X-AI CONSULTING

SMART CONTRACT AUDIT

X.X-AI.CYOU

Findings

ID	Severity	Category	Status
CVF-1	Minor	Procedural	Info
CVF-2	Minor	Unclear behavior	Info
CVF-3	Minor	Unclear behavior	Info
CVF-4	Minor	Bad datatype	Info
CVF-5	Major	Flaw	Fixed
CVF-6	Minor	Suboptimal	Info
CVF-7	Minor	Flaw	Fixed
CVF-8	Minor	Bad naming	Info
CVF-9	Minor	Bad datatype	Info
CVF-10	Minor	Bad datatype	Info
CVF-11	Minor	Bad datatype	Info
CVF-12	Minor	Bad datatype	Info
CVF-13	Minor	Bad datatype	Info
CVF-14	Minor	Bad datatype	Info
CVF-15	Minor	Bad datatype	Info
CVF-16	Minor	Unclear behavior	Fixed
CVF-17	Minor	Flaw	Info
CVF-18	Minor	Bad datatype	Info
CVF-19	Major	Suboptimal	Info
CVF-20	Minor	Suboptimal	Info
CVF-21	Minor	Readability	Info
CVF-22	Minor	Procedural	Info
CVF-23	Minor	Overflow/Underflow	Info
CVF-24	Minor	Overflow/Underflow	Info
CVF-25	Minor	Overflow/Underflow	Info
CVF-26	Minor	Bad datatype	Info
CVF-27	Minor	Procedural	Info

ID	Severity	Category	Status
CVF-28	Major	Flaw	Fixed
CVF-29	Minor	Unclear behavior	Info
CVF-30	Minor	Suboptimal	Info
CVF-31	Minor	Suboptimal	Info
CVF-32	Moderate	Flaw	Info
CVF-33	Minor	Suboptimal	Info
CVF-34	Minor	Readability	Info
CVF-35	Minor	Procedural	Info
CVF-36	Minor	Procedural	Info
CVF-37	Minor	Procedural	Info
CVF-38	Minor	Suboptimal	Info
CVF-39	Minor	Suboptimal	Info
CVF-40	Minor	Bad datatype	Fixed
CVF-41	Minor	Suboptimal	Info
CVF-42	Major	Unclear behavior	Info
CVF-43	Minor	Suboptimal	Info
CVF-44	Minor	Suboptimal	Info
CVF-45	Minor	Bad naming	Info
CVF-46	Minor	Suboptimal	Info
CVF-47	Minor	Suboptimal	Info
CVF-48	Minor	Readability	Info
CVF-49	Minor	Bad datatype	Info
CVF-50	Minor	Bad datatype	Info
CVF-51	Minor	Suboptimal	Info
CVF-52	Major	Suboptimal	Info
CVF-53	Minor	Suboptimal	Info
CVF-54	Major	Suboptimal	Info
CVF-55	Minor	Bad datatype	Info
CVF-56	Minor	Flaw	Info
CVF-57	Minor	Bad datatype	Info

ID	Severity	Category	Status
CVF-58	Minor	Suboptimal	Info
CVF-59	Minor	Procedural	Info
CVF-60	Minor	Overflow/Underflow	Info
CVF-61	Minor	Procedural	Info
CVF-62	Minor	Suboptimal	Info
CVF-63	Minor	Suboptimal	Info
CVF-64	Minor	Overflow/Underflow	Info
CVF-65	Minor	Procedural	Info
CVF-66	Minor	Procedural	Info
CVF-67	Minor	Bad datatype	Info
CVF-68	Minor	Bad datatype	Info
CVF-69	Minor	Bad datatype	Info
CVF-70	Minor	Suboptimal	Info
CVF-71	Minor	Suboptimal	Info
CVF-72	Minor	Bad datatype	Info

Contents

1 Document properties

Version

Version	Date	Author	Description
0.1	July 21, 2022	D. Khovratovich	Initial Draft
0.2	July 21, 2022	D. Khovratovich	Minor revision
1.0	July 22, 2022	D. Khovratovich	Release

Contact

D. Khovratovich

khovratovich@gmail.com

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting at the customer request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations.
We have reviewed the following contracts at repository:

