Exponential.fraction(uint256,uint256) (contracts/Exponential.sol#347-349) is never used and should be removed
Exponential.greaterThanExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#221-223) is never used and should be removed
Exponential.isZeroExp(Exponential.Exp) (contracts/Exponential.sol#228-230) is never used and should be removed
Exponential.lessThanExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#207-209) is never used and should be removed
Exponential.lessThanOrEqualExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#214-216) is never used and should be removed
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) is never used and should be removed
Exponential.mulExp(uint256,uint256) (contracts/Exponential.sol#171-173) is never used and should be removed
Exponential.mulExp3(Exponential.Exp,Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#178-184) is never used and should be removed
Exponential.mul_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#289-291) is never used and should be removed
Exponential.mul_(Exponential.Double,uint256) (contracts/Exponential.sol#293-295) is never used and should be removed
Exponential.mul_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#277-279) is never used and should be removed
Exponential.mul_(Exponential.Exp,uint256) (contracts/Exponential.sol#281-283) is never used and should be removed
Exponential.mul_(uint256,Exponential.Double) (contracts/Exponential.sol#297-299) is never used and should be removed
Exponential.safe224(uint256,string) (contracts/Exponential.sol#232-235) is never used and should be removed
Exponential.safe32(uint256,string) (contracts/Exponential.sol#237-240) is never used and should be removed
Exponential.subExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#57-61) is never used and should be removed
Exponential.sub_(Exponential.Double,Exponential.Double) (contracts/Exponential.sol#264-266) is never used and should be removed
Exponential.sub_(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#260-262) is never used and should be removed
SafeMath.add(uint256,uint256) (contracts/SafeMath.sol#28-33) is never used and should be removed
SafeMath.add(uint256,uint256,string) (contracts/SafeMath.sol#43-48) is never used and should be removed
SafeMath.div(uint256,uint256) (contracts/SafeMath.sol#132-134) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/SafeMath.sol#147-154) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/SafeMath.sol#167-169) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/SafeMath.sol#182-185) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (contracts/SafeMath.sol#107-119) is never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/SafeMath.sol#58-60) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (contracts/SafeMath.sol#70-75) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Constant ComptrollerInterface.isComptroller (contracts/ComptrollerInterface.sol#5) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.expScale (contracts/Exponential.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.doubleScale (contracts/Exponential.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.halfExpScale (contracts/Exponential.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.mantissaOne (contracts/Exponential.sol#16) 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
Function SErc20._addReserves(uint256) (contracts/SErc20.sol#117-119) is not in mixedCase
Function SToken._setPendingAdmin(address) (contracts/SToken.sol#1135-1151) is not in mixedCase
Function SToken._acceptAdmin() (contracts/SToken.sol#1158-1178) is not in mixedCase
Function SToken._setComptroller(ComptrollerInterface) (contracts/SToken.sol#1185-1202) is not in mixedCase
Function SToken._setReserveFactor(uint256) (contracts/SToken.sol#1209-1217) is not in mixedCase
Function SToken._reduceReserves(uint256) (contracts/SToken.sol#1316-1324) is not in mixedCase
Function SToken._transferReserves(uint256) (contracts/SToken.sol#1388-1388) is not in mixedCase
Function SToken._setInterestRateModel(InterestRateModel) (contracts/SToken.sol#1445-1453) is not in mixedCase
Variable STokenStorage._notEntered (contracts/STokenInterfaces.sol#10) is not in mixedCase
Constant STokenStorage.borrowRateMaxMantissa (contracts/STokenInterfaces.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.reserveFactorMaxMantissa (contracts/STokenInterfaces.sol#36) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.protocolSeizeShareMantissa (contracts/STokenInterfaces.sol#121) is not in UPPER_CASE_WITH_UNDERSCORES
Function STokenInterface._setPendingAdmin(address) (contracts/STokenInterfaces.sol#246) is not in mixedCase
Function STokenInterface._acceptAdmin() (contracts/STokenInterfaces.sol#247) is not in mixedCase
Function STokenInterface._setComptroller(ComptrollerInterface) (contracts/STokenInterfaces.sol#248) is not in mixedCase
Function STokenInterface._setReserveFactor(uint256) (contracts/STokenInterfaces.sol#249) is not in mixedCase
Function STokenInterface._reduceReserves(uint256) (contracts/STokenInterfaces.sol#250) is not in mixedCase
Function STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251) is not in mixedCase
Constant STokenInterface.isSToken (contracts/STokenInterfaces.sol#129) is not in UPPER_CASE_WITH_UNDERSCORES
Function SErc20Interface._addReserves(uint256) (contracts/STokenInterfaces.sol#276) is not in mixedCase
Function SDelegatorInterface._setImplementation(address,bool,bytes) (contracts/STokenInterfaces.sol#298) is not in mixedCase
Function SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307) is not in mixedCase
Function SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312) is not in mixedCase
Parameter StrikePriceOracle.setRefAddress(IStdReference)._ref (contracts/StrikePriceOracle.sol#38) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface).seizeTokens (contracts/SToken.sol#996) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable SToken.seize(address,address,uint256).seizeTokens (contracts/SToken.sol#1031) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable STokenInterface.seize(address,address,uint256).seizeTokens (contracts/STokenInterfaces.sol#241) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable SToken.seizeInternal(address,address,address,uint256).seizeTokens (contracts/SToken.sol#1057) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

SDelegationStorage.implementation (contracts/STokenInterfaces.sol#283) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - SErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SErc20.sol#21-34)
_setInterestRateModel(InterestRateModel) should be declared external:
        - SToken._setInterestRateModel(InterestRateModel) (contracts/SToken.sol#1445-1453)
        - STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (contracts/STokenInterfaces.sol#298)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312)
setRefAddress(IStdReference) should be declared external:
        - StrikePriceOracle.setRefAddress(IStdReference) (contracts/StrikePriceOracle.sol#38-41)
getRefAddress() should be declared external:
        - StrikePriceOracle.getRefAddress() (contracts/StrikePriceOracle.sol#43-45)
getUnderlyingPrice(SToken) should be declared external:
        - StrikePriceOracle.getUnderlyingPrice(SToken) (contracts/StrikePriceOracle.sol#47-67)
setUnderlyingPrice(SToken,uint256) should be declared external:
        - StrikePriceOracle.setUnderlyingPrice(SToken,uint256) (contracts/StrikePriceOracle.sol#69-74)
setDirectPrice(address,uint256) should be declared external:
        - StrikePriceOracle.setDirectPrice(address,uint256) (contracts/StrikePriceOracle.sol#76-80)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

STRK.sol

```
STRK._writeCheckpoint(address,uint32,uint96,uint96) (contracts/STRK.sol#262-275) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (contracts/STRK.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

STRK.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (contracts/STRK.sol#160-169) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now <= expiry,Strk::delegateBySig: signature expired) (contracts/STRK.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
STRK.getChainId() (contracts/STRK.sol#298-302) uses assembly
        - INLINE ASM (contracts/STRK.sol#300)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Constant STRK.totalSupply (contracts/STRK.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

delegate(address) should be declared external:
        - STRK.delegate(address) (contracts/STRK.sol#147-149)
delegateBySig(address,uint256,uint8,bytes32,bytes32) should be declared external:
        - STRK.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (contracts/STRK.sol#160-169)
getPriorVotes(address,uint256) should be declared external:
        - STRK.getPriorVotes(address,uint256) (contracts/STRK.sol#188-220)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Timelock.sol

