Quantstamp Security Assessment Certificate

2024/12/23— Quantstamp Verified

X-AI

This audit report was prepared by Quantstamp, the leader in blockchain security.

Executive Summary

Type	Privacy-preserving DeFi platform
Auditors	Fayçal Lalidji, Senior Security Engineer Cristiano Silva, Research Engineer Guillermo Escobero, Security Auditor
Timeline	2024-10-6 through 2024-12-22
EVM	London
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	High
Test Quality	Medium

Source Code

Repository	Commit
Core Contracts	9c20d23
Bridge Contracts	Initial Reaudit (4c9e45c)

Total Issues	**14** (8 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	**5** (4 Resolved)
Low Risk Issues	**3** (1 Resolved)
Informational Risk Issues	**6** (3 Resolved)
Undetermined Risk Issues	0 (0 Resolved)

ALL ISSUES ADDRESSED

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
Undetermined	The impact of the issue is uncertain.
Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
Acknowledged	The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).
Fixed	Adjusted program implementation, requirements or constraints to eliminate the risk.
Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

Initial Audit:
Through reviewing the code, we found 19 potential issues with four medium severity issues, six low, and 9 informational. We recommend carefully re-considering the logic to ensure the safety of the users.

First Reaudit: Most previously highlighted issues have been fixed, acknowledged, or mitigated except QSP-7, while new issues that must be fixed before deployment have been added to the report (QSP-14 and 15).

Final Reaudit: All highlighted issues have been addressed.

ID	Description	Severity	Status
QSP-1	Violating Checks Effects Interactions Pattern	^ Medium	Mitigated
QSP-2	Unsafe Cast Operation	^ Medium	Fixed
QSP-3	Adding New Bridge Asset May Fail	^ Medium	Fixed
QSP-4	Adding New Bridge Asset Do Not Sync the Bridge Pool	^ Medium	Fixed
QSP-5	Cannot Add Previously Removed Bridge Asset	v Low	Fixed
QSP-6	Using `call()` Instead of `transfer()` For Sending Ether	v Low	Acknowledged
QSP-7	Confusion In Return Value	v Low	Acknowledged
QSP-8	Unlocked Pragma	O Informational	Acknowledged
QSP-9	Unnecessary Public Visibility for State Variables	O Informational	Fixed
QSP-10	Use of Hard-Coded Values	O Informational	Fixed
QSP-11	Clone-and-Own	O Informational	Acknowledged
QSP-12	Allowance Double-Spend Exploit	O Informational	Mitigated
QSP-13	Ownership Can Be Renounced	O Informational	Acknowledged
QSP-14	`assertandgetdecimals(...)` Does Not Throw in Case of a Contract that Is Not Erc20 Compliant	^ Medium	Acknowledged

Quantstamp Audit Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:

- Slither v0.8.3

Steps taken to run the tools:

1. Installed the Slither tool: `pip install slither-analyzer`
2. Run Slither from the project directory: `slither .`

Findings

QSP-1 Violating Checks Effects Interactions Pattern

Severity: *Medium Risk*

Status: Mitigated

File(s) affected: `contracts/*`

Description: The Checks-Effects-Interactions (CEI) pattern describes a way of organizing the statements in a function such that a contract's state is left in a consistent state before calling out to other contracts. This is done by classifying every statement as either a check, an effect (state change), or an interaction, and ensuring that they are strictly in this order. By placing effects before interactions, we make sure that all state changes are done before any potential reentrancy point, leaving the state consistent. In fact, even when we use the modifier `nonReentrant`, we must always use the Checks-Effects-Interaction pattern to reduce the attack surface for malicious contracts trying to hijack control flow after an external call. The CEI pattern is not adopted in several functions of the application. As an example, let's take a look at the implementation of the function `BaseSilo._deposit(...)` presented below.

```
    function
        BaseSilo._deposit( add
        ress _asset, address
        _from,
        address _depositor,
        uint256 _amount,
        bool _collateralOnly
    )
        internal
        nonReentrant
        validateMaxDepositsAfter(_asset)
    {
        // MUST BE CALLED AS FIRST METHOD! we can allow for checks to be run before
        _accrueInterest(_asset, block.timestamp);

        if (!depositPossible(_asset, _depositor)) revert("DepositNotPossible()");

        AssetStorage storage _state = state[_asset];

        uint256 balanceBefore = ERC20(_asset).balanceOf(address(this));
        ERC20(_asset).safeTransferFrom(_from, address(this), _amount);
        uint256 balanceAfter =ERC20(_asset).balanceOf(address(this));

        _amount = balanceAfter - balanceBefore;
        uint256 totalDepositsCached = _collateralOnly ? _state.collateralOnlyDeposits : _state.totalDeposits;

        if (_collateralOnly) {
            uint256 share = _amount.toShare(totalDepositsCached,_state.collateralOnlyToken.totalSupply());
            _state.collateralOnlyDeposits = totalDepositsCached + _amount;
            _state.collateralOnlyToken.mint(_depositor, share);
        } else {
            uint256 share = _amount.toShare(totalDepositsCached, _state.collateralToken.totalSupply());
            _state.totalDeposits = totalDepositsCached + _amount;
            _state.collateralToken.mint(_depositor, share);
        }

        emit Deposit(_asset, _depositor, _amount, _collateralOnly);
    }
```

We notice that the interaction with the external contract happens in the middle of the function.
When following the CEI pattern, this line should be the last line of the function. Adapting the function to such a scenario is simple. Basically, we must postpone the external call and include a `require` such as the transferred `amount` (new variable) is equal to the input parameter `_amount`. The code will look similar to the one below.

```
    function
        _deposit( address
        _asset, address
        _from, address
        _depositor, uint256
        _amount, bool
        _collateralOnly
    )
        internal
        nonReentrant
        validateMaxDepositsAfter(_asset)
    {
        // Checks section: preparing the environment for executing the function
        _accrueInterest(_asset, block.timestamp);
        if (!depositPossible(_asset, _depositor)) revert("DepositNotPossible()");
        AssetStorage storage _state = state[_asset];
        uint256 totalDepositsCached = _collateralOnly ? _state.collateralOnlyDeposits : _state.totalDeposits;

        // Effects section: changing state variables
        if (_collateralOnly) {
            uint256 share = _amount.toShare(totalDepositsCached,_state.collateralOnlyToken.totalSupply());
            _state.collateralOnlyDeposits = totalDepositsCached + _amount;
            _state.collateralOnlyToken.mint(_depositor, share);
        } else {
            uint256 share = _amount.toShare(totalDepositsCached, _state.collateralToken.totalSupply());
            _state.totalDeposits = totalDepositsCached + _amount;
            _state.collateralToken.mint(_depositor, share);
        }

        // Interactions Section: making external call to other contracts
        uint256 balanceBefore = ERC20(_asset).balanceOf(address(this));
        ERC20(_asset).safeTransferFrom(_from, address(this), _amount);
        uint256 balanceAfter = ERC20(_asset).balanceOf(address(this));

        // Should we revert?
        uint256 amount = balanceAfter - balanceBefore;
        require(_amount==amount, "Incorrect amount: reverting the whole operation");

        emit Deposit(_asset, _depositor, _amount, _collateralOnly);
    }
```

The same logic must be applied to each and every function making external calls:

- `BaseSilo._withdraw(...)`, execute external function calls when runing `BaseSilo._withdrawAsset(...)` before setting the final contract state. We recommend to execute the transfer calls in a third function after setting `State.collateralOnlyDeposits` or `State.totalDeposits`.
- `_repay` execute a transfer before setting the final contract state.
- `_repay` must include a non-reentrant modifier for safety.

All the other contracts that present calls to external contracts must be adapted to the CEI pattern, even those having the `nonReentrant` modifier.

Recommendation: Review all the contracts in order to assure that all the functions making external calls are following the Checks-Effects-Interaction Pattern, even functions having the `nonReentrant` modifier must follow the CEI pattern. Otherwise the application will be under risk.

Update: QSP-1 is partially fixed, `BaseSilo._repay(...)` still does not respect the CEI pattern.

QSP-2 Unsafe Cast Operation

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `contracts/lib/ModelStats.sol`

Description: `ModelStats.calculateUtilization(...)` should use `SafeCast` when converting `_dp` to `uint256`, or if `_dp` is an always positive value change its declaration to `uint256`. Please note that using solidity 0.8.0 or higher does not prevent incorrect cast operations.

QSP-3 Adding New Bridge Asset May Fail

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `contracts/SiloRepository.sol`

Description: If a new bridge asset is meant to be added to the pool and if the bridge asset is already set within a silo the `SiloRepository.bridgePool` is set then the admin won't be able to add that asset as a bridge asset. An attacker can use this to prevent the admins from adding new bridge assets purposefully since adding new silo is allowed to anyone.

Recommendation: This behavior should be either clearly documented or fixed.

Update: Fixed by adding extra comments in https://github.com/silo-finance/silo-contracts/pull/322.

QSP-4 Adding New Bridge Asset Do Not Sync the Bridge Pool

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `contracts/SiloRepository.sol`

Description: in `SiloRepository` adding new bridge asset won't sync the actual bridge pool since the external call is set before adding the asset to the bridge list.

Recommendation: Sync the bridge assets after adding the new asset to the list.

Update: Fixed in https://github.com/silo-finance/silo-contracts/pull/316.

QSP-5 Cannot Add Previously Removed Bridge Asset

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts/SiloRepository.sol`

Description: Adding back a bridge asset that was removed using `SiloRepository.addBridgeAsset(...)` will not allow its reactivation in the Silo contract since there is a check in `_initAssetsTokens(...)` that prevent that.

Recommendation: Add the missing else branch in the if condition of L229, resetting the asset status to active.

Update: Fixed in https://github.com/silo-finance/silo-contracts/pull/224.

QSP-6 Using `call()` Instead of `transfer()` For Sending Ether

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `contracts/SiloRouter.sol`

Description: The functions below are using `call()` to transfer Ether instead of the function `transfer()`. Since `call()` forwards all the gas, it can be exploited in reentrancy attacks.

- `SiloRouter._sendAsset(...)`
- `SiloRouter.execute(...)`

Update: "We won't do `transfer()` it will fail for some smart contracts".

QSP-7 Confusion In Return Value

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `contracts/lib/Ping.sol`

Description: `ERC20` Standard `decimals()` can return 0 as a decimal value. Therefore, returning 0 in case of an unsuccessful transaction or an invalid address can lead to confusion or to a possible issue when using `Ping.decimals(...)`.

Recommendation: Change the return value in case of a failed transaction or invalid address.