- interfaces/IBaseSilo.sol
- interfaces/IERC20R.sol
- interfaces/IFlashLiquidationReceiver.sol
- interfaces/IGuardedLaunch.sol
- interfaces/IInterestRateModel.sol
- interfaces/INotificationReceiver.sol
- interfaces/IPriceProvider.sol
- interfaces/IPriceProvidersRepository.sol
- interfaces/IShareToken.sol
- interfaces/ISilo.sol
- interfaces/ISiloFactory.sol
- interfaces/ISiloRepository.sol
- interfaces/ISwapper.sol
- interfaces/ITokensFactory.sol
- interfaces/IWrappedNativeToken.sol
- lib/EasyMath.sol
- lib/ModelStats.sol
- lib/Ping.sol
- lib/PRBMathCommon.sol
- lib/PRBMathSD59x18.sol
- lib/Solvency.sol
- lib/TokenSymbol.sol
- priceProviders/balancerV2/BalancerV2PriceProvider.sol
- priceProviders/uniswapV3/TwoStepOwnable.sol

- priceProviders/uniswapV3/UniswapV3PriceProvider.sol
- priceProviders/PriceProvider.sol
- utils/ERC20R.sol
- utils/GuardedLaunch.sol
- utils/Managable.sol
- utils/ShareCollateralToken.sol
- utils/ShareDebtToken.sol
- utils/ShareToken.sol
- utils/TwoStepOwnable.sol
- BaseSilo.sol
- Error.sol
- InterestRateModel.sol
- PriceProvidersRepository.sol
- Silo.sol
- SiloFactory.sol
- SiloLens.sol
- SiloRepository.sol
- SiloRouter.sol
- TokensFactory.sol

2.1 About ABDK

ABDK Consulting, established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function. The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

2.2 Disclaimer

Note that the performed audit represents current best practices and smart contract standards which are relevant at the date of publication. After fixing the indicated issues the smart contracts should be re-audited.

2.3 Methodology

The methodology is not a strict formal procedure, but rather a collection of methods and tactics that combined differently and tuned for every particular project, depending on the project structure and and used technologies, as well as on what the client is expecting from the audit. In current audit we use:

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows code best practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.

- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places and that their visibility scopes and access levels are relevant. At this phase we understand overall system architecture and how different parts of the code are related to each other.

- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and is done properly. At this phase we understand user roles and permissions, as well as what assets the system ought to protect.

- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check that code actually does what it is supposed to do, that algorithms are optimal and correct, and that proper data types are used. We also check that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

3 Detailed Results

3.1 CVF-1

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** INotificationReceiver.sol

Recommendation Consider specifying as "^0.8.0" to avoid changing compiler version in all files at each compiler upgrade. Also relevant for next files: TokensFactory.sol, SiloRouter.sol, SiloRepository.sol, SiloFactory.sol, Silo.sol, PriceProvidersRepository.sol, InterestRateModel.sol, SiloLens.sol, BaseSilo.sol, ShareToken.sol, ShareDebtToken.sol, ShareCollateralToken.sol, Managable.sol, LiquidationReentrancyGuard.sol, InterestRateDataResolver.sol, GuardedLaunch.sol, TokenHelper.sol, Solvency.sol, PRBMathSD59x18.sol, PRBMathCommon.sol, EasyMath.sol, IWrappedNativeToken.sol, ITokensFactory.sol, ISiloFactory.sol, ISilo.sol, ISiloRepository.sol, IShareToken.sol, IInterestRateModel.sol, IGuardedLaunch.sol, IFlashLiquidationReceiver.sol, IERC20R.sol, IBaseSilo.sol.
Client Comment Acknowledged.

Listing 1:

```
3 +pragma solidity 0.8.13;
```

3.2 CVF-2

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** TokensFactory.sol

Description This function should emit some event.
Client Comment Acknowledged.

Listing 2:

```
24 +function initRepository(address _repository) external {
```

3.3 CVF-3

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** TokensFactory.sol

Recommendation As zero repository address has a special meaning, consider adding an explicit check to ensure that "_repository" is not zero.
Client Comment Acknowledged.