```
Timelock.constructor(address,uint256).admin_ (contracts/Timelock.sol#26) lacks a zero-check on :
                - admin = admin_ (contracts/Timelock.sol#30)
Timelock.setPendingAdmin(address).pendingAdmin_ (contracts/Timelock.sol#53) lacks a zero-check on :
                - pendingAdmin = pendingAdmin_ (contracts/Timelock.sol#55)
Timelock.executeTransaction(address,uint256,string,bytes,uint256).target (contracts/Timelock.sol#80) lacks a zero-check on :
                - (success,returnData) = target.call.value(value)(callData) (contracts/Timelock.sol#99)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in Timelock.executeTransaction(address,uint256,string,bytes,uint256) (contracts/Timelock.sol#80-105):
        External calls:
        - (success,returnData) = target.call.value(value)(callData) (contracts/Timelock.sol#99)
        Event emitted after the call(s):
        - ExecuteTransaction(txHash,target,value,signature,data,eta) (contracts/Timelock.sol#102)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Timelock.queueTransaction(address,uint256,string,bytes,uint256) (contracts/Timelock.sol#60-69) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(eta >= getBlockTimestamp().add(delay),Timelock::queueTransaction: Estimated execution block must satisfy delay.) (contracts/Timelock.sol#62)
Timelock.executeTransaction(address,uint256,string,bytes,uint256) (contracts/Timelock.sol#80-105) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(getBlockTimestamp() >= eta,Timelock::executeTransaction: Transaction hasn't surpassed time lock.) (contracts/Timelock.sol#85)
        - require(bool,string)(getBlockTimestamp() <= eta.add(GRACE_PERIOD),Timelock::executeTransaction: Transaction is stale.) (contracts/Timelock.sol#86)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

SafeMath.add(uint256,uint256,string) (contracts/SafeMath.sol#43-48) is never used and should be removed
SafeMath.div(uint256,uint256) (contracts/SafeMath.sol#132-134) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/SafeMath.sol#147-154) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/SafeMath.sol#167-169) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/SafeMath.sol#182-185) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/SafeMath.sol#85-97) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (contracts/SafeMath.sol#107-119) is never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/SafeMath.sol#58-60) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (contracts/SafeMath.sol#70-75) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Low level call in Timelock.executeTransaction(address,uint256,string,bytes,uint256) (contracts/Timelock.sol#80-105):
        - (success,returnData) = target.call.value(value)(callData) (contracts/Timelock.sol#99)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

setDelay(uint256) should be declared external:
        - Timelock.setDelay(uint256) (contracts/Timelock.sol#36-43)
acceptAdmin() should be declared external:
        - Timelock.acceptAdmin() (contracts/Timelock.sol#45-51)
setPendingAdmin(address) should be declared external:
        - Timelock.setPendingAdmin(address) (contracts/Timelock.sol#53-58)
queueTransaction(address,uint256,string,bytes,uint256) should be declared external:
        - Timelock.queueTransaction(address,uint256,string,bytes,uint256) (contracts/Timelock.sol#60-69)
cancelTransaction(address,uint256,string,bytes,uint256) should be declared external:
        - Timelock.cancelTransaction(address,uint256,string,bytes,uint256) (contracts/Timelock.sol#71-78)
executeTransaction(address,uint256,string,bytes,uint256) should be declared external:
        - Timelock.executeTransaction(address,uint256,string,bytes,uint256) (contracts/Timelock.sol#80-105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Unitroller.sol

```
Unitroller.fallback() (contracts/Unitroller.sol#135-147) uses delegatecall to a input-controlled function id
        - (success) = comptrollerImplementation.delegatecall(msg.data) (contracts/Unitroller.sol#137)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-delegatecall

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#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/SToken.sol#390)
SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - require(bool,string)(mathErr == MathError.NO_ERROR,could not calculate block delta) (contracts/SToken.sol#406)
SToken.balanceOfUnderlying(address) (contracts/SToken.sol#190-195) uses a dangerous strict equality:
        - require(bool,string)(mErr == MathError.NO_ERROR,balance could not be calculated) (contracts/SToken.sol#193)
SToken.borrowBalanceStored(address) (contracts/SToken.sol#271-275) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,borrowBalanceStored: borrowBalanceStoredInternal failed) (contracts/SToken.sol#273)
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) uses a dangerous strict equality:
        - b == 0 (contracts/CarefulMath.sol#42)
SToken.exchangeRateStored() (contracts/SToken.sol#328-332) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,exchangeRateStored: exchangeRateStoredInternal failed) (contracts/SToken.sol#330)
SToken.exchangeRateStoredInternal() (contracts/SToken.sol#339-369) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/SToken.sol#341)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/SToken.sol#33)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting comptroller failed) (contracts/SToken.sol#41)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting interest rate model failed) (contracts/SToken.sol#49)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == uint256(Error.NO_ERROR),LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/SToken.sol#997)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(seizeError == uint256(Error.NO_ERROR),token seizure failed) (contracts/SToken.sol#1011)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_EXCHANGE_CALCULATION_FAILED) (contracts/SToken.sol#536)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_TOTAL_SUPPLY_CALCULATION_FAILED) (contracts/SToken.sol#544)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#547)
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) uses a dangerous strict equality:
        - assert(bool)(err2 == MathError.NO_ERROR) (contracts/Exponential.sol#163)
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) uses a dangerous strict equality:
        - a == 0 (contracts/CarefulMath.sol#25)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - a == 0 || b == 0 (contracts/Exponential.sol#306)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - require(bool,string)(c / a == b,errorMessage) (contracts/Exponential.sol#310)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#899)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#902)
SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,exchange rate math error) (contracts/SToken.sol#1089)
SToken.transfer(address,uint256) (contracts/SToken.sol#135-137) uses a dangerous strict equality:
        - transferTokens(msg.sender,msg.sender,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#136)
SToken.transferFrom(address,address,uint256) (contracts/SToken.sol#146-148) uses a dangerous strict equality:
        - transferTokens(msg.sender,src,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
```

```
SToken.redeemUnderlyingInternal(uint256) (contracts/SToken.sol#585-593) is never used and should be removed
SToken.repayBorrowBehalfInternal(address,uint256) (contracts/SToken.sol#822-830) is never used and should be removed
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) is never used and should be removed
SToken.repayBorrowInternal(uint256) (contracts/SToken.sol#806-814) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Low level call in Unitroller.fallback() (contracts/Unitroller.sol#135-147):
        - (success) = comptrollerImplementation.delegatecall(msg.data) (contracts/Unitroller.sol#137)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Constant ComptrollerInterface.isComptroller (contracts/ComptrollerInterface.sol#5) is not in UPPER_CASE_WITH_UNDERSCORES
Variable ComptrollerV2Storage._mintGuardianPaused (contracts/ComptrollerStorage.sol#89) is not in mixedCase
Variable ComptrollerV2Storage._borrowGuardianPaused (contracts/ComptrollerStorage.sol#90) is not in mixedCase
Constant Exponential.expScale (contracts/Exponential.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.doubleScale (contracts/Exponential.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.halfExpScale (contracts/Exponential.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exponential.mantissaOne (contracts/Exponential.sol#16) 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
Function SToken._setPendingAdmin(address) (contracts/SToken.sol#1135-1151) is not in mixedCase
Function SToken._acceptAdmin() (contracts/SToken.sol#1158-1178) is not in mixedCase
Function SToken._setComptroller(ComptrollerInterface) (contracts/SToken.sol#1185-1202) is not in mixedCase
Function SToken._setReserveFactor(uint256) (contracts/SToken.sol#1209-1217) is not in mixedCase
Function SToken._reduceReserves(uint256) (contracts/SToken.sol#1316-1324) is not in mixedCase
Function SToken._transferReserves(uint256) (contracts/SToken.sol#1380-1388) is not in mixedCase
Function SToken._setInterestRateModel(InterestRateModel) (contracts/SToken.sol#1445-1453) is not in mixedCase
Variable STokenStorage._notEntered (contracts/STokenInterfaces.sol#10) is not in mixedCase
Constant STokenStorage.borrowRateMaxMantissa (contracts/STokenInterfaces.sol#13) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.reserveFactorMaxMantissa (contracts/STokenInterfaces.sol#16) is not in UPPER_CASE_WITH_UNDERSCORES
Constant STokenStorage.protocolSeizeShareMantissa (contracts/STokenInterfaces.sol#121) is not in UPPER_CASE_WITH_UNDERSCORES
Function STokenInterface._setPendingAdmin(address) (contracts/STokenInterfaces.sol#246) is not in mixedCase
Function STokenInterface._acceptAdmin() (contracts/STokenInterfaces.sol#247) is not in mixedCase
Function STokenInterface._setComptroller(ComptrollerInterface) (contracts/STokenInterfaces.sol#248) is not in mixedCase
Function STokenInterface._setReserveFactor(uint256) (contracts/STokenInterfaces.sol#249) is not in mixedCase
Function STokenInterface._reduceReserves(uint256) (contracts/STokenInterfaces.sol#250) is not in mixedCase
Function STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251) is not in mixedCase
Constant STokenInterface.isSToken (contracts/STokenInterfaces.sol#129) is not in UPPER_CASE_WITH_UNDERSCORES
Function SErc20Interface._addReserves(uint256) (contracts/STokenInterfaces.sol#276) is not in mixedCase
Function SDelegatorInterface._setImplementation(address,bool,bytes) (contracts/STokenInterfaces.sol#298) is not in mixedCase
Function SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307) is not in mixedCase
Function SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312) is not in mixedCase
Function Unitroller._setPendingImplementation(address) (contracts/Unitroller.sol#38-51) is not in mixedCase
Function Unitroller._acceptImplementation() (contracts/Unitroller.sol#58-76) is not in mixedCase
Function Unitroller._setPendingAdmin(address) (contracts/Unitroller.sol#85-101) is not in mixedCase
Function Unitroller._acceptAdmin() (contracts/Unitroller.sol#108-128) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable STokenInterface.seize(address,address,uint256).seizeTokens (contracts/STokenInterfaces.sol#241) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable SToken.seizeInternal(address,address,address,uint256).seizeTokens (contracts/SToken.sol#1057) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface).seizeTokens (contracts/SToken.sol#996) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Variable SToken.seize(address,address,uint256).seizeTokens (contracts/SToken.sol#1031) is too similar to SToken.seizeInternal(address,address,address,uint256).seizerToken (contracts/SToken.sol#1057)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

SToken (contracts/SToken.sol#16-1525) does not implement functions:
        - SToken.doTransferIn(address,uint256) (contracts/SToken.sol#1504)
        - SToken.doTransferOut(address,uint256) (contracts/SToken.sol#1511)
        - SToken.getCashPrior() (contracts/SToken.sol#1498)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

ComptrollerV1Storage.closeFactorMantissa (contracts/ComptrollerStorage.sol#38) should be constant
ComptrollerV1Storage.liquidationIncentiveMantissa (contracts/ComptrollerStorage.sol#43) should be constant
ComptrollerV1Storage.maxAssets (contracts/ComptrollerStorage.sol#48) should be constant
ComptrollerV2Storage._borrowGuardianPaused (contracts/ComptrollerStorage.sol#90) should be constant
ComptrollerV2Storage._mintGuardianPaused (contracts/ComptrollerStorage.sol#89) should be constant
ComptrollerV2Storage.pauseGuardian (contracts/ComptrollerStorage.sol#88) should be constant
ComptrollerV2Storage.seizeGuardianPaused (contracts/ComptrollerStorage.sol#92) should be constant
ComptrollerV2Storage.transferGuardianPaused (contracts/ComptrollerStorage.sol#91) should be constant
ComptrollerV3Storage.strikeRate (contracts/ComptrollerStorage.sol#110) should be constant
ComptrollerV6Storage.reserveAddress (contracts/ComptrollerStorage.sol#152) should be constant
ComptrollerV6Storage.reserveGuardian (contracts/ComptrollerStorage.sol#149) should be constant
ComptrollerV6Storage.strkStaking (contracts/ComptrollerStorage.sol#155) should be constant
ComptrollerV7Storage.marketCapGuardian (contracts/ComptrollerStorage.sol#160) should be constant
ComptrollerV7Storage.protocolPaused (contracts/ComptrollerStorage.sol#171) should be constant
SDelegationStorage.implementation (contracts/STokenInterfaces.sol#283) should be constant
SErc20Storage.underlying (contracts/STokenInterfaces.sol#258) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) should be declared external:
        - SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57)
_setInterestRateModel(InterestRateModel) should be declared external:
        - SToken._setInterestRateModel(InterestRateModel) (contracts/SToken.sol#1445-1453)
        - STokenInterface._setInterestRateModel(InterestRateModel) (contracts/STokenInterfaces.sol#251)
_setImplementation(address,bool,bytes) should be declared external:
        - SDelegatorInterface._setImplementation(address,bool,bytes) (contracts/STokenInterfaces.sol#298)
_becomeImplementation(bytes) should be declared external:
        - SDelegateInterface._becomeImplementation(bytes) (contracts/STokenInterfaces.sol#307)
_resignImplementation() should be declared external:
        - SDelegateInterface._resignImplementation() (contracts/STokenInterfaces.sol#312)
_setPendingImplementation(address) should be declared external:
        - Unitroller._setPendingImplementation(address) (contracts/Unitroller.sol#38-51)
_acceptImplementation() should be declared external:
        - Unitroller._acceptImplementation() (contracts/Unitroller.sol#58-76)
_setPendingAdmin(address) should be declared external:
        - Unitroller._setPendingAdmin(address) (contracts/Unitroller.sol#85-101)
_acceptAdmin() should be declared external:
        - Unitroller._acceptAdmin() (contracts/Unitroller.sol#108-128)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

WhitePaperInterestRateModel.sol