Update: Acknowledged in commit 4be2bddae241fccf3b45d69b2d47f7f4c40eaf52

QSP-8 Unlocked Pragma

Severity: *Informational*

Status: Acknowledged

Related Issue(s): SWC-103

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.*.*`. The caret (`^`) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version *and above*, hence the term "unlocked".
The project is using different versions of solidity and pragma directives: `0.7.6`, `>=0.4.0`, `>=0.5.0`, `>=0.5.0<0.8.0`, `>=0.6.0<0.8.0`, `>=0.6.0<0.9.0`, `>=0.7.0`, `>=0.7.0<0.9.0`, `^0.7.0`, `0.8.7`, `>=0.7.5`, `^0.8.0`.

Recommendation: For consistency and to prevent unexpected behavior in the future, we recommend removing the caret to lock the file onto a specific Solidity version.

QSP-9 Unnecessary Public Visibility for State Variables

Severity: *Informational*

Status: Fixed

File(s) affected: `contracts/*`

Description: Several contracts present state variables with `public` visibility. A contract variable marked public will generate a getter function to read its value, and there's no way to apply a modifier to that function. This opens up the possibility for exploitation, since it can result in other contracts observing inconsistent state due to broken invariants.

Recommendation: Turning the visibility of the state variables to `private` will reduce contract size and reduce the risk of possible exploits.

Update: Fixed in https://github.com/silo-finance/silo-contracts/pull/321.

QSP-10 Use of Hard-Coded Values

Severity: *Informational*

Status: Fixed

File(s) affected: `contracts/SiloLens.sol`

Description: The function `SiloLens.depositAPY(...)` has the hard-coded value `1e18`, which is not a good programming practice. The function is listed below.

```
    function depositAPY(ISilo _silo, address _asset) external view returns (uint256)
        { IPriceProvidersRepository priceProviderRepo =
        siloRepository.priceProvidersRepository(); uint256 assetPrice =
        priceProviderRepo.getPrice(_asset);
        uint256 assetDecimals = ERC20(_asset).decimals();

        // amount of debt generated per year in asset decimals
        uint256 generatedDebtAmount = totalBorrowAmountWithInterest(_silo, _asset) * borrowAPY(_silo, _asset) / 1e18;
        // generated debt value in ETH per year in 18 decimals
        uint256 generatedDebtValue = generatedDebtAmount * assetPrice / 10 ** assetDecimals;
        // value of deposits in ETH in 18 decimals
        uint256 totalDepositsValue = totalDepositsWithInterest(_silo, _asset) * assetPrice / 10 ** assetDecimals;

        return generatedDebtValue * 1e18 / totalDepositsValue;
    }
```

Recommendation: Use the proper constant to represent the value. In case the values are related, use the same constant.

Update: Fixed in https://github.com/silo-finance/silo-contracts/pull/237.

QSP-11 Clone-and-Own

Severity: *Informational*

Status: Acknowledged

File(s) affected: `contracts/governance/TreasuryVester.sol`, `contracts/lib/PRBMathCommon.sol`, `contracts/lib/PRBMathSD59x18.sol`

Description: The clone-and-own approach involves copying and adjusting open source code at one's own discretion. From the development perspective, it is initially beneficial as it reduces the amount of effort. However, from the security perspective, it involves some risks as the code may not follow the best practices, may contain a security vulnerability, or may include intentionally or unintentionally modified upstream libraries.

Recommendation: Rather than the clone-and-own approach, a good industry practice is to use a package manager (e.g., npm) for handling library dependencies. This eliminates the clone-and-own risks yet allows for following best practices, such as, using libraries. If the file is cloned anyway, a comment including the repository, commit hash of the version cloned, and the summary of modifications (if any) should be added. This helps to improve traceability of the file.

QSP-12 Allowance Double-Spend Exploit

Severity: *Informational*

Status: Mitigated

File(s) affected: `contracts/governance/SiloGovernanceToken.sol`, `contracts/utils/ShareToken.sol`

Description: As they presently are constructed, `SiloGovernanceToken` and `ShareToken` tokens are vulnerable to the allowance allowance double-spend exploit, as with other ERC20 tokens.

Exploit Scenario: 1. Alice allows Bob to transfer `N` amount of Alice's tokens (`N>0`) by calling the `approve()` method on `Token` smart contract (passing Bob's address and `N` as method arguments)

1. After some time, Alice decides to change from `N` to `M` (`M>0`) the number of Alice's tokens Bob is allowed to transfer, so she calls the `approve()` method again, this time passing Bob's address and `M` as method arguments
2. Bob notices Alice's second transaction before it was mined and quickly sends another transaction that calls the `transferFrom()` method to transfer `N` Alice's tokens somewhere
3. If Bob's transaction will be executed before Alice's transaction, then Bob will successfully transfer `N` Alice's tokens and will gain an ability to transfer another `M` tokens
4. Before Alice notices any irregularities, Bob calls `transferFrom()` method again, this time to transfer `M` Alice's tokens.

Recommendation: The exploit (as described above) is mitigated through use of functions that increase/decrease the allowance relative to its current value, such as `increaseAllowance()` and `decreaseAllowance()`. Furthermore, we recommend that developers of applications dependent on `approve()` / `transferFrom()` should keep in mind that they have to set allowance to 0 first and verify if it was used before setting the new value.

QSP-13 Ownership Can Be Renounced

Severity: *Informational*

Status: Acknowledged

File(s) affected: `contracts/InterestRateModel.sol`, `contracts/PriceProvidersRepository.sol`, `contracts/SiloRepository.sol`, `contracts/governance/SiloGovernanceToken.sol`, `contracts/governance/TreasuryVester.sol`, `contracts/liquidation/LiquidationHelper.sol`, `contracts/priceProviders/balancerV2/BalancerV2PriceProvider.sol`, `contracts/priceProviders/uniswapV3/UniswapV3PriceProvider.sol`, `contracts/utils/GuardedLaunch.sol`

Description: If the owner renounces their ownership, all ownable contracts will be left without an owner. Consequently, any function guarded by the `onlyOwner` modifier will no longer be able to be executed

Recommendation: Double check if this is the intended behavior.

QSP-14 `assertandgetdecimals(...)` Does Not Throw in Case of a Contract that Is Not Erc20 Compliant

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `contracts/lib/TokenHelper.sol`

Description: `TokenHelper.assertandgetdecimals(...)`does not revert in case of a contract that is not ERC20 compliant. Please note that the function has been used on multiple occasions to check if an address is a valid ERC20 contract.

Recommendation: When the call to `IERC20Metadata.decimals` fails, clearly revert with the correct message otherwise the return value cannot be distinguished between a contract that has zero decimals and a failing call.

Update: Acknowledged in commit 4be2bddae241fccf3b45d69b2d47f7f4c40eaf52

Automated Analyses

Slither

Slither did not return any significant result.

Adherence to Best Practices

1. the folllwing assignement `uint256 totalDepositsCached = _collateralOnly ? _state.collateralOnlyDeposits : _state.totalDeposits` in `BaseSilo.deposit(...)` can be put inside the if/else condition to save gas.
2. `SiloSnapshotWrapper` implementation inherits from `ERC20` when it is not needed. if the contract needs to act as a wrapper, only the required functions can be implemented.
3. LiquidationHelper.sol: 1.1. The `IWrappedNativeToken` interface is declared. However, this interface already exists in ./contracts/interfaces/IWrappedNativeToken.sol. Consider importing it from that file.
   1.2. L158: change the require revert message to one more descriptive one.
   1.3. `checkDebt(...)` should check input array lengths (similar approach as done in `checkSolvency(...)`)
4. In `Solvency.sol` some functions are not called outside the library (e.g. `getBorrowAmounts(...)`, `convertAmountsToValues(...)` or `getUserCollateralValues(...)`). Consider labeling them as private to improve encapsulation.
5. In `SiloLens.sol` (L298 and L304), and in `SiloRepository.sol` (L76) a non-documented constant is used (1e18). It seems to be related to `Solvency._PRECISION_DECIMALS` constant. Use it or declare a new named constant in the contract.
6. Gas optimizations: 1.1. Declare array length used in loop condition as variable before for loops. 1.2. In SiloSnapshotWrapper.sol consider declaring `siloToken` variable as immutable.
7. The following functions are not called internally. Consider labeling them as external to save gas:
   1. `TwoStepOwnable.renounceOwnership()`
   2. `TwoStepOwnable.transferOwnership(address)`
   3. `TwoStepOwnable.transferPendingOwnership(address)`
   4. `TwoStepOwnable.acceptOwnership()`
   5. `UniswapV3PriceProvider.getPrice(address)`
   6. `PriceProvidersRepository.getPrice(address)`
   7. `Silo.accrueInterest(address)`
   8. `UniswapV3Swap.pathToBytes(address[],uint24[])`
   9. `ERC20R.decreaseReceiveAllowance(address,uint256)`
   10. `ERC20R.increaseReceiveAllowance(address,uint256)`