Listing 3:

```
26 +if (address(_siloRepository) != address(0)) revert
     ↪ SiloRepositoryAlreadySet();
```

3.4 CVF-4

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** SiloRouter.sol

Recommendation The type of the "_wrappedNativeToken" argument should be "IWrappedNativeToken". The type of the "_siloRepository" argument should be "ISiloRepository".
Client Comment Acknowledged.

Listing 4:

```
53 +constructor (address _wrappedNativeToken, address
    ↪ _siloRepository) {
```

3.5 CVF-5

- **Severity** Major
- **Status** Fixed
- **Category** Flaw
- **Source** SiloRouter.sol

Description This allows anybody to take any ether residing on the contract's balance before the transaction.
Recommendation Consider returning only the leftover from what the user provided, i.e. return: balanceAfter - balanceBefore.
Client Comment Leftover is returned to msg.sender so Router should never hold any token or ETH.

Listing 5:

```
98 +if (msg.value != 0 && address(this).balance != 0) {
```

3.6 CVF-6

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** SiloRouter.sol

Recommendation This condition could be simplified as: !success || data.length != 0 && abi.decode(data, (bool))
Client Comment Acknowledged.

Listing 6:

```
178 +if(!success || !(data.length == 0 || abi.decode(data, (bool))))
    ↪ {
```

3.7 CVF-7

- **Severity** Minor
- **Status** Fixed
- **Category** Flaw
- **Source** SiloRouter.sol

Description This allows a user to use ether residing at the contracts' balance before the transaction.
Recommendation Consider adding an explicit check that: balanceAfter >= balanceBefore - msg.value, i.e. that the user didn't decrease the original contract's balance.
Client Comment Leftover is returned to msg.sender so Router should never hold any token or ETH.

Listing 7:

```
204 +if (msg.value != 0 && _asset == wrappedNativeToken) {
```

3.8 CVF-8

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** SiloRepository.sol

Description The name is more appropriate for a getter function, rather than for a variable.
Recommendation Consider renaming to something like "assetToVersion".
Client Comment Acknowledged.

Listing 8:

```
67 +mapping(address => uint128) public override getVersionForAsset;
```

3.9 CVF-9

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** SiloRepository.sol

Recommendation The type of this variable should be: mapping (IERC20 => uint128)
Client Comment Acknowledged.

Listing 9:

```
67 +mapping(address => uint128) public override getVersionForAsset;
```

3.10 CVF-10

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** SiloRepository.sol

Recommendation The type of this argument should be "ISiloFactory".
Client Comment Acknowledged.

Listing 10:

```
156 address _siloFactory,
```

3.11 CVF-11

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** SiloRepository.sol

Recommendation The type of this argument should be "ITokensFactory".
Client Comment Acknowledged.

Listing 11:

```
157 address _tokensFactory,
```

3.12 CVF-12

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** SiloRepository.sol

Recommendation The type of this argument should be "IERC20[]".
Client Comment Acknowledged.

Listing 12:

```
160 address[] memory _initialBridgeAssets
```

3.13 CVF-13

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** SiloRepository.sol

Recommendation The return type should be "ISilo".
Client Comment Acknowledged.

Listing 13:

```
657 +) internal returns (address createdSilo) {
```

3.14 CVF-14

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** SiloFactory.sol

Recommendation The type of this variable should be "ISiloRepository".
Client Comment Acknowledged.

Listing 14:

```
17 +address private _siloRepository;
```

3.15 CVF-15

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** SiloFactory.sol

Recommendation The argument type should be "ISiloRepository".
Client Comment Acknowledged.

Listing 15:

```
23 +function initRepository(address _repository) external {
```

3.16 CVF-16

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** SiloFactory.sol

Description This function should emit some event.
Client Comment Event emitted.

Listing 16:

```
23 +function initRepository(address _repository) external {
```

3.17 CVF-17

- **Severity** Minor
- **Status** Info
- **Category** Flaw
- **Source** SiloFactory.sol

Recommendation Once zero repository address has a special meaning, consider adding an explicit check that "_repository" is not zero.
Client Comment Acknowledged.

Listing 17:

```
25 +if (_siloRepository != address(0)) revert RepositoryAlreadySet
    ↪ ();
```

3.18 CVF-18

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** SiloFactory.sol

Recommendation The return type should be 'ISiloRepository".
Client Comment Acknowledged.