```
WhitePaperInterestRateModel.getSupplyRate(uint256,uint256,uint256,uint256) (contracts/WhitePaperInterestRateModel.sol#79-84) performs a multiplication on the result of a division:
        -rateToPool = borrowRate.mul(oneMinusReserveFactor).div(1e18) (contracts/WhitePaperInterestRateModel.sol#82)
        -utilizationRate(cash,borrows,reserves).mul(rateToPool).div(1e18) (contracts/WhitePaperInterestRateModel.sol#83)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

SafeMath.add(uint256,uint256,string) (contracts/SafeMath.sol#43-48) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/SafeMath.sol#167-169) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/SafeMath.sol#182-185) is never used and should be removed
SafeMath.mul(uint256,uint256,string) (contracts/SafeMath.sol#107-119) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Constant InterestRateModel.isInterestRateModel (contracts/InterestRateModel.sol#9) is not in UPPER_CASE_WITH_UNDERSCORES
Constant WhitePaperInterestRateModel.blocksPerYear (contracts/WhitePaperInterestRateModel.sol#19) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

getSupplyRate(uint256,uint256,uint256,uint256) should be declared external:
        - WhitePaperInterestRateModel.getSupplyRate(uint256,uint256,uint256,uint256) (contracts/WhitePaperInterestRateModel.sol#79-84)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