Test Results

Test Suite Results

```
Run yarn test
yarn run v1.22.18
warning package.json: License should be a valid SPDX license expression
$ npx hardhat test
hardhat forking OFF
No need to generate any newer typings.
  SiloGovernanceToken
    when deployed
      ✓ deployer has 1e9 tokens
-------------------- evm_revert: 0x1
  SiloGovernor
    ✓ setup (83ms)
    testing execution flow
      ✓ propose() (44ms)
      proposed
        ✓ castVote() (38ms)
        voted
          ✓ queue() & execute() (491ms)
-------------------- evm_revert: 0x4
  InterestRateModel
    ✓ #DP
    ✓ getConfig() (42ms)
    ✓ setConfig() (43ms)
    calculateCurrentInterestRate()
      ✓ reverts if timestamps are invalid
gas used: 28325
      ✓ estimateGas() (189ms)
    calculateCompoundInterestRate()
      ✓ reverts if timestamps are invalid
gas used: 33901
      ✓ estimateGas()
  TokenHelper library
    ✓ expect to support standard string ERC20.symbol() Token ABC (42ms)
    ✓ expect to support bytes32 ERC20.symbol() 0x546f6b656e2041424300000000000000000000000000000000000000000000000000 (40ms)
    ✓ expect return question mark on error (94ms)
  LiquidationHelper
    - #executeLiquidation
    - #checkSolvency
    - #checkDebt
    - #findPriceProvider
    when deployed
      - #siloRepository
      - #lens
      - #quoteToken
      - #priceProvidersWithSwapOption
      - #priceProvidersWithSwapOption
      - #swappers
  #siloLiquidationCallback
    - throws when called not by silo
    - throws when not able to repay all debt eg in case when swap was not enough
    - throws when liquidation not profitable
    when #siloLiquidationCallback executed
      - expect valid values in LiquidationBalance event
      - #earnings
BalancerV2PriceProvider
  ✓ #changeSecondsAgo
  ✓ #getPoolQuoteLiquidity (43ms)
  when deployed
    ✓ #vault
    ✓ #secondsAgo is 0
    ✓ #periodForAvgPrice
  #setupAsset
    ✓ throws on invalid verification (99ms)
    ✓ #assetSupported returns FALSE before initialization
    ✓ throws when can't get price for asset (101ms)
    when pool is setup
      ✓ #assetSupported returns TRUE
      ✓ expect to save state for asset
  #changePeriodForAvgPrice
    ✓ throws when period 0
    ✓ expect to change period
  #changeSettings
    ✓ throws when period 0
    ✓ expect to change period and secs ago
  #priceBufferReady
    ✓ returns FALSE when pool is NOT initialized with buffer
    ✓ returns TRUE when pool is initialized with buffer (193ms)
  #getPrice (TWAP calculations)
    ✓ reverts when asset not initialised
    ✓ reverts when pool does NOT have full buffer for TWAP calculations (279ms)
    ✓ return price when pool does have full buffer for TWAP calculations (204ms)
    must work for asset with any decimals
      ✓ returns the price for 18 decimals token (113ms)
      ✓ returns the price for asset with different decimals eg 6 (107ms)
      ✓ returns ONE for quote token
  #verifyPool
    ✓ throws on empty asset
    ✓ throws on invalid pool id (125ms)
    ✓ throws when invalid pool for asset (168ms)
    ✓ throws when invalid pool for quote token (68ms)
    ✓ throws when pool has no quote balance - case 1 [asset, quote] (66ms)
    ✓ throws when pool has no quote balance - case 2 [quote, asset] (157ms)
    ✓ throws when pool has no quote balance (150ms)
    ✓ returns tokens list in original order [asset ,quote] (136ms)
    ✓ returns tokens list in original order [quote, asset] (72ms)
UniswapV3PriceProvider
  when deployed
    ✓ #PriceCalculationData
    ✓ #uniswapV3Factory
    ✓ does NOT have pool for asset
  #setupAsset
    ✓ #assetSupported returns FALSE
    ✓ throws when verification failed (99ms)
    ✓ throws when pool is not ready to provide prices (240ms)
    when asset initialized
      ✓ #assetSupported returns TRUE
      ✓ expect to have pool for asset
  #changePeriodForAvgPrice
    ✓ throws on period 0
    ✓ throws on period greater than or equal timestamp
    ✓ throws when called NOT by manager
    when period set
      ✓ expect have new period
  #changeBlockTime
    ✓ throws on blockTime 0
    ✓ throws on blockTime >= 60
    ✓ throws when called NOT by manager
    when period set
      ✓ expect have new period
  #adjustOracleCardinality
    ✓ expect NOT to increase when has required cardinality (39ms)
    ✓ expect to increase when has required cardinality (111ms)
  #hasEnoughObservations
    ✓ returns TRUE when oldest timestamp is less that required period (146ms)
    ✓ returns FALSE when oldest timestamp is greater that required period (64ms)
  #verifyPool
    ✓ throws on empty asset address
    ✓ throws on empty pool address
    ✓ throws when pool is invalid pool for asset (101ms)
    ✓ throws when pool for asset is empty address (126ms)
    ✓ throws when no liquidity (132ms)
    ✓ returns TRUE when all good (81ms)
  #getPrice
    ✓ throws when asset not initialized
    must work for asset with any decimals
      ✓ returns the price for 18 decimals token (218ms)
      ✓ returns the price for asset with different decimals eg 6 (201ms)
      ✓ returns ONE for quote token
PriceProvidersRepository
  ✓ deployment fails when quote token is not 18 decimals
  when deployed
    ✓ #siloRepository
    ✓ #quoteToken
    ✓ #providerList returns empty array
#Manageable
    ✓ expect manager to be owner by default
    ✓ #changeManager
#addPriceProvider
    ✓ throws when called NOT by owner
    ✓ throws when invalid provider.quoteToken (38ms)
    ✓ emits event NewPriceProvider (44ms)
    when added
      ✓ throws when try to add again
      ✓ expect to be registered
      ✓ #providersCount to be 1
      ✓ #providerList to return providers
#removePriceProvider
    ✓ throws when called NOT by owner
    ✓ throws when not exists
    when exists
      ✓ emits event PriceProviderRemoved
      when removed
        ✓ expect to NOT be registered
        ✓ #providersCount to be 1
#setPriceProviderForAsset
    ✓ throws when called NOT by manager
```

```
      ✓ throws when provider not registered
      when provider registered
        ✓ throws when asset not supported
        ✓ #providersReadyForAsset to be FALSE
        ✓ emits event PriceProviderForAsset (39ms)
        when provider set for asset
          ✓ expect to be provider for asset
          ✓ #providersReadyForAsset to be TRUE
    #getPrice
      ✓ returns ONE for quote token
      ✓ throws when provider reverts (87ms)
      ✓ returns price (97ms)
  Silo unit tests
-------------------- evm_revert: 0x12bb
    ✓ emits AssetStatusUpdate when syncing removed bridge assets (125ms)
-------------------- evm_revert: 0x12bc
    ✓ expect share tokens are not zero addresses
    #getAssets
-------------------- evm_revert: 0x12c1
      ✓ returns all synced assets
      when new bridge asset is  added
-------------------- evm_revert: 0x12c2
        ✓ does not return unsynced bridge asset
        when Silo is synced
-------------------- evm_revert: 0x12c3
          ✓ returns all assets after sync
      when bridge asset is removed
-------------------- evm_revert: 0x12c8
        ✓ returns all assets *before* sync, including removed asset
        when Silo is synced
-------------------- evm_revert: 0x12cd
          ✓ returns all assets *after* sync, including removed asset
          when removed asset is added back
-------------------- evm_revert: 0x12df
            ✓ returns all assets *before* sync, including removed-added asset
            when Silo is synced
-------------------- evm_revert: 0x12f1
              ✓ returns all assets after sync, including removed-added asset
    bridge assets management in the SiloRepository affects silo behavior
      #deposit and #borrow are disabled for removed bridge asset
-------------------- evm_revert: 0x1310
        ✓ #deposit should fail for the removed bridge asset
-------------------- evm_revert: 0x132f
        ✓ #borrow should fail for the removed bridge asset
        #deposit and #borrow are available after added removed bridge asset
-------------------- evm_revert: 0x134b
          ✓ #deposit should work for the bridge asset added after removal (46ms)
-------------------- evm_revert: 0x1367
          ✓ #borrow should work for the bridge asset added after removal (94ms)
    #deposit
      [#0] allows to deposit all possible assets
-------------------- evm_revert: 0x1396
        ✓ [#0] throws on empty asset
-------------------- evm_revert: 0x13c5
        ✓ [#0] emits event (148ms)
-------------------- evm_revert: 0x13c6
        ✓ [#0] emits event for collateral only (140ms)
-------------------- evm_revert: 0x13cf
        ✓ [#0] #getLTV is zero when nothing borrowed
        test collateralOnly option
          [#0] when userA do collateralOnly deposit(collateralAsset)
-------------------- evm_revert: 0x13d8
            ✓ liquidity does not change
-------------------- evm_revert: 0x13d9
            ✓ AssetStorage.collateralOnlyDeposits should change
-------------------- evm_revert: 0x13e2
            ✓ AssetStorage.totalDeposits should not change
            when someone borrows collateral
              when accrueInterest
-------------------- evm_revert: 0x13eb
                ✓ there should be interest, but not for user A
-------------------- evm_revert: 0x13f4
                ✓ user A withdraws collateralOnly without any interest earned (63ms)
            #borrow
-------------------- evm_revert: 0x141d
              ✓ throws when trying to borrow() collateralOnly deposit
-------------------- evm_revert: 0x1446
              ✓ should borrow() using collateralOnly deposit as collateral (354ms)
        [#0] when asset deposited by userA
-------------------- evm_revert: 0x1457
          ✓ throws when userA wants to borrow collateral asset (279ms)
-------------------- evm_revert: 0x147a
          ✓ userB don't have asset and collateral token
-------------------- evm_revert: 0x1495
          ✓ #getLTV is still zero because nothing borrowed
-------------------- evm_revert: 0x149e
          ✓ expect to have valid total deposits
-------------------- evm_revert: 0x14a7
          ✓ #liquidity is equal to deposited value
-------------------- evm_revert: 0x14b0
          ✓ balances are correct after deposit
-------------------- evm_revert: 0x14b9
          ✓ userA can deposit again (163ms)
-------------------- evm_revert: 0x14c2
          ✓ userB can also deposit (164ms)
          [#0] #withdrawFor
-------------------- evm_revert: 0x14d3
            ✓ throws when done NOT by router
            when withdrawFor executed
-------------------- evm_revert: 0x14e4
              ✓ depositor has no deposit
-------------------- evm_revert: 0x14ef
              ✓ receiver got deposit
          [#0] #withdraw
-------------------- evm_revert: 0x1504
            ✓ [#0] throws when withdrawing more collateralOnly then deposited into the silo (165ms)
-------------------- evm_revert: 0x1519
            ✓ [#0] throws when withdrawing more collateral then deposited into the silo (166ms)
-------------------- evm_revert: 0x152e
            ✓ [#0] throws when withdraw collateral but such deposits NOT exist (186ms)
-------------------- evm_revert: 0x1543
            ✓ [#0] throws when withdraw collateralOnly but such deposits NOT exist (165ms)
-------------------- evm_revert: 0x1558
            ✓ [#0] expect to withdraw MAX (195ms)
            [#0] when withdrawn
-------------------- evm_revert: 0x156d
              ✓ tokens balances are correct
          #withdrawFor
-------------------- evm_revert: 0x1582
            ✓ throws when withdrawFor(userA) is done NOT by router