Listing 18:

```
35 +function createSilo(address _siloAsset, uint128 _version, bytes
    ↪  memory) external override returns (address silo) {
```

3.19 CVF-19

- **Severity** Major
- **Status** Info
- **Category** Suboptimal
- **Source** PriceProvidersRepository.sol

Description Requiring the quote token to always have exactly 18 decimals is quite limiting. This would not allow using USDT as the quote token.
Recommendation Consider supporting quote tokens with arbitrary number of decimals.
Client Comment Acknowledged.

Listing 19:

```
67 +if (TokenHelper.assertAndGetDecimals(_quoteToken) !=
    ↪ QUOTE_TOKEN_DECIMALS) {
```

3.20 CVF-20

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** PriceProvidersRepository.sol

Description In ERC-20 the "decimals" property is used by UI to render token amounts in a human-friendly way. Using this property in smart contracts is discouraged.
Recommendation Consider treating all token amounts as integers.
Client Comment Acknowledged.

Listing 20:

```
67 +if (TokenHelper.assertAndGetDecimals(_quoteToken) !=
   ↪ QUOTE_TOKEN_DECIMALS) {
```

3.21 CVF-21

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** SiloLens.sol

Description Thus makes code harder to read. Does this really save any gas?
Client Comment Acknowledged.

Listing 21:

```
283 +unchecked {
+       i++;
+}
```

3.22 CVF-22

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** SiloLens.sol

Description Here all the assets are loaded from the storage into the memory, while it would be much more efficient to stop loading after finding the first non-zero asset.
Recommendation Consider refactoring.
Client Comment Acknowledged.

Listing 22:

```
297 +(, ISilo.AssetStorage[] memory assetsStorage) = _silo.
   ↪ getAssetsWithState();
```

3.23 CVF-23

- **Severity** Minor
- **Status** Info
- **Category** Overflow/Underflow
- **Source** SiloLens.sol

Description Phantom overflow is possible here.
Recommendation Consider using the "muldiv" function.
Client Comment Acknowledged.

Listing 23:

```
348 +uint256 generatedDebtAmount = totalBorrowAmountWithInterest(
    ↪ _silo, _asset) * borrowAPY(_silo, _asset) / dp;

351 +return generatedDebtAmount * Solvency._PRECISION_DECIMALS /
    ↪ totalDepositsAmount;
```

3.24 CVF-24

- **Severity** Minor
- **Status** Info
- **Category** Overflow/Underflow
- **Source** SiloLens.sol

Recommendation Consider using the "muldiv" function to avoid overflow.
Client Comment Acknowledged.

Listing 24:

```
363 +// If we overflow on multiplication it should not revert tx, we
    ↪  will get lower fees
+return _amount * entryFee / Solvency._PRECISION_DECIMALS;
```

3.25 CVF-25

- **Severity** Minor
- **Status** Info
- **Category** Overflow/Underflow
- **Source** BaseSilo.sol

Description Phantom overflow is possible here.
Recommendation Consider using the "muldiv" function.
Client Comment Acknowledged.

Listing 25:

```
109 +     priceProviderRepo.getPrice(_asset) * allDeposits / (10 **
    ↪ IERC20Metadata(_asset).decimals()) >

898 +accruedInterest = totalBorrowAmountCached * rcomp / Solvency.
    ↪ _PRECISION_DECIMALS;

902 +     protocolShare = accruedInterest * protocolShareFee /
    ↪ Solvency._PRECISION_DECIMALS;
```

3.26 CVF-26

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** BaseSilo.sol

Recommendation The type of the "_asset" argument should be "IERC20".
Client Comment Acknowledged.

Listing 26:

```
250 +function liquidity(address _asset) public view returns (uint256
    ↪ ) {

337 +function _generateSharesNames(address _asset, bool
    ↪ _isBridgeAsset)
```

3.27 CVF-27

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** BaseSilo.sol

Recommendation The value of "_asset" should be casted to the "IERC20" interface rather than to a particular implementation of this interface.
Client Comment Acknowledged.