GovernorAlpha.sol

```
GovernorAlpha.execute(uint256) (contracts/governance/GovernorAlpha.sol#192-200) sends eth to arbitrary user
        Dangerous calls:
        - timelock.executeTransaction.value(proposal.values[i])(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/governance/GovernorAlpha.sol#197)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

GovernorAlpha._queueOrRevert(address,uint256,string,bytes,uint256) (contracts/governance/GovernorAlpha.sol#187-190) ignores return value by timelock.queueTransaction(target,value,signature,data,eta) (contracts/governance/GovernorAlpha.sol#189)
GovernorAlpha.execute(uint256) (contracts/governance/GovernorAlpha.sol#192-200) ignores return value by timelock.executeTransaction.value(proposal.values[i])(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/governance/GovernorAlpha.sol#197)
GovernorAlpha.__queueSetTimelockPendingAdmin(address,uint256) (contracts/governance/GovernorAlpha.sol#291-294) ignores return value by timelock.queueTransaction(address(timelock),0,setPendingAdmin(address),abi.encode(newPendingAdmin),eta) (contracts/governance/GovernorAlpha.sol#293)
GovernorAlpha.__executeSetTimelockPendingAdmin(address,uint256) (contracts/governance/GovernorAlpha.sol#296-299) ignores return value by timelock.executeTransaction(address(timelock),0,setPendingAdmin(address),abi.encode(newPendingAdmin),eta) (contracts/governance/GovernorAlpha.sol#298)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

GovernorAlpha.cancel(uint256).state (contracts/governance/GovernorAlpha.sol#203) shadows:
        - GovernorAlpha.state(uint256) (contracts/governance/GovernorAlpha.sol#226-246) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

GovernorAlpha.constructor(address,address,address).guardian_ (contracts/governance/GovernorAlpha.sol#130) lacks a zero-check on :
                - guardian = guardian_ (contracts/governance/GovernorAlpha.sol#133)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
GovernorAlpha._queueOrRevert(address,uint256,string,bytes,uint256) (contracts/governance/GovernorAlpha.sol#187-190) has external calls inside a loop: require(bool,string)(! timelock.queuedTransactions(keccak256(bytes)(abi.encode(target,value,signature,data,eta))),GovernorAlpha::_queueOrRevert: proposal action already queued at eta) (contracts/governance/GovernorAlpha.sol#188)
GovernorAlpha._queueOrRevert(address,uint256,string,bytes,uint256) (contracts/governance/GovernorAlpha.sol#187-190) has external calls inside a loop: timelock.queueTransaction(target,value,signature,data,eta) (contracts/governance/GovernorAlpha.sol#189)
GovernorAlpha.execute(uint256) (contracts/governance/GovernorAlpha.sol#192-200) has external calls inside a loop: timelock.executeTransaction.value(proposal.values[i])(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/governance/GovernorAlpha.sol#197)
GovernorAlpha.cancel(uint256) (contracts/governance/GovernorAlpha.sol#202-215) has external calls inside a loop: timelock.cancelTransaction(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/governance/GovernorAlpha.sol#211)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

GovernorAlpha._queueOrRevert(address,uint256,string,bytes,uint256) (contracts/governance/GovernorAlpha.sol#187-190) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(! timelock.queuedTransactions(keccak256(bytes)(abi.encode(target,value,signature,data,eta))),GovernorAlpha::_queueOrRevert: proposal action already queued at eta) (contracts/governance/GovernorAlpha.sol#188)
GovernorAlpha.state(uint256) (contracts/governance/GovernorAlpha.sol#226-246) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= add256(proposal.eta,timelock.GRACE_PERIOD()) (contracts/governance/GovernorAlpha.sol#241)
GovernorAlpha.add256(uint256,uint256) (contracts/governance/GovernorAlpha.sol#301-305) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(c >= a,addition overflow) (contracts/governance/GovernorAlpha.sol#303)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

GovernorAlpha.getChainId() (contracts/governance/GovernorAlpha.sol#312-316) uses assembly
        - INLINE ASM (contracts/governance/GovernorAlpha.sol#314)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

GovernorAlpha._castVote(address,uint256,bool) (contracts/governance/GovernorAlpha.sol#261-279) compares to a boolean constant:
        -require(bool,string)(receipt.hasVoted == false,GovernorAlpha::_castVote: voter already voted) (contracts/governance/GovernorAlpha.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Function GovernorAlpha.__acceptAdmin() (contracts/governance/GovernorAlpha.sol#281-284) is not in mixedCase
Function GovernorAlpha.__abdicate() (contracts/governance/GovernorAlpha.sol#286-289) is not in mixedCase
Function GovernorAlpha.__queueSetTimelockPendingAdmin(address,uint256) (contracts/governance/GovernorAlpha.sol#291-294) is not in mixedCase
Function GovernorAlpha.__executeSetTimelockPendingAdmin(address,uint256) (contracts/governance/GovernorAlpha.sol#296-299) is not in mixedCase
Function TimelockInterface.GRACE_PERIOD() (contracts/governance/GovernorAlpha.sol#321) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

propose(address[],uint256[],string[],bytes[],string) should be declared external:
        - GovernorAlpha.propose(address[],uint256[],string[],bytes[],string) (contracts/governance/GovernorAlpha.sol#136-174)
queue(uint256) should be declared external:
        - GovernorAlpha.queue(uint256) (contracts/governance/GovernorAlpha.sol#176-185)
execute(uint256) should be declared external:
        - GovernorAlpha.execute(uint256) (contracts/governance/GovernorAlpha.sol#192-200)
cancel(uint256) should be declared external:
        - GovernorAlpha.cancel(uint256) (contracts/governance/GovernorAlpha.sol#202-215)
getActions(uint256) should be declared external:
        - GovernorAlpha.getActions(uint256) (contracts/governance/GovernorAlpha.sol#217-220)
getReceipt(uint256,address) should be declared external:
        - GovernorAlpha.getReceipt(uint256,address) (contracts/governance/GovernorAlpha.sol#222-224)
castVote(uint256,bool) should be declared external:
        - GovernorAlpha.castVote(uint256,bool) (contracts/governance/GovernorAlpha.sol#248-250)
castVoteBySig(uint256,bool,uint8,bytes32,bytes32) should be declared external:
        - GovernorAlpha.castVoteBySig(uint256,bool,uint8,bytes32,bytes32) (contracts/governance/GovernorAlpha.sol#252-259)
__acceptAdmin() should be declared external:
        - GovernorAlpha.__acceptAdmin() (contracts/governance/GovernorAlpha.sol#281-284)
__abdicate() should be declared external:
        - GovernorAlpha.__abdicate() (contracts/governance/GovernorAlpha.sol#286-289)
__queueSetTimelockPendingAdmin(address,uint256) should be declared external:
        - GovernorAlpha.__queueSetTimelockPendingAdmin(address,uint256) (contracts/governance/GovernorAlpha.sol#291-294)
__executeSetTimelockPendingAdmin(address,uint256) should be declared external:
        - GovernorAlpha.__executeSetTimelockPendingAdmin(address,uint256) (contracts/governance/GovernorAlpha.sol#296-299)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

GovernorAlpha2.sol

```
GovernorAlpha2._queueOrRevert(address,uint256,string,bytes,uint256) (contracts/governance/GovernorAlpha2.sol#188-191) ignores return value by timelock.queueTransaction(target,value,signature,data,eta) (contracts/governance/GovernorAlpha2.sol#190)
GovernorAlpha2.execute(uint256) (contracts/governance/GovernorAlpha2.sol#193-201) ignores return value by timelock.executeTransaction(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/governance/GovernorAlpha2.sol#198)
GovernorAlpha2.__queueSetTimelockPendingAdmin(address,uint256) (contracts/governance/GovernorAlpha2.sol#292-295) ignores return value by timelock.queueTransaction(address(timelock),0,setPendingAdmin(address),abi.encode(newPendingAdmin),eta) (contracts/governance/GovernorAlpha2.sol#294)
GovernorAlpha2.__executeSetTimelockPendingAdmin(address,uint256) (contracts/governance/GovernorAlpha2.sol#297-300) ignores return value by timelock.executeTransaction(address(timelock),0,setPendingAdmin(address),abi.encode(newPendingAdmin),eta) (contracts/governance/GovernorAlpha2.sol#299)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

GovernorAlpha2.cancel(uint256).state (contracts/governance/GovernorAlpha2.sol#204) shadows:
        - GovernorAlpha2.state(uint256) (contracts/governance/GovernorAlpha2.sol#227-247) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

GovernorAlpha2.constructor(address,address,address,uint256).guardian_ (contracts/governance/GovernorAlpha2.sol#130) lacks a zero-check on :
        - guardian = guardian_ (contracts/governance/GovernorAlpha2.sol#133)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