-------------------- evm_revert: 0x158b
            ✓ expect to emit event (221ms)
            when withdrawn
-------------------- evm_revert: 0x1596
              ✓ expect depositor to have no balance
-------------------- evm_revert: 0x15ab
              ✓ expect receiver got deposit
          #calculateCollateralValue
-------------------- evm_revert: 0x15c0
            ✓ should be equal original amount when no interests
-------------------- evm_revert: 0x15d5
            ✓ value should be greater than original amount when interests are included (50ms)
-------------------- evm_revert: 0x15de
            ✓ collateral only should be included into collateral value (187ms)
-------------------- evm_revert: 0x15e9
            ✓ should depend on assetPrice (including collateral only) (288ms)
          collateral token integration tests
-------------------- evm_revert: 0x15fa
            ✓ should #mint collateral tokens to userA
-------------------- evm_revert: 0x1618
            ✓ should #burn collateral tokens on withdraw (230ms)
            #transfer
-------------------- evm_revert: 0x1621
              ✓ userA can #transfer collateral tokens (122ms)
              when userB deposits other asset
-------------------- evm_revert: 0x1636
                ✓ throws when userA transfers collateral to userB who has debt in that asset (213ms)
-------------------- evm_revert: 0x1641
                ✓ throws when userA becomes insolvent after transfer (236ms)
      [#1] allows to deposit all possible assets
-------------------- evm_revert: 0x165e
        ✓ [#1] throws on empty asset
-------------------- evm_revert: 0x167b
        ✓ [#1] emits event (138ms)
-------------------- evm_revert: 0x167c
        ✓ [#1] emits event for collateral only (131ms)
-------------------- evm_revert: 0x1685
        ✓ [#1] #getLTV is zero when nothing borrowed
        test collateralOnly option
          [#1] when userA do collateralOnly deposit(collateralAsset)
-------------------- evm_revert: 0x168e
            ✓ liquidity does not change
-------------------- evm_revert: 0x168f
            ✓ AssetStorage.collateralOnlyDeposits should change
-------------------- evm_revert: 0x1698
            ✓ AssetStorage.totalDeposits should not change
```

```
            when someone borrows collateral
              when accrueInterest
-------------------- evm_revert: 0x16a1
                ✓ there should be interest, but not for user A
-------------------- evm_revert: 0x16aa
                ✓ user A withdraws collateralOnly without any interest earned (60ms)
            #borrow
-------------------- evm_revert: 0x16d3
              ✓ throws when trying to borrow() collateralOnly deposit
-------------------- evm_revert: 0x16fc
              ✓ should borrow() using collateralOnly deposit as collateral (374ms)
          [#1] when asset deposited by userA
-------------------- evm_revert: 0x170d
            ✓ throws when userA wants to borrow collateral asset (286ms)
-------------------- evm_revert: 0x1730
            ✓ userB don't have asset and collateral token
-------------------- evm_revert: 0x174b
            ✓ #getLTV is still zero because nothing borrowed
-------------------- evm_revert: 0x1754
            ✓ expect to have valid total deposits
-------------------- evm_revert: 0x175d
            ✓ #liquidity is equal to deposited value
-------------------- evm_revert: 0x1766
            ✓ balances are correct after deposit
-------------------- evm_revert: 0x176f
            ✓ userA can deposit again (163ms)
-------------------- evm_revert: 0x1778
            ✓ userB can also deposit (175ms)
          [#1] #withdrawFor
-------------------- evm_revert: 0x1789
            ✓ throws when done NOT by router
            when withdrawFor executed
-------------------- evm_revert: 0x179a
              ✓ depositor has no deposit
-------------------- evm_revert: 0x17a5
              ✓ receiver got deposit
          [#1] #withdraw
-------------------- evm_revert: 0x17ba
            ✓ [#1] throws when withdrawing more collateralOnly then deposited into the silo (171ms)
-------------------- evm_revert: 0x17cf
            ✓ [#1] throws when withdrawing more collateral then deposited into the silo (176ms)
-------------------- evm_revert: 0x17e4
            ✓ [#1] throws when withdraw collateral but such deposits NOT exist  (179ms)
-------------------- evm_revert: 0x17f9
            ✓ [#1] throws when withdraw collateralOnly but such deposits NOT exist  (174ms)
-------------------- evm_revert: 0x180e
            ✓ [#1] expect to withdraw MAX (195ms)
            [#1] when withdrawn
-------------------- evm_revert: 0x1823
              ✓ tokens balances are correct
          #withdrawFor
-------------------- evm_revert: 0x1838
            ✓ throws when withdrawFor(userA) is done NOT by router
-------------------- evm_revert: 0x1841
            ✓ expect to emit event (197ms)
            when withdrawn
-------------------- evm_revert: 0x184c
              ✓ expect depositor to have no balance
-------------------- evm_revert: 0x1861
              ✓ expect receiver got deposit
          #calculateCollateralValue
-------------------- evm_revert: 0x1876
            ✓ should be equal original amount when no interests
-------------------- evm_revert: 0x188b
            ✓ value should be greater than original amount when interests are included  (46ms)
-------------------- evm_revert: 0x1894
            ✓ collateral only should be included into collateral value (159ms)
-------------------- evm_revert: 0x189f
            ✓ should depend on assetPrice (including collateral only) (272ms)
          collateral token integration tests
-------------------- evm_revert: 0x18b0
            ✓ should #mint collateral tokens to userA
-------------------- evm_revert: 0x18ce
            ✓ should #burn collateral tokens on withdraw (205ms)
            #transfer
-------------------- evm_revert: 0x18d7
              ✓ userA can #transfer collateral tokens (129ms)
              when userB deposits other asset
-------------------- evm_revert: 0x18ec
                ✓ throws when userA transfers collateral to userB who has debt in that asset (206ms)
-------------------- evm_revert: 0x18f7
                ✓ throws when userA becomes insolvent after transfer (233ms)
      when guarded launch is ON
        throws on limitedMaxLiquidity for every asset
-------------------- evm_revert: 0x1914
          ✓ [0] expect to fail for asset (188ms)
-------------------- evm_revert: 0x1931
          ✓ [0] expect to fail for asset (collateralOnly) (177ms)
-------------------- evm_revert: 0x193c
          ✓ [1] expect to fail for asset (191ms)
-------------------- evm_revert: 0x1947
          ✓ [1] expect to fail for asset (collateralOnly) (187ms)
      #deposit with limitedMaxLiquidity in 2 steps should  fail
        fails for every asset
-------------------- evm_revert: 0x1952
          ✓ [0] expect to fail for asset (227ms)
-------------------- evm_revert: 0x195d
          ✓ [1] expect to fail for asset (231ms)
      depositFor(userB)
        depositFor(userB) all possible assets
-------------------- evm_revert: 0x1968
          ✓ [0] router can depositFor(userB) asset (177ms)
-------------------- evm_revert: 0x1973
          ✓ [0] anyone can depositFor(userB) asset (162ms)
-------------------- evm_revert: 0x197e
          ✓ [1] router can depositFor(userB) asset (198ms)
-------------------- evm_revert: 0x1987
          ✓ [1] anyone can depositFor(userB) asset (169ms)
      when userA made two types of collateral deposits
-------------------- evm_revert: 0x1992
        ✓ userA has two types of deposits
        when userA (with two types of deposit) borrows
          #flashLiquidate when userA is solvent
-------------------- evm_revert: 0x199b
            ✓ expect to NOT liquidate(userA) with two types of deposits as collateral (87ms)
          #flashLiquidate when userA became insolvent
-------------------- evm_revert: 0x19ac
            ✓ will update the silo state during liquidation (203ms)
-------------------- evm_revert: 0x19cf
            ✓ fail to liquidate(userA) when repay amount not enough (178ms)
            when userA liquidated
-------------------- evm_revert: 0x19f6
              ✓ expect tx to emit Liquidate events
-------------------- evm_revert: 0x1a1d
              ✓ expect tx to emit Transfer events
-------------------- evm_revert: 0x1a44
              ✓ expect to have no debt (100ms)
-------------------- evm_revert: 0x1a6b
              ✓ expect to decrease total deposit
-------------------- evm_revert: 0x1a92
              ✓ expect to send both types of deposits to liquidator on  liquidate(userA)
-------------------- evm_revert: 0x1ab9
              ✓ expect view to returns valid assets
-------------------- evm_revert: 0x1ae0
              ✓ expect view to returns valid collaterals data
-------------------- evm_revert: 0x1b07
              ✓ expect view to returns valid amounts to repay
    #borrow
      [0] with all assets
-------------------- evm_revert: 0x1b2e
        ✓ [0] expect to throw when nothing to borrow (175ms)
-------------------- evm_revert: 0x1b55
        ✓ throws when userB wants to borrow more that silo has  (285ms)
-------------------- evm_revert: 0x1b60
        ✓ expect to emit event (482ms)
        [0] when user B borrow
-------------------- evm_revert: 0x1b71
          ✓ expect valid state of tokens (104ms)
      [1] with all assets
-------------------- evm_revert: 0x1b8c
        ✓ [1] expect to throw when nothing to borrow (181ms)
-------------------- evm_revert: 0x1ba7
        ✓ throws when userB wants to borrow more that silo has  (294ms)
-------------------- evm_revert: 0x1bb2
        ✓ expect to emit event (462ms)
        [1] when user B borrow
-------------------- evm_revert: 0x1bc3
          ✓ expect valid state of tokens (105ms)
    #deposit and #borrow for every pair of  assets
      [0] when user A deposits currentAsset
        [0] when user B deposit other asset as collateral
-------------------- evm_revert: 0x1bde
          ✓ silo shares are right before borrow
-------------------- evm_revert: 0x1bf9
          ✓ userB has right LTV after #borrow (329ms)
          test maximumLTV
```

```
            when there is enough deposit
-------------------- evm_revert: 0x1c0a
              ✓ throws when userB wants to borrow more that 100% (maximumLTV) (366ms)
-------------------- evm_revert: 0x1c27
              ✓ userB can borrow maximumLTV and stay solvent (125ms)
          borrowFor(userA)
-------------------- evm_revert: 0x1c5a
            ✓ router can borrowFor(userB) (265ms)
-------------------- evm_revert: 0x1c7b
            ✓ throws when borrowFor() is done NOT by router
        when userB borrows currentAsset
-------------------- evm_revert: 0x1ca0
          ✓ tokens balances are correct
-------------------- evm_revert: 0x1cbb
          ✓ #calculateBorrowValue
-------------------- evm_revert: 0x1cd6
          ✓ #getBorrowAmount
-------------------- evm_revert: 0x1cf1
          ✓ throws when userB wants to deposit
-------------------- evm_revert: 0x1d0c
          ✓ #withdraw (235ms)
         debt token integration tests
-------------------- evm_revert: 0x1d27
            ✓ should #mint debt token to userB
-------------------- evm_revert: 0x1d42
            ✓ userB can #transfer debt (902ms)
            should #burn debt token on  repay
-------------------- evm_revert: 0x1d5d
              ✓ should #burn all debt when repay amountToBorrow and no interest apply  (50ms)