Listing 27:

```
251 +return ERC20(_asset).balanceOf(address(this)) - _assetStorage[
    ↪ _asset].collateralOnlyDeposits;
```

3.28 CVF-28

- **Severity** Major
- **Status** Fixed
- **Category** Flaw
- **Source** BaseSilo.sol

Description This function doesn't check whether the asset is already initialized, thus it could be used to initialize the same asset several times.
Recommendation Consider adding an explicit check to prevent this.
Client Comment Check is made outside of the function.

Listing 28:

```
300 +_allSiloAssets.push(_asset);
```

3.29 CVF-29

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** BaseSilo.sol

Description This event is emitted even if the asset status didn't change.
Client Comment Acknowledged.

Listing 29:

```
329 +emit AssetStatusUpdate(bridgeAsset, AssetStatus.Active);
```

3.30 CVF-30

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** BaseSilo.sol

Description Here concatenations calculated above are concatenated again, which means that their content is copied once again.
Recommendation Consider refactoring the code to construct token metadata without double copying data.
Client Comment Acknowledged.

Listing 30:

```
359 +metadata.collateralName = string.concat(metadata.collateralName
    ↪ , " in ", baseSymbol, " Silo");
360 +metadata.collateralSymbol = string.concat(metadata.
    ↪ collateralSymbol, "-", baseSymbol);

362 +metadata.protectedName = string.concat(metadata.protectedName,
    ↪ " in ", baseSymbol, " Silo");
    +metadata.protectedSymbol = string.concat(metadata.
    ↪ protectedSymbol, "-", baseSymbol);

365 +metadata.debtName = string.concat(metadata.debtName, " in ",
    ↪ baseSymbol, " Silo");
    +metadata.debtSymbol = string.concat(metadata.debtSymbol, "-",
    ↪ baseSymbol);
```

3.31 CVF-31

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** BaseSilo.sol

Description It would be safer not to change state after an external contract call.
Recommendation Consider moving the external call to the very end of the function, as the function logic doesn't depend on it.
Client Comment The team disagrees because in this case, state changes are beneficial to the msg.sender. External call is made when msg.sender still has a debt, as far as the system is concerned.

Listing 31:

```
607 +// change debt state after token transfer (having debt is bad
    ↪ while trying to re-enter)
    +_state.debtToken.burn(_borrower, repaidShare);
    +_state.totalBorrowAmount -= repaidAmount;
```

3.32 CVF-32

- **Severity** Moderate
- **Status** Info
- **Category** Flaw
- **Source** BaseSilo.sol

Description This not only allows calling "_repay" recursively under "_userLiquidation' but also allows calling '_userLiquidation" under any "nonReentrant" function which could be very dangerous.
Recommendation Consider not allowing calling "_userLiquidation" under "nonReentrant" functions.
Client Comment Acknowledged.

Listing 32:

```
627 +// we can not use `nonReentrant` because we are using it in `
    ↪ _repay`,
+// and `_repay` needs to be reentered as part of a liquidation
+liquidationNonReentrant
```

3.33 CVF-33

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** BaseSilo.sol

Description This return statement assigns zero to the "harvestedFees" return value, which is already guaranteed to be zero here.
Recommendation Consider refactoring like this: if (harvestedFees != 0) { // Rest of the function here }
Client Comment Acknowledged.

Listing 33:

```
839 +return 0;
```

3.34 CVF-34

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** BaseSilo.sol

Description A parameter "_amount" and a returned value "amount" have very similar names. This makes the code harder to read.
Recommendation Consider renaming "_amount" to something like "desiredAmount" and "amount" to something like "actualAmount".
Client Comment Acknowledged.

Listing 34:

```
939 +function _calculateDebtAmountAndShare(AssetStorage storage
    ↪ _state, address _borrower, uint256 _amount)

942         returns (uint256 amount, uint256 repayShare)
```

3.35 CVF-35

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** TwoStepOwnable.sol

Recommendation Consider specifying as "^0.7.0 || ^0.8.0". Also relevant for the next files: PriceProvider.sol, ISwapper.sol, IPriceProvidersRepository.sol, IPriceProvider.sol.
Client Comment Acknowledged.