GovernorAlpha2._queueOrRevert(address,uint256,string,bytes,uint256) (contracts/governance/GovernorAlpha2.sol#188-191) has external calls inside a loop: require(bool,string)(! timelock.queuedTransactions(keccak256(bytes)(abi.encode(target,value,signature,data,eta))),GovernorAlpha::_queueOrRevert: proposal action already queued at eta) (contracts/governance/GovernorAlpha2.sol#189)
GovernorAlpha2._queueOrRevert(address,uint256,string,bytes,uint256) (contracts/governance/GovernorAlpha2.sol#188-191) has external calls inside a loop: timelock.queueTransaction(target,value,signature,data,eta) (contracts/governance/GovernorAlpha2.sol#190)
GovernorAlpha2.execute(uint256) (contracts/governance/GovernorAlpha2.sol#193-201) has external calls inside a loop: timelock.executeTransaction(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/governance/GovernorAlpha2.sol#198)
GovernorAlpha2.cancel(uint256) (contracts/governance/GovernorAlpha2.sol#203-216) has external calls inside a loop: timelock.cancelTransaction(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/governance/GovernorAlpha2.sol#212)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

GovernorAlpha2._queueOrRevert(address,uint256,string,bytes,uint256) (contracts/governance/GovernorAlpha2.sol#188-191) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(! timelock.queuedTransactions(keccak256(bytes)(abi.encode(target,value,signature,data,eta))),GovernorAlpha::_queueOrRevert: proposal action already queued at eta) (contracts/governance/GovernorAlpha2.sol#189)
GovernorAlpha2.state(uint256) (contracts/governance/GovernorAlpha2.sol#227-247) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= add256(proposal.eta,timelock.GRACE_PERIOD()) (contracts/governance/GovernorAlpha2.sol#242)
GovernorAlpha2.add256(uint256,uint256) (contracts/governance/GovernorAlpha2.sol#302-306) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(c >= a,addition overflow) (contracts/governance/GovernorAlpha2.sol#304)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

GovernorAlpha2.getChainId() (contracts/governance/GovernorAlpha2.sol#313-317) uses assembly
        - INLINE ASM (contracts/governance/GovernorAlpha2.sol#315)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

GovernorAlpha2._castVote(address,uint256,bool) (contracts/governance/GovernorAlpha2.sol#262-280) compares to a boolean constant:
        -require(bool,string)(receipt.hasVoted == false,GovernorAlpha::_castVote: voter already voted) (contracts/governance/GovernorAlpha2.sol#266)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Function GovernorAlpha2.__acceptAdmin() (contracts/governance/GovernorAlpha2.sol#282-285) is not in mixedCase
Function GovernorAlpha2.__abdicate() (contracts/governance/GovernorAlpha2.sol#287-290) is not in mixedCase
Function GovernorAlpha2.__queueSetTimelockPendingAdmin(address,uint256) (contracts/governance/GovernorAlpha2.sol#292-295) is not in mixedCase
Function GovernorAlpha2.__executeSetTimelockPendingAdmin(address,uint256) (contracts/governance/GovernorAlpha2.sol#297-300) is not in mixedCase
Function TimelockInterface.GRACE_PERIOD() (contracts/governance/GovernorAlpha2.sol#322) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable GovernorAlpha2.constructor(address,address,address,uint256).latestProposalId_ (contracts/governance/GovernorAlpha2.sol#130) is too similar to GovernorAlpha2.latestProposalIds (contracts/governance/GovernorAlpha2.sol#107)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

propose(address[],uint256[],string[],bytes[],string) should be declared external:
        - GovernorAlpha2.propose(address[],uint256[],string[],bytes[],string) (contracts/governance/GovernorAlpha2.sol#137-175)
queue(uint256) should be declared external:
        - GovernorAlpha2.queue(uint256) (contracts/governance/GovernorAlpha2.sol#177-186)
execute(uint256) should be declared external:
        - GovernorAlpha2.execute(uint256) (contracts/governance/GovernorAlpha2.sol#193-201)
cancel(uint256) should be declared external:
        - GovernorAlpha2.cancel(uint256) (contracts/governance/GovernorAlpha2.sol#203-216)
getActions(uint256) should be declared external:
        - GovernorAlpha2.getActions(uint256) (contracts/governance/GovernorAlpha2.sol#218-221)
getReceipt(uint256,address) should be declared external:
        - GovernorAlpha2.getReceipt(uint256,address) (contracts/governance/GovernorAlpha2.sol#223-225)
castVote(uint256,bool) should be declared external:
        - GovernorAlpha2.castVote(uint256,bool) (contracts/governance/GovernorAlpha2.sol#249-251)
castVoteBySig(uint256,bool,uint8,bytes32,bytes32) should be declared external:
        - GovernorAlpha2.castVoteBySig(uint256,bool,uint8,bytes32,bytes32) (contracts/governance/GovernorAlpha2.sol#253-260)
__acceptAdmin() should be declared external:
        - GovernorAlpha2.__acceptAdmin() (contracts/governance/GovernorAlpha2.sol#282-285)
__abdicate() should be declared external:
        - GovernorAlpha2.__abdicate() (contracts/governance/GovernorAlpha2.sol#287-290)
__queueSetTimelockPendingAdmin(address,uint256) should be declared external:
        - GovernorAlpha2.__queueSetTimelockPendingAdmin(address,uint256) (contracts/governance/GovernorAlpha2.sol#292-295)
__executeSetTimelockPendingAdmin(address,uint256) should be declared external:
        - GovernorAlpha2.__executeSetTimelockPendingAdmin(address,uint256) (contracts/governance/GovernorAlpha2.sol#297-300)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