-------------------- evm_revert: 0x1d8e
              ✓ should NOT #burn all debt when repay amount without interest  (105ms)
-------------------- evm_revert: 0x1da9
              ✓ should #burn all debt token on full repay (77ms)
            #transfer
-------------------- evm_revert: 0x1dc8
              ✓ throws when userA did not allow for transfer (168ms)
-------------------- evm_revert: 0x1de3
              ✓ throws when userB transfers debt to someone who has collateral in that asset (323ms)
-------------------- evm_revert: 0x1e08
              ✓ throws when userA become insolvent after debt transfer from userB  (64ms)
-------------------- evm_revert: 0x1e37
              ✓ throws when amount exceeds allowance (182ms)
          when userB borrows again
-------------------- evm_revert: 0x1e56
            ✓ #liquidity is zero
-------------------- evm_revert: 0x1e7d
            ✓ lens borrow data are correct
-------------------- evm_revert: 0x1eb2
            ✓ tokens balances are correct
-------------------- evm_revert: 0x1ee7
            ✓ there are no interests because no time passed
          when a week passed interests should  appear
            when all interests goes to the protocol
-------------------- evm_revert: 0x1f1c
              ✓ #harvestProtocolFees (121ms)
-------------------- evm_revert: 0x1f51
              ✓ userA do not have interests
            when protocol fees is 0%
-------------------- evm_revert: 0x1f74
              ✓ userA got interests
-------------------- evm_revert: 0x1f95
              ✓ accrueInterest() increases the total borrowAmount and deposits
-------------------- evm_revert: 0x1fb4
              ✓ total deposit increased by protocol interests
          #repay
-------------------- evm_revert: 0x1fd3
            ✓ expect to repay all using exact amount (175ms)
-------------------- evm_revert: 0x1ff2
            ✓ expect to repay all using max uint256 amount (181ms)
-------------------- evm_revert: 0x200d
            ✓ expect to repay all providing higher amount than actual debt  (179ms)
-------------------- evm_revert: 0x2028
            ✓ expect to repay part of debt (111ms)
          #repayFor
-------------------- evm_revert: 0x2043
            ✓ anyone can repayFor(userB) if it is solvent (215ms)
            when userB becomes insolvent
-------------------- evm_revert: 0x205e
              ✓ anyone can repayFor(userB) if it is insolvent (175ms)
          #flashLiquidation
            when flashLiquidation is done on solvent userB
-------------------- evm_revert: 0x2079
              ✓ expect to not change assets, debt and collateral tokens balances for  userB
-------------------- evm_revert: 0x2096
              ✓ expect totalBorrowAmount, totalDeposits of assets should not change
            when userB is NOT solvent
-------------------- evm_revert: 0x20b3
              ✓ ltv > liquidationThreshold
              when flashLiquidation executed (interest ON)
-------------------- evm_revert: 0x20d0
                ✓ expect protocol got liquidation fees
-------------------- evm_revert: 0x20ed
                ✓ expect userB to be solvent, there is no debt (102ms)
-------------------- evm_revert: 0x2112
                ✓ expect userB to loose his collateral
-------------------- evm_revert: 0x2137
                ✓ expect userB to have borrowed asset
                - expect liquidatorHelper to have some remaining quote token
-------------------- evm_revert: 0x215c
                ✓ expect userA earned fees on borrowed asset
-------------------- evm_revert: 0x2181
                ✓ expect interests to be applied
      [1] when user A deposits currentAsset
        [1] when user B deposit other asset as collateral
-------------------- evm_revert: 0x21a6
          ✓ silo shares are right before borrow
-------------------- evm_revert: 0x21cb
          ✓ userB has right LTV after #borrow (336ms)
          test maximumLTV
            when there is enough deposit
-------------------- evm_revert: 0x21dc
              ✓ throws when userB wants to borrow more that 100% (maximumLTV) (361ms)
-------------------- evm_revert: 0x21f9
              ✓ userB can borrow maximumLTV and stay solvent (133ms)
          borrowFor(userA)
-------------------- evm_revert: 0x222c
            ✓ router can borrowFor(userB) (272ms)
-------------------- evm_revert: 0x224d
            ✓ throws when borrowFor() is done NOT by router
        when userB borrows currentAsset
-------------------- evm_revert: 0x2272
          ✓ tokens balances are correct (39ms)
-------------------- evm_revert: 0x228d
          ✓ #calculateBorrowValue
-------------------- evm_revert: 0x22a8
          ✓ #getBorrowAmount
-------------------- evm_revert: 0x22c3
          ✓ throws when userB wants to deposit
-------------------- evm_revert: 0x22de
          ✓ #withdraw (235ms)
         debt token integration tests
-------------------- evm_revert: 0x22f9
            ✓ should #mint debt token to userB
-------------------- evm_revert: 0x2314
            ✓ userB can #transfer debt (937ms)
            should #burn debt token on  repay
-------------------- evm_revert: 0x232f
              ✓ should #burn all debt when repay amountToBorrow and no interest apply  (51ms)
-------------------- evm_revert: 0x2360
              ✓ should NOT #burn all debt when repay amount without interest  (100ms)
-------------------- evm_revert: 0x237b
              ✓ should #burn all debt token on full repay (71ms)
            #transfer
-------------------- evm_revert: 0x239a
              ✓ throws when userA did not allow for transfer (157ms)
-------------------- evm_revert: 0x23b5
              ✓ throws when userB transfers debt to someone who has collateral in that asset (320ms)
-------------------- evm_revert: 0x23da
              ✓ throws when userA become insolvent after debt transfer from userB  (62ms)
-------------------- evm_revert: 0x2409
              ✓ throws when amount exceeds allowance (183ms)
          when userB borrows again
-------------------- evm_revert: 0x2428
            ✓ #liquidity is zero
-------------------- evm_revert: 0x244f
            ✓ lens borrow data are correct
-------------------- evm_revert: 0x2484
            ✓ tokens balances are correct
-------------------- evm_revert: 0x24b9
            ✓ there are no interests because no time passed
          when a week passed interests should  appear
            when all interests goes to the protocol
-------------------- evm_revert: 0x24ee
              ✓ #harvestProtocolFees (123ms)
-------------------- evm_revert: 0x2523
              ✓ userA do not have interests
```

```
            when protocol fees is 0%
-------------------- evm_revert: 0x2546
              ✓ userA got interests
-------------------- evm_revert: 0x2567
              ✓ accrueInterest() increases the total borrowAmount and deposits
-------------------- evm_revert: 0x2586
              ✓ total deposit increased by protocol interests
          #repay
-------------------- evm_revert: 0x25a5
            ✓ expect to repay all using exact amount (177ms)
-------------------- evm_revert: 0x25c4
            ✓ expect to repay all using max uint256 amount (175ms)
-------------------- evm_revert: 0x25df
            ✓ expect to repay all providing higher amount than actual debt  (173ms)
-------------------- evm_revert: 0x25fa
            ✓ expect to repay part of debt (111ms)
          #repayFor
-------------------- evm_revert: 0x2615
            ✓ anyone can repayFor(userB) if it is solvent (236ms)
            when userB becomes insolvent
-------------------- evm_revert: 0x2630
              ✓ anyone can repayFor(userB) if it is insolvent (180ms)
          #flashLiquidation
            when flashLiquidation is done on solvent userB
-------------------- evm_revert: 0x264b
              ✓ expect to not change assets, debt and collateral tokens balances for userB
-------------------- evm_revert: 0x2668
              ✓ expect totalBorrowAmount, totalDeposits of assets should not change
            when userB is NOT solvent
-------------------- evm_revert: 0x2685
              ✓ ltv > liquidationThreshold
              when flashLiquidation executed (interest ON)
-------------------- evm_revert: 0x26a2
                ✓ expect protocol got liquidation fees
-------------------- evm_revert: 0x26bf
                ✓ expect userB to be solvent, there is no debt (101ms)
-------------------- evm_revert: 0x26e4
                ✓ expect userB to loose his collateral
-------------------- evm_revert: 0x2709
                ✓ expect userB to have borrowed asset
                - expect liquidatorHelper to have some remaining quote token
-------------------- evm_revert: 0x272e
                ✓ expect userA earned fees on borrowed asset
-------------------- evm_revert: 0x2753
                ✓ expect interests to be applied
-------------------- evm_revert: 0x127c
  SiloFactory
    ✓ #siloFactoryPing
  SiloLens
    #protocolFees
      ✓ expect to return correct protocolFees
    #lensPing
      ✓ expect to return correct lensPing
    #getModel
      ✓ expect to return correct getModel
    when user deposit and borrow
      ✓ #liquidity
      ✓ #totalDeposits
      ✓ #collateralOnlyDeposits
      ✓ #totalBorrowAmount
      ✓ #borrowShare
      ✓ #totalBorrowShare
      ✓ #getBorrowAmount (47ms)
      ✓ #collateralBalanceOfUnderlying (50ms)
      ✓ #balanceOfUnderlying (78ms)
      ✓ #debtBalanceOfUnderlying
      ✓ #calculateCollateralValue (81ms)
      ✓ #calculateBorrowValue (68ms)
      ✓ #totalDepositsWithInterest (46ms)
      ✓ #totalBorrowAmountWithInterest (38ms)
      ✓ #getUtilization
      ✓ #borrowAPY
      #depositAPY
        ✓ expect to return 0 when no deposit (131ms)
        ✓ expect to calculate APY (86ms)
      LTV
        ✓ #getUserLTV (91ms)
        ✓ #getUserMaximumLTV (64ms)
        ✓ #getUserLiquidationThreshold (65ms)
    #hasPosition
      ✓ expect to return FALSE for address(0) (158ms)
      ✓ expect to return FALSE if user not using Silo (329ms)
      returns TRUE when user has at least one  position
        ✓ [0] expect to return TRUE for 1,0,0,0,0,0 (152ms)
        ✓ [1] expect to return TRUE for 0,1,0,0,0,0 (197ms)
        ✓ [2] expect to return TRUE for 0,0,1,0,0,0 (205ms)
        ✓ [3] expect to return TRUE for 0,0,0,1,0,0 (213ms)
        ✓ [4] expect to return TRUE for 0,0,0,0,1,0 (218ms)
        ✓ [5] expect to return TRUE for 0,0,0,0,0,1 (211ms)
  SiloRepository
    ✓ #defaultAssetConfig returns default values (62ms)
    ✓ isSilo()
    ✓ getMaximumLTV()
    ✓ getLiquidationThreshold()
  when deployed
    ✓ #bridgeAssets are setup
    ✓ #siloLatestVersion is 1st version
    ✓ #siloDefaultVersion is 1st version
    ✓ expect siloFactory(0) returns empty address
    ✓ expect silo factory is not empty for the default version
    ✓ #siloFactory returns address
    ✓ fees are 0
    ✓ #siloRepositoryPing
  #setFees
    throws when any fee is >= than 100%
      ✓ check for entryFee
      ✓ check for protocolShareFee
      ✓ check for protocolLiquidationFee
    when fees updated
      ✓ expect to saved fees
  #setNotificationReceiver
    ✓ expect to not have NotificationReceiver set
    ✓ throw when called NOT by owner
    with NotificationReceiver set
      ✓ expect to have NotificationReceiver set
  #setAssetConfig
    ✓ throws when ltv is zero
    ✓ throws when ltv == liquidationThreshold
    ✓ throws when ltv > liquidationThreshold