Listing 35:

```
3 +pragma solidity >=0.7.6 <0.9.0;
```

3.36 CVF-36

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** TwoStepOwnable.sol

Description This version requirement is inconsistent with version requirements in other files.
Recommendation Consider using consistent version requirements.
Client Comment Acknowledged.

Listing 36:

```
3 +pragma solidity >=0.7.6 <0.9.0;
```

3.37 CVF-37

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** TwoStepOwnable.sol

Recommendation This commented errors should be either uncommented and used or removed.
Client Comment Acknowledged.

Listing 37:

```
34 +/**
+ *  error OnlyOwner();
+ *  error OnlyPendingOwner();
+ *  error OwnerIsZero();
+ */
```

3.38 CVF-38

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** TwoStepOwnable.sol

Description String error messages are inefficient.
Recommendation Consider using named errors instead.
Client Comment Acknowledged.

Listing 38:

```
45 +if (owner() != msg.sender) revert("OnlyOwner");

73 +if (newOwner == address(0)) revert("OwnerIsZero");

103 +if (msg.sender != pendingOwner()) revert("OnlyPendingOwner");

127 +if (_owner == newOwner) revert("OwnerDidNotChange");

142 +if (_pendingOwner == newPendingOwner) revert("
    ↪ PendingOwnerDidNotChange");
```

3.39 CVF-39

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** TwoStepOwnable.sol

Description This function is redundant, as “transferPendingOwnership(address(0))” basically does the same.
Client Comment Acknowledged.

Listing 39:

```
93 +function removePendingOwnership() public virtual onlyOwner {
```

3.40 CVF-40

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ShareToken.sol

Recommendation The type of the “_asset” asset should be “IERC20”.
Client Comment IERC20 does not have "decimals" in its interface.

Listing 40:

```
49 +_decimals = IERC20Metadata(_asset).decimals();
```

3.41 CVF-41

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ShareToken.sol

Description The “decimals” property is optional.
Recommendation Consider not relying on its existence in all tokens.
Client Comment Acknowledged.

Listing 41:

```
49 +_decimals = IERC20Metadata(_asset).decimals();
```

3.42 CVF-42

- **Severity** Major
- **Status** Info
- **Category** Unclear behavior
- **Source** ShareToken.sol

Description The "!= 0" part actually allows burning all the tokens. Thus the total supply may drop below "MINIMUM_SHARE_AMOUNT" after the first mint. Is this intentional?
Client Comment Yes, 0 is allowed. The goal is to avoid small amounts only.

Listing 42:

```
83 +if (total != 0 && total < MINIMUM_SHARE_AMOUNT) revert
     ↪ MinimumShareRequirement();
```

3.43 CVF-43

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ShareDebtToken.sol

Description The "_recipient" is actually the message sender, so this check is very unlikely to every succeed.
Recommendation Consider removing this check.
Client Comment Acknowledged.

Listing 43:

```
83 +if (_recipient == address(0)) revert RecipientIsZero();
```

3.44 CVF-44

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LiquidationReentrancyGuard.sol

Description This contract loos like a generic reentrancy guard that is not specific to liquidation logic.
Recommendation Consider removing the "liquidation" word from its code.
Client Comment Acknowledged.

Listing 44:

```
4 +abstract contract LiquidationReentrancyGuard {
```

3.45 CVF-45

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** LiquidationReentrancyGuard.sol

Recommendation Consider changing to "_liquidationStatus != LIQUIDATION_NOT_ENTERED" for strictness.
Client Comment Acknowledged.

Listing 45:

```
13 +if (_liquidationStatus == _LIQUIDATION_ENTERED) {
```

3.46 CVF-46

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** InterestRateDataResolver.sol

Recommendation It would be more efficient to pass a single array of structs with two fields, rather than two parallel arrays. This would also make the length check unnecessary.
Client Comment Acknowledged.

Listing 46:

```
27 +function getDataBatch(ISilo[] calldata _silos, address[]
   ↪ calldata _assets)
```

3.47 CVF-47

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** InterestRateDataResolver.sol

Recommendation It would be more efficient to return a single array of structs with four fields, rather than four parallel arrays.
Client Comment Acknowledged.