AUTOMATED TESTING

StrikeLens.sol

```
GovernorAlpha.execute(uint256) (contracts/Governance/GovernorAlpha.sol#192-200) sends eth to arbitrary user
        Dangerous calls:
        - timelock.executeTransaction.value(proposal.values[i])(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/Governance/GovernorAlpha.sol#197)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

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#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

STRK._writeCheckpoint(address,uint32,uint96,uint96) (contracts/Governance/STRK.sol#263-276) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (contracts/Governance/STRK.sol#266)
SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - accrualBlockNumberPrior == currentBlockNumber (contracts/SToken.sol#390)
SToken.accrueInterest() (contracts/SToken.sol#384-462) uses a dangerous strict equality:
        - require(bool,string)(mathErr == MathError.NO_ERROR,could not calculate block delta) (contracts/SToken.sol#406)
SToken.balanceOfUnderlying(address) (contracts/SToken.sol#190-195) uses a dangerous strict equality:
        - require(bool,string)(mErr == MathError.NO_ERROR,balance could not be calculated) (contracts/SToken.sol#193)
SToken.borrowBalanceStored(address) (contracts/SToken.sol#271-275) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,borrowBalanceStored: borrowBalanceStoredInternal failed) (contracts/SToken.sol#273)
CarefulMath.divUInt(uint256,uint256) (contracts/CarefulMath.sol#41-47) uses a dangerous strict equality:
        - b == 0 (contracts/CarefulMath.sol#42)
SToken.exchangeRateStored() (contracts/SToken.sol#328-332) uses a dangerous strict equality:
        - require(bool,string)(err == MathError.NO_ERROR,exchangeRateStored: exchangeRateStoredInternal failed) (contracts/SToken.sol#330)
SToken.exchangeRateStoredInternal() (contracts/SToken.sol#339-369) uses a dangerous strict equality:
        - _totalSupply == 0 (contracts/SToken.sol#341)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(accrualBlockNumber == 0 && borrowIndex == 0,market may only be initialized once) (contracts/SToken.sol#33)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting comptroller failed) (contracts/SToken.sol#41)
SToken.initialize(ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SToken.sol#26-57) uses a dangerous strict equality:
        - require(bool,string)(err == uint256(Error.NO_ERROR),setting interest rate model failed) (contracts/SToken.sol#49)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(amountSeizeError == uint256(Error.NO_ERROR),LIQUIDATE_COMPTROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (contracts/SToken.sol#997)
SToken.liquidateBorrowFresh(address,address,uint256,STokenInterface) (contracts/SToken.sol#952-1020) uses a dangerous strict equality:
        - require(bool,string)(seizeError == uint256(Error.NO_ERROR),token seizure failed) (contracts/SToken.sol#1011)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_EXCHANGE_CALCULATION_FAILED) (contracts/SToken.sol#536)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_TOTAL_SUPPLY_CALCULATION_FAILED) (contracts/SToken.sol#544)
SToken.mintFresh(address,uint256) (contracts/SToken.sol#497-561) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,MINT_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#547)
Exponential.mulExp(Exponential.Exp,Exponential.Exp) (contracts/Exponential.sol#146-166) uses a dangerous strict equality:
        - assert(bool)(err2 == MathError.NO_ERROR) (contracts/Exponential.sol#163)
CarefulMath.mulUInt(uint256,uint256) (contracts/CarefulMath.sol#24-36) uses a dangerous strict equality:
        - a == 0 (contracts/CarefulMath.sol#25)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - a == 0 || b == 0 (contracts/Exponential.sol#306)
Exponential.mul_(uint256,uint256,string) (contracts/Exponential.sol#305-312) uses a dangerous strict equality:
        - require(bool,string)(c / a == b,errorMessage) (contracts/Exponential.sol#310)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_ACCOUNT_BORROW_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#899)
SToken.repayBorrowFresh(address,address,uint256) (contracts/SToken.sol#850-916) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,REPAY_BORROW_NEW_TOTAL_BALANCE_CALCULATION_FAILED) (contracts/SToken.sol#902)
SToken.seizeInternal(address,address,address,uint256) (contracts/SToken.sol#1057-1124) uses a dangerous strict equality:
        - require(bool,string)(vars.mathErr == MathError.NO_ERROR,exchange rate math error) (contracts/SToken.sol#1089)
SToken.transfer(address,uint256) (contracts/SToken.sol#135-137) uses a dangerous strict equality:
        - transferTokens(msg.sender,msg.sender,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#136)
SToken.transferFrom(address,address,uint256) (contracts/SToken.sol#146-148) uses a dangerous strict equality:
        - transferTokens(msg.sender,src,dst,amount) == uint256(Error.NO_ERROR) (contracts/SToken.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in SToken.liquidateBorrowInternal(address,uint256,STokenInterface) (contracts/SToken.sol#926-941):
        External calls:
        - error = sTokenCollateral.accrueInterest() (contracts/SToken.sol#933)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - allowed = comptroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1059)
                - allowed = comptroller.liquidateBorrowAllowed(address(this),address(sTokenCollateral),liquidator,borrower,repayAmount) (contracts/SToken.sol#954)
                - allowed = comptroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (contracts/SToken.sol#852)
                - comptroller.repayBorrowVerify(address(this),payer,borrower,vars.actualRepayAmount,vars.borrowerIndex) (contracts/SToken.sol#913)
                - seizeError = sTokenCollateral.seize(liquidator,borrower,seizeTokens) (contracts/SToken.sol#1007)
                - comptroller.seizeVerify(address(this),seizerToken,liquidator,borrower,seizeTokens) (contracts/SToken.sol#1121)
                - comptroller.liquidateBorrowVerify(address(this),address(sTokenCollateral),liquidator,borrower,actualRepayAmount,seizeTokens) (contracts/SToken.sol#1017)
        State variables written after the call(s):
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalBorrows = vars.totalBorrowsNew (contracts/SToken.sol#907)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,sTokenCollateral) (contracts/SToken.sol#940)
                - totalReserves = vars.totalReservesNew (contracts/SToken.sol#1110)
Reentrancy in SToken.redeemFresh(address,uint256,uint256) (contracts/SToken.sol#613-707):
        External calls:
        - allowed = comptroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (contracts/SToken.sol#653)
        State variables written after the call(s):
        - totalSupply = vars.totalSupplyNew (contracts/SToken.sol#696)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

SToken.seizeInternal(address,address,address,uint256).vars (contracts/SToken.sol#1073) is a local variable never initialized
SToken.borrowFresh(address,uint256).vars (contracts/SToken.sol#753) is a local variable never initialized
SToken.repayBorrowFresh(address,address,uint256).vars (contracts/SToken.sol#862) is a local variable never initialized
SToken.redeemFresh(address,uint256,uint256).vars (contracts/SToken.sol#636) is a local variable never initialized
SToken._addReservesFresh(uint256).actualAddAmount (contracts/SToken.sol#1274) is a local variable never initialized
SToken.mintFresh(address,uint256).vars (contracts/SToken.sol#509) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

GovernorAlpha._queueOrRevert(address,uint256,string,bytes,uint256) (contracts/Governance/GovernorAlpha.sol#187-190) ignores return value by timelock.queueTransaction(target,value,signature,data,eta) (contracts/Governance/GovernorAlpha.sol#189)
GovernorAlpha.execute(uint256) (contracts/Governance/GovernorAlpha.sol#192-200) ignores return value by timelock.executeTransaction.value(proposal.values[i])(proposal.targets[i],proposal.values[i],proposal.signatures[i],proposal.calldatas[i],proposal.eta) (contracts/Governance/GovernorAlpha.sol#197)
GovernorAlpha.__queueSetTimelockPendingAdmin(address,uint256) (contracts/Governance/GovernorAlpha.sol#291-294) ignores return value by timelock.queueTransaction(address(timelock),0,setPendingAdmin(address),abi.encode(newPendingAdmin),eta) (contracts/Governance/GovernorAlpha.sol#293)
GovernorAlpha.__executeSetTimelockPendingAdmin(address,uint256) (contracts/Governance/GovernorAlpha.sol#296-299) ignores return value by timelock.executeTransaction(address(timelock),0,setPendingAdmin(address),abi.encode(newPendingAdmin),eta) (contracts/Governance/GovernorAlpha.sol#298)
SErc20.initialize(address,ComptrollerInterface,InterestRateModel,uint256,string,string,uint8) (contracts/SErc20.sol#21-34) ignores return value by EIP20Interface(underlying).totalSupply() (contracts/SErc20.sol#33)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

Context.sol

```
Context._msgData() (contracts/Lib/GSN/Context.sol#23-26) is never used and should be removed
Context._msgSender() (contracts/Lib/GSN/Context.sol#19-21) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.5.0 (contracts/Lib/GSN/Context.sol#1) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Redundant expression "this (contracts/Lib/GSN/Context.sol#24)" inContext (contracts/Lib/GSN/Context.sol#13-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
```

Math.sol

```
Math.average(uint256,uint256) (contracts/Lib/math/Math.sol#25-28) is never used and should be removed
Math.max(uint256,uint256) (contracts/Lib/math/Math.sol#10-12) is never used and should be removed
Math.min(uint256,uint256) (contracts/Lib/math/Math.sol#17-19) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.5.0 (contracts/Lib/math/Math.sol#1) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

Ownable.sol

```
Context._msgData() (contracts/Lib/GSN/Context.sol#23-26) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.5.0 (contracts/Lib/GSN/Context.sol#1) allows old versions
Pragma version^0.5.0 (contracts/Lib/ownership/Ownable.sol#1) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Redundant expression "this (contracts/Lib/GSN/Context.sol#24)" inContext (contracts/Lib/GSN/Context.sol#13-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

owner() should be declared external:
        - Ownable.owner() (contracts/Lib/ownership/Ownable.sol#30-32)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (contracts/Lib/ownership/Ownable.sol#56-59)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (contracts/Lib/ownership/Ownable.sol#65-67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

IERC20.sol

```
Pragma version^0.5.0 (contracts/Lib/token/ERC20/IERC20.sol#1) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