    ✓ throws when liquidationThreshold >= 100%
    ✓ throws when silo empty
    ✓ throws on empty interestRateModel
    ✓ throws when invalid interestRateModel
    ✓ throws when asset empty
    ✓ emits AssetConfigUpdate event (41ms)
    when config set
      ✓ expect to have valid values in storage
  setDefaultInterestRateModel()
    ✓ expect interest rate model is set in default config
    ✓ expect default interest rate model is set for random silo
  #setDefaultMaximumLTV
    ✓ expect to set new MaximumLTV
    ✓ throws when ltv is zero
    ✓ throws when ltv == liquidationThreshold
    ✓ throws when ltv > liquidationThreshold
  #setDefaultLiquidationThreshold
    ✓ expect to set new value
    ✓ throws when ltv == liquidationThreshold
    ✓ throws when ltv > liquidationThreshold
    ✓ throws when liquidationThreshold >= 100%
  #setPriceProvidersRepository
    ✓ expect to set repo address (88ms)
    ✓ throws on invalid address
    ✓ throws on empty address
  #setRouter
    ✓ expect to set repo address
    ✓ throws on invalid address
    ✓ throws on empty address
  #addBridgeAsset
    ✓ expect to revert when called NOT by owner
    ✓ expect to revert when price provider is not ready for asset
    ✓ expect to revert when empty asset
    ✓ emits BridgePool event when silo already exists for asset (228ms)
    when silo for newBridgeAsset already exists
      ✓ expect to add bridge asset and set bridge pool (90ms)
    when BridgePool exists
      when regular Silo exists for asset X
        ✓ throws when adding asset X as a bridge
    when new bridge asset added
      ✓ expect to have newAsset in bridgeAssets
      ✓ expect to revert when try to add same asset again
  #removeBridgeAsset
    ✓ expect to revert when called NOT by owner
    ✓ expect to revert when removing main bridge asset
    ✓ expect to revert when try to remove empty asset
    with 3 bridge assets
```

```
      ✓ expect to revert when asset does not exists
    when removed
      ✓ asset not exists as bridge asset
      ✓ asset exists as removed asset
    when silo for removing asset exists (it is bridge pool)
      ✓ expect to reset bridge pool on removal asset for existing silo(asset)  (86ms)
      ✓ does not reset bridgePool on removal asset that is not main bridgePool asset (94ms)
#newSilo
  ✓ throws when price provider not setup (40ms)
  ✓ throws if silo version does not exist (158ms)
  ✓ expect to create silo using default version (0) (174ms)
  ✓ emits BridgePool when created silo for bridge asset (150ms)
  when Silo created
    ✓ expect silo(asset) returns silo address
    ✓ expect siloReverse(siloAddress) returns asset
    ✓ expect isSilo(siloAddress) returns true
  with new silo version (not default)
    ✓ expect to create silo for OLD version (133ms)
    ✓ expect to create silo for NEW version (187ms)
#replaceSilo
  ✓ expect to throw when there is nothing to replace (silo not exists)
  when silo for asset exists
    ✓ expect to throw when called not by owner
    when replaced
      ✓ expect to replace silo
      ✓ expect siloReverse(newSilo) returns asset
      ✓ expect siloReverse(oldSilo) still returns asset
      ✓ expect isSilo(oldSilo) returns true
      ✓ expect isSilo(newSilo) returns true
#registerSiloVersion
  ✓ throws when called NOT by owner
  ✓ throws when empty factory
  ✓ throws when invalid factory
  ✓ expect to emit events (73ms)
  when silo version registered as NOT default version
    ✓ siloDefaultVersion NOT change
  when silo version registered as default version
    ✓ siloVersion is valid
    ✓ expect siloFactory(1) returns old version
    ✓ expect siloFactory(2) returns new version
#unregisterSiloVersion
  ✓ throws when NOT and owner
  ✓ throws when unregistering default version
  ✓ throws when unregistering nonexistent version
  ✓ emits event (92ms)
#setDefaultSiloVersion
  ✓ throw when NOT and owner
  ✓ throws when there is no factory for selected version
  ✓ expect to emit SiloDefaultVersion
  when default version set
    ✓ expect to have valid version
#ensureCanCreateSiloFor
  with just one bridge asset
    ✓ throws when asset is a bridge
  with many bridge assets
    ✓ throws when silo already exists for asset (135ms)
    ✓ allows to create when asset is a bridge asset
    ✓ allows to create when asset is NOT a bridge asset
    when asset is a bridge
      ✓ throws when bridge pool already exists (154ms)
      ✓ throws when bridge pool for other bridge asset already exists  (145ms)
SiloRouter unit tests
  when deployed
    ✓ wrappedNativeToken is set
  eth refunds
    ✓ refunds remaining eth if the user sent eth
    ✓ does not refund remaining eth if the user did not sent eth (40ms)
  execute single action
    ✓ Action.Deposit (222ms)
    ✓ Action.Withdraw (156ms)
    ✓ Action.Borrow (153ms)
    ✓ Action.Repay (157ms)
    using ETH
      Action.Deposit ETH
        ✓ expect to have correct ETH balance
      Action.Withdraw ETH
        ✓ expect to have correct ETH balance
      Action.Borrow ETH
        ✓ expect to have correct ETH balance
      Action.Repay ETH
        ✓ expect to have correct ETH balance
  execute bundle
    ✓ Action.Deposit => Action.Borrow (359ms)
    ✓ Action.Withdraw => Action.Action.Repay (355ms)
    ✓ Action.Deposit => Action.Borrow => Action.Withdraw => Action.Withdraw  (648ms)
    ✓ Action.Deposit => Action.Deposit => Action.Deposit => Action.Borrow (784ms)
    using ETH
      Action.Deposit ETH => Action.Deposit ETH
        ✓ expect to have correct ETH balance
TokensFactory
  #factory should create all types of tokens
    #createShareCollateralToken
      ✓ creates token
      ✓ silo is token deployer
    #createShareDebtToken
      ✓ creates token
      ✓ silo is token deployer
GuardedLaunch
  after deployment
    ✓ #globalToggle
    ✓ #defaultMaxLiquidity
  #getMaxSiloDepositsValue
    after deployment
      Test case 0
        ✓ expect correct max deposits
      Test case 1
        ✓ expect correct max deposits
      Test case 2
        ✓ expect correct max deposits
      Test case 3
        ✓ expect correct max deposits
    #toggleLimitedMaxLiquidity
      Test case 0
        ✓ expects no limit
      Test case 1
        ✓ expects no limit
      Test case 2
        ✓ expects no limit
      Test case 3
        ✓ expects no limit
    #setDefaultSiloMaxDepositsLimit
      Test case 0
        ✓ expects new deafult limit
      Test case 1
        ✓ expects new deafult limit
      Test case 2
        ✓ expects new deafult limit
      Test case 3
        ✓ expects new deafult limit
    #setSiloMaxDepositsLimit
      Test case 0
        ✓ expects new limit for a Silo
      Test case 1
        ✓ expects new limit for a Silo
      Test case 2
        ✓ expects new limit for a Silo
      Test case 3
        ✓ expects new limit for a Silo
  #isSiloPaused
    after deployment
      Test case 0
        ✓ expects Silo to be unpaused
      Test case 1
        ✓ expects Silo to be unpaused
      Test case 2
        ✓ expects Silo to be unpaused
      Test case 3
        ✓ expects Silo to be unpaused
  #setGlobalPause
    Test case 0
      ✓ expects Silo to be paused
    Test case 1
      ✓ expects Silo to be paused
    Test case 2
      ✓ expects Silo to be paused
    Test case 3
      ✓ expects Silo to be paused
    global unpause Silo
      Test case 0
        ✓ expects Silo to be unpaused
      Test case 1
        ✓ expects Silo to be unpaused
      Test case 2
        ✓ expects Silo to be unpaused
      Test case 3
        ✓ expects Silo to be unpaused
```

```
    #setSiloPause
      pause Silo
        Test case 0
          ✓ expects Silo to be paused
        Test case 1
          ✓ expects Silo to be paused
        Test case 2
          ✓ expects Silo to be paused
        Test case 3
          ✓ expects Silo to be paused
        unpause Silo
          Test case 0
            ✓ expects Silo to be unpaused
          Test case 1
            ✓ expects Silo to be unpaused
          Test case 2
            ✓ expects Silo to be unpaused
          Test case 3
            ✓ expects Silo to be unpaused
      pause Asset
        Test case 0
          ✓ expects asset to be paused
        Test case 1
          ✓ expects asset to be paused
        Test case 2
          ✓ expects asset to be paused
        Test case 3
          ✓ expects asset to be paused
        unpause Asset
          Test case 0
            ✓ expects Asset to be unpaused
          Test case 1
            ✓ expects Asset to be unpaused
          Test case 2
            ✓ expects Asset to be unpaused
          Test case 3
            ✓ expects Asset to be unpaused
ShareCollateralToken
  #mint
    ✓ expect balance
    with NotificationReceiver set
      ✓ expect balance
  #burn
    ✓ expect balance
  #transfer
    successful transfer
      ✓ expect correct balances
    throws when
      ✓ userA transfers collateral to someone who has debt in that asset  (61ms)
      ✓ userA become unsolvent after transfer (64ms)
  #transferFrom
    successful trasnfer
      with misconfigured NotificationReceiver
        ✓ expect correct balances
      with properly configured NotificationReceiver
        ✓ expect NotificationSent event with value true (204ms)
    throws when
      ✓ not enough allowance from userA (91ms)
      ✓ userC transfers userAs asset deposit to userB who has debt in that asset (63ms)
      ✓ userA become unsolvent after transferFrom to userB (67ms)
ShareDebtToken
  #mint
    ✓ expect balance
    with NotificationReceiver set
      ✓ expect balance
  #burn
    ✓ expect balance
  #transfer debt
    ✓ expect correct balances
    throws when
      ✓ recipient did not allow for transfer (65ms)
      ✓ userA transfers debt to someone who has collateral in that asset  (134ms)
      ✓ userB become unsolvent after debt transfer from userA (201ms)
  #transferFrom of debt
    ✓ expect correct balances
    throws when
      ✓ not enough allowance from userA to userC (who transfers)  (95ms)
      ✓ not enough receive allowance from userB to userA (71ms)
      ✓ userC transfers userAs debt to userB who has collateral in that asset (150ms)
      ✓ userB become insolvent after transferFrom debt from userA (174ms)
  #setReceiveApproval
    ✓ expect to set receive approval from random address
    ✓ throws when receive approval sender is 0x0
  #decreaseReceiveAllowance
    ✓ expect to decrease allowance by 25%
    ✓ reverts if decreasing receive allowance results in an underflow
  #increaseReceiveAllowance
    ✓ expect to increase allowance x3
    ✓ reverts if increasing receive allowance results in an overflow
ShareToken
  when share token is ShareCollateralToken
    when deployed
      ✓ expect to have name set
      ✓ expect to have symbol set
      ✓ expect to have silo set
      ✓ expect to have asset set
    #mint
      ✓ throws when mint by NOT an owner
      ✓ owner should mint tokens (39ms)
      ✓ should emit event on mint
    #burn
      ✓ throws when burn NOT by owner
      ✓ owner should burn tokens (53ms)
      ✓ should emit event on burn
  when share token is  ShareDebtToken
    when deployed
      ✓ expect to have name set
      ✓ expect to have symbol set
      ✓ expect to have silo set
      ✓ expect to have asset set
    #mint
      ✓ throws when mint by NOT an owner
      ✓ owner should mint tokens (40ms)
      ✓ should emit event on mint
    #burn
      ✓ throws when burn NOT by owner
      ✓ owner should burn tokens (50ms)
      ✓ should emit event on burn
  4764 passing (6m)
  17 pending
Done in 406.79s.
```