Listing 47:

```
31 +IInterestRateModel.Config[] memory modelConfigs,
+uint256[] memory currentInterestRates,
+uint256[] memory siloUtilizations,
+uint256[] memory totalDepositsWithInterest
```

3.48 CVF-48

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** GuardedLaunch.sol

Recommendation This value could be rendered as “250e18".
Client Comment Acknowledged.

Listing 48:

```
17 +uint256 private constant _INITIAL_MAX_LIQUIDITY = 250 * 1e18;
```

3.49 CVF-49

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** TokenHelper.sol

Recommendation The argument type should be “IERC20”.
Client Comment Acknowledged.

Listing 49:

```
13 +function assertAndGetDecimals(address _token) internal view
    ↪ returns (uint256) {

27 +function symbol(address _token) internal view returns (string
    ↪ memory assetSymbol) {
```

3.50 CVF-50

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** TokenHelper.sol

Recommendation The return type should be “uint8”.
Client Comment Acknowledged.

Listing 50:

```
13 +function assertAndGetDecimals(address _token) internal view
    ↪ returns (uint256) {
```

3.51 CVF-51

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** TokenHelper.sol

Description In ERC20, the "decimals" property is used by UI to render token amounts in a human readable way. Using this property in smart contracts is discouraged.
Recommendation Consider treating all token amounts as integers.
Client Comment Acknowledged.

Listing 51:

```
14 +(bool hasMetadata, bytes memory data) = _tokenMetadataCall(
     ↪ _token, abi.encodeCall(IERC20Metadata.decimals,()));
```

3.52 CVF-52

- **Severity** Major
- **Status** Info
- **Category** Suboptimal
- **Source** TokenHelper.sol

Description This function tries to remove zeros not only from the beginning and the end of the provided array, but also from the middle, which is inefficient.
Recommendation Consider reimplementing to not remove zeros from the middle. This would allow using binary search instead of linear search.
Client Comment Acknowledged.

Listing 52:

```
42 +function removeZeros(bytes memory _data) internal pure returns
     ↪ (bytes memory result) {
```

3.53 CVF-53

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** TokenHelper.sol

Description The function is used only for arrays of 32 bytes.
Recommendation Consider taking this into account.
Client Comment Acknowledged.

Listing 53:

```
43 +uint256 n = _data.length;
```

3.54 CVF-54

- **Severity** Major
- **Status** Info
- **Category** Suboptimal
- **Source** TokenHelper.sol

Description This allocated a new array every time, and copies the contents of the existing array into it. This is inefficient.
Recommendation Consider reusing the array.
Client Comment Acknowledged.

Listing 54:

```
49 +result = abi.encodePacked(result, _data[i]);
```

3.55 CVF-55

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** TokenHelper.sol

Recommendation The type of the "_token" argument should be "IERC20".
Client Comment Acknowledged.

Listing 55:

```
55 +function _tokenMetadataCall(address _token, bytes memory _data)
    ↪  private view returns(bool, bytes memory) {
```

3.56 CVF-56

- **Severity** Minor
- **Status** Info
- **Category** Flaw
- **Source** TokenHelper.sol

Description Here, the revert message is ignored.
Recommendation Consider returning it to the caller.
Client Comment Acknowledged.

Listing 56:

```
63 +return (false, "");
```

3.57 CVF-57

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** UniswapV3PriceProvider.sol

Recommendation This value should be a named constant.
Client Comment Acknowledged.

Listing 57:

```
76 +uint24 defaultFee = 500;
```

3.58 CVF-58

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** UniswapV3PriceProvider.sol

Description The division here rounds down.
Recommendation Consider rounding up to ensure that the desired period is fully covered.
Client Comment Acknowledged.

Listing 58:

```
355 +uint256 requiredCardinality = _periodForAvgPrice / _blockTime;
```

3.59 CVF-59

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** BalancerV2PriceProvider.sol

Description Here an arbitrary function is called on a pool to check whether it is valid.
Recommendation Consider calling a dedicated function or implementing EIP-165.
Client Comment Acknowledged.