SafeERC20.sol

```
Address.isContract(address) (contracts/Lib/utils/Address.sol#24-33) uses assembly
        - INLINE ASM (contracts/Lib/utils/Address.sol#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity are used:
        - Version used: ['^0.5.0', '^0.5.5']
        - ^0.5.0 (contracts/Lib/math/SafeMath.sol#1)
        - ^0.5.0 (contracts/Lib/token/ERC20/IERC20.sol#1)
        - ^0.5.0 (contracts/Lib/token/ERC20/SafeERC20.sol#1)
        - ^0.5.5 (contracts/Lib/utils/Address.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.isContract(address) (contracts/Lib/utils/Address.sol#24-33) is never used and should be removed
Address.sendValue(address,uint256) (contracts/Lib/utils/Address.sol#63-69) is never used and should be removed
Address.toPayable(address) (contracts/Lib/utils/Address.sol#41-43) is never used and should be removed
SafeERC20.callOptionalReturn(IERC20,bytes) (contracts/Lib/token/ERC20/SafeERC20.sol#55-74) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (contracts/Lib/token/ERC20/SafeERC20.sol#28-37) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (contracts/Lib/token/ERC20/SafeERC20.sol#44-47) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (contracts/Lib/token/ERC20/SafeERC20.sol#39-42) is never used and should be removed
SafeERC20.safeTransfer(IERC20,address,uint256) (contracts/Lib/token/ERC20/SafeERC20.sol#20-22) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (contracts/Lib/token/ERC20/SafeERC20.sol#24-26) is never used and should be removed
SafeMath.add(uint256,uint256) (contracts/Lib/math/SafeMath.sol#26-31) is never used and should be removed
SafeMath.div(uint256,uint256) (contracts/Lib/math/SafeMath.sol#98-100) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/Lib/math/SafeMath.sol#115-122) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/Lib/math/SafeMath.sol#135-137) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/Lib/math/SafeMath.sol#152-155) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/Lib/math/SafeMath.sol#73-85) is never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/Lib/math/SafeMath.sol#42-44) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (contracts/Lib/math/SafeMath.sol#57-62) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.5.0 (contracts/Lib/math/SafeMath.sol#1) allows old versions
Pragma version^0.5.0 (contracts/Lib/token/ERC20/IERC20.sol#1) allows old versions
Pragma version^0.5.0 (contracts/Lib/token/ERC20/SafeERC20.sol#1) allows old versions
Pragma version^0.5.5 (contracts/Lib/utils/Address.sol#1) is known to contain severe issues (https://solidity.readthedocs.io/en/latest/bugs.html)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in SafeERC20.callOptionalReturn(IERC20,bytes) (contracts/Lib/token/ERC20/SafeERC20.sol#55-74):
        - (success,returndata) = address(token).call(data) (contracts/Lib/token/ERC20/SafeERC20.sol#67)
Low level call in Address.sendValue(address,uint256) (contracts/Lib/utils/Address.sol#63-69):
        - (success) = recipient.call.value(amount)() (contracts/Lib/utils/Address.sol#67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

Address.sol

```
Address.isContract(address) (contracts/Lib/utils/Address.sol#24-33) uses assembly
        - INLINE ASM (contracts/Lib/utils/Address.sol#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Address.isContract(address) (contracts/Lib/utils/Address.sol#24-33) is never used and should be removed
Address.sendValue(address,uint256) (contracts/Lib/utils/Address.sol#63-69) is never used and should be removed
Address.toPayable(address) (contracts/Lib/utils/Address.sol#41-43) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.5.5 (contracts/Lib/utils/Address.sol#1) is known to contain severe issues (https://solidity.readthedocs.io/en/latest/bugs.html)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (contracts/Lib/utils/Address.sol#63-69):
        - (success) = recipient.call.value(amount)() (contracts/Lib/utils/Address.sol#67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

IStrikeStaking.sol

No issues were found by Slither.

StrikeStaking.sol

```
StrikeStaking (contracts/Staking/StrikeStaking.sol#15-640) contract sets array length with a user-controlled value:
        - mintersArray.push(_minters[i]) (contracts/Staking/StrikeStaking.sol#156)
StrikeStaking (contracts/Staking/StrikeStaking.sol#15-640) contract sets array length with a user-controlled value:
        - userLocks[msg.sender].push(lockedBalance(amount,unlockTime)) (contracts/Staking/StrikeStaking.sol#343)
StrikeStaking (contracts/Staking/StrikeStaking.sol#15-640) contract sets array length with a user-controlled value:
        - rewardTokens.push(_stakingToken) (contracts/Staking/StrikeStaking.sol#160)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#array-length-assignment

StrikeStakingProxy.fallback() (contracts/Staking/StrikeStakingProxy.sol#112-124) uses delegatecall to a input-controlled function id
        - (success) = strikeStakingImplementation.delegatecall(msg.data) (contracts/Staking/StrikeStakingProxy.sol#114)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-delegatecall

StrikeStaking.stake(uint256,bool) (contracts/Staking/StrikeStaking.sol#331-352) performs a multiplication on the result of a division:
        -unlockTime = block.timestamp.div(groupDuration).mul(groupDuration).add(lockDuration) (contracts/Staking/StrikeStaking.sol#340)
StrikeStaking.mint(address,uint256) (contracts/Staking/StrikeStaking.sol#357-374) performs a multiplication on the result of a division:
        -unlockTime = block.timestamp.div(groupDuration).mul(groupDuration).add(lockDuration) (contracts/Staking/StrikeStaking.sol#364)
StrikeStaking.withdraw(uint256) (contracts/Staking/StrikeStaking.sol#379-438) performs a multiplication on the result of a division:
        -requiredAmount = remaining.mul(WHOLE).div(WHOLE.sub(penaltyFactor)) (contracts/Staking/StrikeStaking.sol#400)
        -penaltyAmount = penaltyAmount.add(requiredAmount.mul(penaltyFactor).div(WHOLE)) (contracts/Staking/StrikeStaking.sol#411)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

StrikeStaking.withdrawableBalance(address) (contracts/Staking/StrikeStaking.sol#301-314) uses a dangerous strict equality:
        - earnedAmount == 0 (contracts/Staking/StrikeStaking.sol#307)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in StrikeStaking.emergencyWithdraw() (contracts/Staking/StrikeStaking.sol#470-484):
        External calls:
        - stakingToken.safeTransfer(msg.sender,amount) (contracts/Staking/StrikeStaking.sol#479)
        State variables written after the call(s):
        - _notifyReward(address(stakingToken),penaltyAmount) (contracts/Staking/StrikeStaking.sol#481)
                - rewardData[_rewardsToken].rewardRate = reward.div(rewardsDuration) (contracts/Staking/StrikeStaking.sol#547)
                - rewardData[_rewardsToken].rewardRate = reward.add(leftover).div(rewardsDuration) (contracts/Staking/StrikeStaking.sol#551)
                - rewardData[_rewardsToken].lastUpdateTime = block.timestamp (contracts/Staking/StrikeStaking.sol#554)
                - rewardData[_rewardsToken].periodFinish = block.timestamp.add(rewardsDuration) (contracts/Staking/StrikeStaking.sol#555)
```

```
Reentrancy in StrikeStaking.emergencyWithdraw() (contracts/Staking/StrikeStaking.sol#470-484):
        External calls:
        - stakingToken.safeTransfer(msg.sender,amount) (contracts/Staking/StrikeStaking.sol#479)
        - getReward() (contracts/Staking/StrikeStaking.sol#483)
                - (success,returndata) = address(token).call(data) (contracts/Lib/token/ERC20/SafeERC20.sol#67)
                - IERC20(_rewardToken).safeTransfer(msg.sender,reward) (contracts/Staking/StrikeStaking.sol#463)
        State variables written after the call(s):
        - getReward() (contracts/Staking/StrikeStaking.sol#483)
                - rewardData[token].rewardPerTokenStored = _rewardPerToken(token,supply) (contracts/Staking/StrikeStaking.sol#582)
                - rewardData[token].lastUpdateTime = lastTimeRewardApplicable(token) (contracts/Staking/StrikeStaking.sol#583)
                - i < rewardTokens.length (contracts/Staking/StrikeStaking.sol#592)
                - rewardData[token].rewardPerTokenStored = _rewardPerToken(token,supply) (contracts/Staking/StrikeStaking.sol#594)
                - rewardData[token].lastUpdateTime = lastTimeRewardApplicable(token) (contracts/Staking/StrikeStaking.sol#595)
        - getReward() (contracts/Staking/StrikeStaking.sol#483)
                - rewards[account][token] = _earned(account,token,balances[account].locked,supply) (contracts/Staking/StrikeStaking.sol#586)
                - rewards[msg.sender][_rewardToken] = 0 (contracts/Staking/StrikeStaking.sol#462)
                - rewards[account][token] = _earned(account,token,balance,supply) (contracts/Staking/StrikeStaking.sol#597)
        - getReward() (contracts/Staking/StrikeStaking.sol#483)
                - userRewardPerTokenPaid[account][token] = rewardData[token].rewardPerTokenStored (contracts/Staking/StrikeStaking.sol#587)
                - userRewardPerTokenPaid[account][token] = rewardData[token].rewardPerTokenStored (contracts/Staking/StrikeStaking.sol#598)
Reentrancy in StrikeStaking.exit() (contracts/Staking/StrikeStaking.sol#441-454):
        External calls:
        - stakingToken.safeTransfer(msg.sender,amount) (contracts/Staking/StrikeStaking.sol#450)
        State variables written after the call(s):
        - _notifyReward(address(stakingToken),penaltyAmount) (contracts/Staking/StrikeStaking.sol#452)
                - rewardData[_rewardsToken].rewardRate = reward.div(rewardsDuration) (contracts/Staking/StrikeStaking.sol#547)
                - rewardData[_rewardsToken].rewardRate = reward.add(leftover).div(rewardsDuration) (contracts/Staking/StrikeStaking.sol#551)
                - rewardData[_rewardsToken].lastUpdateTime = block.timestamp (contracts/Staking/StrikeStaking.sol#554)
                - rewardData[_rewardsToken].periodFinish = block.timestamp.add(rewardsDuration) (contracts/Staking/StrikeStaking.sol#555)
Reentrancy in StrikeStaking.notifyRewardAmount(address,uint256) (contracts/Staking/StrikeStaking.sol#558-566):
        External calls:
        - IERC20(_rewardsToken).safeTransferFrom(msg.sender,address(this),reward) (contracts/Staking/StrikeStaking.sol#563)
        State variables written after the call(s):
        - _notifyReward(_rewardsToken,reward) (contracts/Staking/StrikeStaking.sol#564)
                - rewardData[_rewardsToken].rewardRate = reward.div(rewardsDuration) (contracts/Staking/StrikeStaking.sol#547)
                - rewardData[_rewardsToken].rewardRate = reward.add(leftover).div(rewardsDuration) (contracts/Staking/StrikeStaking.sol#551)
                - rewardData[_rewardsToken].lastUpdateTime = block.timestamp (contracts/Staking/StrikeStaking.sol#554)
                - rewardData[_rewardsToken].periodFinish = block.timestamp.add(rewardsDuration) (contracts/Staking/StrikeStaking.sol#555)
Reentrancy in StrikeStaking.removeBlacklistedLocks(address,address,address) (contracts/Staking/StrikeStaking.sol#510-541):
        External calls:
        - stakingToken.safeTransfer(to,amount) (contracts/Staking/StrikeStaking.sol#531)
        State variables written after the call(s):
        - rewards[account][rewardTokens[i_scope_0]] = 0 (contracts/Staking/StrikeStaking.sol#537)
Reentrancy in StrikeStaking.withdraw(uint256) (contracts/Staking/StrikeStaking.sol#379-438):
        External calls:
        - stakingToken.safeTransfer(msg.sender,amount) (contracts/Staking/StrikeStaking.sol#433)
        State variables written after the call(s):
        - _notifyReward(address(stakingToken),penaltyAmount) (contracts/Staking/StrikeStaking.sol#435)
                - rewardData[_rewardsToken].rewardRate = reward.div(rewardsDuration) (contracts/Staking/StrikeStaking.sol#547)
                - rewardData[_rewardsToken].rewardRate = reward.add(leftover).div(rewardsDuration) (contracts/Staking/StrikeStaking.sol#551)
                - rewardData[_rewardsToken].lastUpdateTime = block.timestamp (contracts/Staking/StrikeStaking.sol#554)
                - rewardData[_rewardsToken].periodFinish = block.timestamp.add(rewardsDuration) (contracts/Staking/StrikeStaking.sol#555)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

StrikeStaking.removeBlacklistedLocks(address,address,address).i_scope_0 (contracts/Staking/StrikeStaking.sol#534) is a local variable never initialized
StrikeStaking.withdraw(uint256).penaltyAmount (contracts/Staking/StrikeStaking.sol#383) is a local variable never initialized
```