Code Coverage

Initial Audit:
Quantstamp usually recommends developers increase the branch coverage to `90%` and above before a project goes live, in order to avoid hidden functional bugs that might not be easy to spot during the development phase. For branch code coverage, the current targeted files by the audit achieve a lower score that can be improved further.
Reaudit update: Coverage could not be generated due to errors.
Final Reaudit: The final repository does not contain a test folder.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	96.41	83.18	94.85	96.52	
BaseSilo.sol	99.46	81.67	100	99.46	395
Error.sol	100	100	100	100	
InterestRateModel.sol	96.55	92.86	91.67	96.43	162,230
PriceProvidersRepository.sol	92.86	81.82	84.62	88.89	38,39,106

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
Silo.sol	100	100	100	100	
SiloFactory.sol	25	0	50	25	18,20,21
SiloLens.sol	92.19	33.33	96.15	95.16	311,312,314
SiloRepository.sol	96.83	90.7	96.77	96.72	117,122,123,232
SiloRouter.sol	95	75	85.71	94.59	80,118
TokensFactory.sol	100	100	100	100	
contracts/governance/	21.82	3.33	48.15	22.22	
SiloGovernanceToken.sol	60	100	60	60	27,51
SiloGovernor.sol	72.73	100	75	72.73	113,141,151
SiloSnapshotWrapper.sol	0	0	0	0	… 42,43,52,57
TreasuryVester.sol	3.7	4.17	25	3.85	… 6,98,99,102
contracts/interfaces/	100	100	100	100	
IBaseSilo.sol	100	100	100	100	
IERC20R.sol	100	100	100	100	
IFlashLiquidationReceiver.sol	100	100	100	100	
IGuardedLaunch.sol	100	100	100	100	
IInterestRateModel.sol	100	100	100	100	
INotificationReceiver.sol	100	100	100	100	
IPriceProvider.sol	100	100	100	100	
IPriceProvidersRepository.sol	100	100	100	100	
IShareToken.sol	100	100	100	100	
ISilo.sol	100	100	100	100	
ISiloFactory.sol	100	100	100	100	
ISiloRepository.sol	100	100	100	100	
ISwapper.sol	100	100	100	100	
ITokensFactory.sol	100	100	100	100	
IWrappedNativeToken.sol	100	100	100	100	
contracts/lib/	95.8	90.32	100	96.45	
EasyMath.sol	100	100	100	100	
ModelStats.sol	66.67	50	100	100	
PRBMathCommon.sol	100	98.51	100	100	
PRBMathSD59x18.sol	61.54	40	100	66.67	42,72,73,77,78
Ping.sol	90	83.33	100	100	
Solvency.sol	94.37	71.43	100	96.97	317,345
TokenSymbol.sol	100	100	100	100	
contracts/liquidation/	77.19	58.33	70.37	79.28	
BalancerV2Swap.sol	72.22	50	71.43	72.22	37,38,75,79,83
LiquidationHelper.sol	80.77	60.71	75	84	… ,93,150,187
UniswapV3Swap.sol	66.67	50	62.5	66.67	… 5,66,98,109
contracts/mock/	75	66.67	70.59	73.68	
Forwarder.sol	100	100	100	100	
MockERC20.sol	80	100	75	75	16
MockLiquidationHelper.sol	100	100	100	100	
MockPriceFetchersRepository.sol	0	0	0	0	15,20,24,28
MockSiloGovernor.sol	100	100	100	100	
TestTokenSymbol.sol	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/priceProviders/	80	75	66.67	83.33	
PriceProvider.sol	80	75	66.67	83.33	37
contracts/priceProviders/balancerV2/	96.36	85.71	100	97.87	
BalancerV2PriceProvider.sol	96.36	85.71	100	97.87	213
contracts/priceProviders/uniswapV3/	78.46	66.67	65	80.65	
TwoStepOwnable.sol	26.67	0	22.22	25	… 68,69,76,83
UniswapV3PriceProvider.sol	94	80	100	100	
contracts/utils/	89.66	81.82	87.8	90.22	
ERC20R.sol	100	75	100	100	
GuardedLaunch.sol	100	100	100	100	
Managable.sol	100	62.5	100	100	
ShareCollateralToken.sol	100	100	100	100	
ShareDebtToken.sol	100	100	100	100	
ShareToken.sol	100	100	100	100	
TwoStepOwnable.sol	40	33.33	44.44	43.75	… 76,77,78,92
All files	89.09	78.64	84.84	89.32	

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Contracts

```
95998c708ca730ad17a740d12580e6e8499248e136e2ae7e040bca801b7ca897 ./contracts/BaseSilo.sol
5af579ca0bb8f7e1af427d4ee34cf60ad45b69ca055b1cc7771fb8504b1df753 ./contracts/SiloLens.sol
6b77f13f4cf726a1b83e81e41d1d45358786e4f956cfe19d7c3acdd91f276888 ./contracts/Error.sol
7e0d7b9543cea347ce116a5202bd5904b33906bd1cdaf2503ff5448e8825c137 ./contracts/InterestRateModel.sol
a1712d2f20bad4c2ddf6573bc4a9cff5be61a4aae420000a7c79b931ae2a4fd3 ./contracts/SiloFactory.sol
1d36412c302e23311ba9378f2a999b18fc5de168151493cb6b94032e98b5322b ./contracts/TokensFactory.sol
7fad817378ed28945217a178d4bc77be08c8935e221ec399fbb16ca91467123c ./contracts/PriceProvidersRepository.sol
637bfa0ab1537140aa5879bc031faae7cb5578fccff6e70040eb76ba63afefb7 ./contracts/SiloRepository.sol
ea5263b309b5a552a790e66bc2119a0826dc62dc18598440192bec060580bf5b ./contracts/Silo.sol
d54d31b0b2438b557f18bed1a1b57b2d1184674b771102b5ce7c1d76b618da34 ./contracts/SiloRouter.sol
de98a7a26587eea251fd8bef24a52bac3e373e2859d542bd14eda97984af2fb3 ./contracts/interfaces/IPriceProvider.sol
047b4735a6a6cc60ab0d6cae7794c9444d87ad5687bc718e1a011b0ae844606d ./contracts/interfaces/ITokensFactory.sol
17a30a4284973cfece06b03591188815bb66b92c79fb1eb6359ff8fcf313d5b1 ./contracts/interfaces/ISiloFactory.sol
14b3c6f52d35ca18b2f1ff876c5e74d15e5ba7b70e01d605baaf0b008a42386a ./contracts/interfaces/IBaseSilo.sol
1ceea4102e8104d2a2ce6f2cbee33aae2f80d1f2100b2f44d50ed749f00834ec ./contracts/interfaces/ISwapper.sol
6680d6110eee287fec4d3dc7b0972adf3937e0866ff509abda3d31d3fe868bb3 ./contracts/interfaces/INotificationReceiver.sol
27524d73f18a0ab38f8909aae9c53b970f915199b4fd0f02f5c9d5263551c201 ./contracts/interfaces/ISilo.sol
5a7d80227dddb4a61fcc4207f842ce63edd93ea0113067299ea7233a4fc22ae7 ./contracts/interfaces/IERC20R.sol
944b0195011f61cef63ac572ff50c8f8d1b1222fcb3c6be39b16d6926ccfa1f2 ./contracts/interfaces/IShareToken.sol
7f2f23cc49df2cda649d7f95b487de6090df366b91279abf8f047d4b57a38dc7 ./contracts/interfaces/IWrappedNativeToken.sol
b2d7a77a5f0bcc5fcf8b3eb06a5995ae72076bcb26a74e3174640f91d72896f1 ./contracts/interfaces/IPriceProvidersRepository.sol
1596843103d3831905ea3cea0d92e977cff6602b37cd3af6a8adedfb3651da5f ./contracts/interfaces/ISiloRepository.sol
0b7f5150822daf384f52e3223484af1bc9da2d2139ffa31e8f6c4be447474734 ./contracts/interfaces/IGuardedLaunch.sol
3d9ac9cea947c2e505f5c72688fdcbad1dffcf8ed9aa4fa78e642b98b713a0a9 ./contracts/interfaces/IInterestRateModel.sol
b1ef3e15efeb41b2f9ec885ec11747f0494b994be1a322fd746b089671ac8fe0 ./contracts/utils/GuardedLaunch.sol
86edfc47938af9aefc034973923e296f3d846ed1cf3d379ce6d9910c52a5bbcc ./contracts/utils/ShareCollateralToken.sol
4ea1540f14d3d1d77132b96fa7538fa398cbdab04bd435ea17f5870d01625fc7 ./contracts/utils/ShareDebtToken.sol
80aa4d19515112dfbe241f54655ceb3e86bd8c16bd9b8f2ef6f576cae4d0df28 ./contracts/utils/Managable.sol
86c038f9812be9ed96c9f7149e63b93ebe0530487c90acf5a51d21029af7a707 ./contracts/utils/ShareToken.sol
471cd799d98b153ac83e17378129ff08deb78283dbde16d0d2a70a3c66c41f5f ./contracts/utils/TwoStepOwnable.sol
43b0dc0965ed48e09767b3e4aa5492dc38f0312343ed0a31518ed59cba43422d ./contracts/utils/ERC20R.sol
8b5b262aae8a6a694675799eb7f1aaaf0e88f9c46ae2ec70dce664e2a10534a8 ./contracts/mock/MockSiloGovernor.sol
```