Listing 59:

```
283 +abi.encodePacked(IPriceOracle.getLargestSafeQueryWindow.
    ↪ selector)
```

3.60 CVF-60

- **Severity** Minor
- **Status** Info
- **Category** Overflow/Underflow
- **Source** Solvency.sol

Description Phantom overflow is possible here.
Recommendation Consider using the "muldiv" function.
Client Comment Acknowledged.

Listing 60:

```
307 +return _assetTotalDeposits + _assetTotalDeposits * _rcomp /
    ↪ _PRECISION_DECIMALS * depositorsShare /
+     _PRECISION_DECIMALS;
```

3.61 CVF-61

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** Ping.sol

Description This version requirements are inconsistent with other files.
Recommendation Consider using consistent version requirements. Also, consider specifying like this: "^0.7.0 || ^0.8.0".
Client Comment Acknowledged.

Listing 61:

```
3 +pragma solidity >=0.7.6 <0.9.0;
```

3.62 CVF-62

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** Ping.sol

Description The zero address check is redundant, as a zero address will not pass the selector test anyway.
Client Comment Acknowledged.

Listing 62:

```
23 +return pingFunction.address != address(0) && pingFunction.
    ↪ selector == pingFunction();
```

3.63 CVF-63

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** EasyMath.sol

Description The "_totalBorrowAmount == 0" check is redundant, as the main expression will anyway return zero in such a case.
Client Comment Acknowledged.

Listing 63:

```
46 +if (_totalDeposits == 0 || _totalBorrowAmount == 0) return 0;
```

3.64 CVF-64

- **Severity** Minor
- **Status** Info
- **Category** Overflow/Underflow
- **Source** EasyMath.sol

Description Phantom overflow is possible here.
Recommendation Consider using the "muldiv" function.
Client Comment Acknowledged.

Listing 64:

```
48 +return _totalBorrowAmount * _dp / _totalDeposits;
```

3.65 CVF-65

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** PriceProvider.sol

Description This version requirement is inconsistent with version requirements in other files.
Recommendation Consider using consistent version requirements.
Client Comment Acknowledged.

Listing 65:

```
3 +pragma solidity >=0.7.6 <0.9.0;
```

3.66 CVF-66

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** IERC20Like.sol

Description This version requirement is inconsistent with version requirements in other files.
Recommendation Consider using consistent version requirements.
Client Comment Acknowledged.

Listing 66:

```
3 +pragma solidity 0.7.6;
```

3.67 CVF-67

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** ITokensFactory.sol

Recommendation The argument type should be "ISiloRepository".
Client Comment Acknowledged.

Listing 67:

```
18 +function initRepository(address _siloRepository) external;
```

3.68 CVF-68

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** ISiloFactory.sol

Recommendation The argument type should be "ISiloRepository".
Client Comment Acknowledged.

Listing 68:

```
16 +function initRepository(address _siloRepository) external;
```

3.69 CVF-69

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** ISiloRepository.sol

Recommendation The return type should be "ISilo".
Client Comment Acknowledged.

Listing 69:

```
175 +function newSilo(address _siloAsset, bytes memory _siloData)
    ↪ external returns (address createdSilo);

190 +) external returns (address createdSilo);
```

3.70 CVF-70

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** IBaseSilo.sol

Description Strings are inefficient. A short string (shorter than 32 bytes) could be very efficiently packed into a word and then unpacked back.
Recommendation Consider using packing for short strings. See the following gist for code example: https://gist.github.com/3sGgpQ8H/567354534170905e047b299286697e19
Client Comment Acknowledged.

Listing 70:

```
55 +string collateralName;

57 +string collateralSymbol;

59 +string protectedName;

61 +string protectedSymbol;

63 +string debtName;

65 +string debtSymbol;
```

3.71 CVF-71

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** IBaseSilo.sol

Recommendation This function should be declared as "view" or even as "pure".
Client Comment Acknowledged.

Listing 71:

```
115 +function VERSION() external returns (uint128); // solhint-
    ↪ disable-line func-name-mixedcase
```

3.72 CVF-72

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** IBaseSilo.sol

Description The "uint128" type is not more efficient than the "uint256" type.
Recommendation Consider using "uint256" for versions.
Client Comment Acknowledged.

Listing 72:

```
115 +function VERSION() external returns (uint128); // solhint-
    ↪ disable-line func-name-mixedcase
```