StrikeStakingProxy.sol

```
StrikeStakingProxy.fallback() (contracts/Staking/StrikeStakingProxy.sol#112-124) uses delegatecall to a input-controlled function id
        - (success) = strikeStakingImplementation.delegatecall(msg.data) (contracts/Staking/StrikeStakingProxy.sol#114)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-delegatecall

StrikeStakingProxy._setPendingImplementation(address).newPendingImplementation (contracts/Staking/StrikeStakingProxy.sol#36) lacks a zero-check on :
        - pendingStrikeStakingImplementation = newPendingImplementation (contracts/Staking/StrikeStakingProxy.sol#41)
StrikeStakingProxy._setPendingAdmin(address).newPendingAdmin (contracts/Staking/StrikeStakingProxy.sol#72) lacks a zero-check on :
        - pendingAdmin = newPendingAdmin (contracts/Staking/StrikeStakingProxy.sol#79)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

StrikeStakingProxy.fallback() (contracts/Staking/StrikeStakingProxy.sol#112-124) uses assembly
        - INLINE ASM (contracts/Staking/StrikeStakingProxy.sol#116-123)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Low level call in StrikeStakingProxy.fallback() (contracts/Staking/StrikeStakingProxy.sol#112-124):
        - (success) = strikeStakingImplementation.delegatecall(msg.data) (contracts/Staking/StrikeStakingProxy.sol#114)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function StrikeStakingProxy._setPendingImplementation(address) (contracts/Staking/StrikeStakingProxy.sol#36-44) is not in mixedCase
Function StrikeStakingProxy._acceptImplementation() (contracts/Staking/StrikeStakingProxy.sol#50-64) is not in mixedCase
Function StrikeStakingProxy._setPendingAdmin(address) (contracts/Staking/StrikeStakingProxy.sol#72-83) is not in mixedCase
Function StrikeStakingProxy._acceptAdmin() (contracts/Staking/StrikeStakingProxy.sol#90-105) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

_setPendingImplementation(address) should be declared external:
        - StrikeStakingProxy._setPendingImplementation(address) (contracts/Staking/StrikeStakingProxy.sol#36-44)
_acceptImplementation() should be declared external:
        - StrikeStakingProxy._acceptImplementation() (contracts/Staking/StrikeStakingProxy.sol#50-64)
_setPendingAdmin(address) should be declared external:
        - StrikeStakingProxy._setPendingAdmin(address) (contracts/Staking/StrikeStakingProxy.sol#72-83)
_acceptAdmin() should be declared external:
        - StrikeStakingProxy._acceptAdmin() (contracts/Staking/StrikeStakingProxy.sol#90-105)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

StrikeStakingStorage.sol

```
StrikeStakingProxyAdminStorage.admin (contracts/Staking/StrikeStakingStorage.sol#7) should be constant
StrikeStakingProxyAdminStorage.pendingAdmin (contracts/Staking/StrikeStakingStorage.sol#12) should be constant
StrikeStakingProxyAdminStorage.pendingStrikeStakingImplementation (contracts/Staking/StrikeStakingStorage.sol#22) should be constant
StrikeStakingProxyAdminStorage.strikeStakingImplementation (contracts/Staking/StrikeStakingStorage.sol#17) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

- All the reentrancies were checked individually and were considered false positives.
- All the delegatecalls are executed securely.
- Ether is never sent to an arbitrary destination and is correctly handled.
- No major issues were found by Slither.

THANK YOU FOR CHOOSING

// HALBORN