8431b80a5ffb2b11b13702de3cafb644b7e22e6e10119ee1979781d576c14de0 ./contracts/mock/Forwarder.sol

04b207c6257e306d8004ebc88cb573f0bcb76862c7f7306a47ad1809915c59da ./contracts/mock/MockPriceFetchersRepository.sol

1b9336c61db1b16e16644918e113ae554e40e901056d9d76d98c5f583a1fca50 ./contracts/mock/MockLiquidationHelper.sol

114b63b53e132f75dfa52493b1c93b264286b2a91dcdd530d52098e1c9e3a473 ./contracts/mock/TestTokenSymbol.sol

4c76500bb82c2569b6894cc441cfe5f13d11ed4afce268eb39c7a877c3c22ef1 ./contracts/mock/MockERC20.sol

ddddb175f5a57dd49614d308b0bf2b9901f5444d2673cc0dc1693274f87b432a ./contracts/lib/PRBMathSD59x18.sol

c435e569b2bdf9e862786552bd2a272118614f420ba6d520608cd72f79109ae9 ./contracts/lib/ModelStats.sol

600acc4dee6f85ff5c187159f5e6ec18b991de8df84da11615c68d6bd1706622 ./contracts/lib/PRBMathCommon.sol

e79a4d1aab098b2a4210ad3478c50b133c8aa665c8c148a5435fe4cbbd13c4f3 ./contracts/lib/Solvency.sol

eba7dd8c38c3f15145582527f4ec6f21845e760b51335bded945983bc4561641 ./contracts/lib/Ping.sol

9d072017a41c43bfb4389357665ae53cfde8a707397fab69c5a280d7dc9906a3 ./contracts/lib/TokenSymbol.sol

cede0685c50e09da38091c8d63d800dcb8a0023ece5fcee5416b369a9ae41ad7 ./contracts/lib/EasyMath.sol

bd8f04a14bc6adcf5ed7a628cdf32feb3d63128d3dd7c281b6bdd18d0803bec7 ./contracts/governance/SiloGovernanceToken.sol

3b7531f754d56e0a5a267ffd133f6eda592e1a2916eda1eb776a107661bfd0a6 ./contracts/governance/SiloGovernor.sol

19b250c00bb6b1a1f7bae02286f04cf48ba8123fdbbaeec43eeb4d4c98a96aba ./contracts/priceProviders/PriceProvider.sol

ce8f4e4a92715a7d7e56aa04db1c98514def5a42eaebb5b4b2d01ba20284f025 ./contracts/priceProviders/uniswapV3/UniswapV3PriceProvider.sol

192d4baa971aa3ad3d9c9cec016b35f227bd557fd2542746f667d385c1e1a29a ./contracts/priceProviders/uniswapV3/TwoStepOwnable.sol

bcd942810a06a6a064dd23c9477f258d14327f76023f7eec3f35fa8d241313fb ./contracts/priceProviders/balancerV2/BalancerV2PriceProvider.sol

Tests

20b4c1b0cf75c9deee7969dfeb6f80a5d373bf69e9093ddfa0ed6b34fddf3339 ./test/InterestRateModel.unit.test.ts

a2c9a6fda2cef370a2a495b3f7a5d694e22fe052d2f23ba9f4c59c49e8dcdbfc ./test/SiloLens.unit.test.ts

144edb49a43e1de7078eee7d32eed116cb396b077369940ef0b7f2d352145e4b ./test/SiloRouter.unit.test.ts

039480fd26aaa03fcc6f9282b22bcfd8410fad2af7486881575058b3021716c0 ./test/TokensFactory.unit.test.ts

1d334ead62f99cc3cbf60ac7a9d89061ed2e9fcf4f95a6fc0edae6db68cf1fa3 ./test/Silo.unit.test.ts

6efd18dd973ea959638316838fd3fe3a4d56b07a3742e1035d38225bda83bd31 ./test/SiloScenarios.test.ts

fd0d53cdb874794cd70e0b1bb5e3efe575869f92a62852686e35dbeae4da0be8 ./test/SiloFactory.unit.test.ts

a1f55c550f9c4444d3a45e9ba85003e7e5f612160f07551f70460d5bdb1a81a3 ./test/deployments.integration.test.ts

829d94fdf20204399a702b53c68e4da626ccd25ff185d9085fd84baf31bcfe76 ./test/PriceProvidersRepository.unit.test.ts

dac3441f6b7c4b7e9e3731a79470695eb25dc756f53f7d27fe1ce8a08543f63b ./test/BaseSilo.test.ts

4544ebbecde9bfc102002e748677f9528da57288ecc6be856797ea8e81599fcf ./test/SiloRepository.unit.test.ts

245d97a008fcd22c061f09540c599b5322958d65c8c8be11191bf33a6758bb35 ./test/SiloRouter.integration.test.ts

79fd05b54490d275d489b71726535d27e9889a224f7b0c930b35bed510446b76 ./test/helpers/mocks.ts

97888bd820ffe59b8c8b88f039860c603ed28b07dad344ee50541d4bf8fcf3e5 ./test/helpers/utils.ts

104d36346081389f8757e5b9f57c5ac1a307a3b1fb22ecc73385ee1468cefb35 ./test/helpers/index.ts

538214c18ca937e12576a10ce253fa35c3835dfface7da3c17b693e7cf09a557 ./test/helpers/time.ts

4a0a0d98c9533abfb3cc98346e6ed90c1a1a5ddde16fcf21815a451c8f556f1e ./test/helpers/intelliJRequirements.ts

f04b2439f62e92833560aae1833a5b95d5f9bec13b2eb2b2572648062ddac6b3 ./test/helpers/assertions.ts

52e503c2719e84c332e11ba3168b781aef2fd88055875e6fd0811deb3dcd522a ./test/helpers/erc20.ts

daa882bad515903899b82814abf2f9a823311930ad3ccccc4537068ad437e5eb ./test/helpers/constants/rinkeby.ts

36e47bebbb69f4825b42777ec7193db66399d41eecf04ae1ec4dcf344cb905b7 ./test/helpers/constants/eth.ts

929d9a678a82be2b0f3bd188558e72284fd703d217d76bd9b6aa41e1a29825af ./test/helpers/constants/polygon.ts

b5a1d1da98dbd8d5395819b86b6e9af8af161bd908bce63792ecc7b493eb4375 ./test/liquidation/LiquidationHelper.test.ts

db699f7fbc14d43222ea1f1c8c49cc3967c0404fd937784e65af04f9ff652e9f ./test/utils/ShareDebtToken.unit.test.ts

83505d2477011c4bd3e1c1e6bcbbcff396d45f7c7a910f3274620e090e01430b ./test/utils/ShareCollateralToken.unit.test.ts

6aa565f8e7c70afbffd10a8621dc9ac02f4488a23e62bb4f052818948a91064d ./test/utils/UniswapV3Swap.test.ts

e8787290a7401f421e6d2a5184831b77b01fd4f7999d7f8d74b550e56134f61a ./test/utils/GuardedLaunch.unit.test.ts

6aa565f8e7c70afbffd10a8621dc9ac02f4488a23e62bb4f052818948a91064d ./test/utils/BalancerV2Swap.test.ts

85ab991ad04cdefcb8f3b071a9ed2345ea2d58538d2ea39d0f258c8b30a6de23 ./test/utils/ShareToken.unit.test.ts

b9438a6ad0c79fa7dbfc8557f0453c173f16dec5be779eab1ee86f13322503a2 ./test/utils/ERC20R.unit.test.ts

d889c1074d4a3686e3ec39a304fb32ebe5bb84be2df331d19fa83f4e2ad37cd7 ./test/utils/common/ISwapper.test.ts

5aa8aab52fada732ce9bf57f9fd9793c9065b337bcb34c1c3d019d8d69fc0e84 ./test/governance/SiloSnapshotWrapper.intergarion.test.ts

e9455b92a314d77ca02cd32c4c405cb71b5777b853a318832b11ad59789f472d ./test/governance/SiloGovernor.unit.test.ts

a900c0c2aa3ed8a41360cbab402de41f0b05cbbfec1c24d41900838110b27798 ./test/governance/TreasuryVester.integration.test.ts

74e213f9e50919c27425711659d9eeec52df0dc2124fa08cb8e178b83d443192 ./test/governance/SiloGovernanceToken.unit.test.ts

4a5056e438be959447e22a116e7a676f19a71e606a028714a829c99871705822 ./test/priceProviders/UniswapV3.unit.test.ts

6926b3ceb75b066126e1d6ab5ae524b0a2f70a6ef4f5b1edd0dc07b73e34a692 ./test/priceProviders/common.integration.test.ts

eda7ad84a6f71a789be21b78d638d963f63a18915da60f878a9cb7915ef2ceb6 ./test/priceProviders/BalancerV2.unit.test.ts

Changelog

- 2024-10-6 - Initial report
- 2024-11-27 - Reaudit update (4c9e45c)
- 2024-12-14 - Final reaudit (4be2bdd)

About Quantstamp

Quantstamp is a Y Combinator-backed company that helps to secure blockchain platforms at scale using computer-aided reasoning tools, with a mission to help boost the adoption of this exponentially growing technology.

With over 1000 Google scholar citations and numerous published papers, Quantstamp's team has decades of combined experience in formal verification, static analysis, and software verification. Quantstamp has also developed a protocol to help smart contract developers and projects worldwide to perform cost-effective smart contract security scans.

To date, Quantstamp has protected $5B in digital asset risk from hackers and assisted dozens of blockchain projects globally through its white glove security assessment services. As an evangelist of the blockchain ecosystem, Quantstamp assists core infrastructure projects and leading community initiatives such as the Ethereum Community Fund to expedite the adoption of blockchain technology.

Quantstamp's collaborations with leading academic institutions such as the National University of Singapore and MIT (Massachusetts Institute of Technology) reflect our commitment to research, development, and enabling world-class blockchain security.