Report

v. 1.0

Customer

Nexa Finance

Smart Contract Audit

Token Portfolio

16th July 2025

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	16.07.25	A. Zveryanskaya	Initial Draft
0.2	16.07.25	A. Zveryanskaya	Minor revision
1.0	16.07.25	A. Zveryanskaya	Release

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting (Mikhail Vladimirov and Dmitry Khovratovich) 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.

Nexa Finance was founded with the mission to provide world-class mortgage broker service throughout your loan journey with our expertise and passion in ensuring a smooth start for you while embarking on your new life chapter.

3 Project scope

We were asked to review:

- Original Code
- Code with Fixes

Files:

/			
	ERC20Base.sol	ERC20Portfolio.sol	ERC20Token.sol
	NexaCoin.sol	NexaManager.sol	PortfolioManager.sol
	TokenManager.sol		
interfaces/			
	INexaManager.sol		
libraries/			
	Errors.sol		
utils/			
	AccessManager.sol		

4 Methodology

The methodology is not a strict formal procedure, but rather a selection of methods and tactics combined differently and tuned for each particular project, depending on the project structure and technologies used, as well as on client expectations from the audit.

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows best code 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 as well as 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 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 if code actually does what it is supposed to do, if that algorithms are optimal and correct, and if proper data types are used. We also make sure 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.

We classify issues by the following severity levels:

- **Critical issue** directly affects the smart contract functionality and may cause a significant loss.
- **Major issue** is either a solid performance problem or a sign of misuse: a slight code modification or environment change may lead to loss of funds or data. Sometimes it is an abuse of unclear code behaviour which should be double checked.
- **Moderate issue** is not an immediate problem, but rather suboptimal performance in edge cases, an obviously bad code practice, or a situation where the code is correct only in certain business flows.
- **Recommendations** contain code style, best practices and other suggestions.

5 Our findings

We found 12 major, and a few less important issues. All identified Major issues have been fixed.

Fixed 12 out of 12 issues

6 Major Issues

CVF-1 FIXED

- **Category** Flaw
- **Source** ERC20Portfolio.sol

Description This function doesn't check that the tokens are unique, i.e. that no token appears in the array more than once.

Recommendation Implement token uniqueness check. For example, require tokens address to go in ascending order. This could also incorporate zero address check like this: address prevToken = address(0); for (uint256 i = 0; i < _tokens.length; i++) { address token = address(_tokens[i]); require (token > prevToken); prevToken = token; }

Client Comment *Instead of forcing the user to create a portfolio inputing the tokens in ascending order, I prefered to use the EnumerableSet library from OpenZeppelin to check if the token was already in the set.*
import {EnumerableSet} from "@openzeppelin/contracts/utils/structs/EnumerableSet.sol";
for(..) { if (!_tempTokenSet.add(token)) { revert Errors.InvalidToken(); } }

```
65 function _checkTokens(ERC20Token[] memory _tokens) internal pure {
```

CVF-2 FIXED

- **Category** Flaw
- **Source** ERC20Portfolio.sol

Description Event isn't emitted here, due to the "return" statement.

Recommendation Emit an event.

Client Comment *Event emitted before the return statement:*
emit PortfolioUpdated(_token, _quantity, i);

```
121     shares[i] = _quantity;
}
return;
```

CVF-3 FIXED

- **Category** Suboptimal
- **Source** ERC20Portfolio.sol

Description This event is emitted when portfolio wasn't actually updated.

Recommendation Throw an error here signalling that the token is unknown.

Client Comment *revert Errors.TokenNotFound(_token);*

```
126 emit PortfolioUpdated(tokens, shares);
```

CVF-4 FIXED

- **Category** Unclear behavior
- **Source** PortfolioManager.sol

Description Dynamic array doesn't make sense as a gap as it effectively occupies only one static storage slot.

Recommendation Use a static array with constant length, such as: uint256[50] private __gap.

Client Comment *uint256[50] private __gap;*

```
28 uint256[] public __gap;
```

CVF-5 FIXED

- **Category** Unclear behavior
- **Source** TokenManager.sol

Description Dynamic array doesn't make sense as a gap. as it effectively occupies only one static storage slot.

Recommendation Use a static array with constant length, such as: uint256[50] private __gap.

Client Comment *uint256[50] private __gap;*

```
19 uint256[] public __gap;
```

CVF-6 FIXED

- **Category** Unclear behavior
- **Source** NexaCoin.sol

Description Dynamic array doesn't make sense as a gap. as it effectively occupies only one static storage slot.

Recommendation Use a static array with constant length, such as: uint256[50] private __gap.

Client Comment *uint256[50] private __gap;*

```
20 uint256[] public __gap;
```

CVF-7 FIXED

- **Category** Unclear behavior
- **Source** NexaManager.sol

Description Dynamic array doesn't make sense as a gap. as it effectively occupies only one static storage slot.

Recommendation Use a static array with constant length, such as: uint256[50] private __gap.

Client Comment *uint256[50] private __gap;*

```
52 uint256[] public __gap;
```

CVF-8 FIXED

- **Category** Unclear behavior
- **Source** NexaManager.sol

Description Unlike other similar functions, this function doesn't check that "portfolioManager" is set.

Recommendation Add portfolio manager check or explain,. why such a check is not required.

Client Comment *Included the check.*

```
287 function sellBackUnderlyingToken(
```

```
437 function mountPortfolio(
```

CVF-9 FIXED

- **Category** Unclear behavior
- **Source** NexaManager.sol

Description If any of the assets is zero, no assets will be returned due to this check. Also, it is unclear how an asset could be zero here.

Recommendation If there are no valid scenarios where an asset could be zero, just remove this check. Otherwise, skip the null asset and return the others.

Client Comment *Skipped the zero asset and continue.*

```
528 if (addr == address(0)) revert Errors.EmptyAddress();
```

CVF-10 FIXED

- **Category** Flaw
- **Source** NexaManager.sol

Description The condition should be: "_index != length -1". With the current condition, if length is 1, and arbitrary "_index" value will be accepted without error.

Recommendation Fix the condition.

Client Comment *Agree.*

```
654 if (length > 1) {
```

CVF-11 FIXED

- **Category** Flaw
- **Source** NexaManager.sol

Description The returned values are ignored.

Recommendation Explicitly require the returned values to be true.

Client Comment *Return checked.*

```
673 stablecoin.approve(address(this), _stablecoinAmount);
    stablecoin.transferFrom(address(this), _user, _stablecoinAmount);
```

```
683 stablecoin.transferFrom(_user, address(this), _stablecoinAmount);
```

CVF-12 FIXED

- **Category** Flaw
- **Source** NexaManager.sol

Description Due to rounding errors, it could be impossible to unlock all the shared.

Recommendation Store the amount of locked shares in WAD format to prevent rounding errors.

Client Comment *Implemented WAD format.*

```
696 uint256 total = _shares * underlayingShares[i] / SHARE_PRECISION;
```

```
698 portfolioManager.unlockShares(address(token), total);
```

7 Moderate Issues

CVF-13 FIXED

- **Category** Procedural
- **Source** ERC20Portfolio.sol

Description This error gives no clue regarding what particular token is invalid.

Recommendation Include token index into the error.

Client Comment *Included the token.*

```
68 revert Errors.InvalidToken();
```

CVF-14 FIXED

- **Category** Procedural
- **Source** ERC20Portfolio.sol

Description This error gives no clue regarding what particular share is invalid.

Recommendation Include share index into the error.

Client Comment *Included the share.*

```
78 revert Errors.InvalidAmount();
```

CVF-15 FIXED

- **Category** Suboptimal
- **Source** ERC20Portfolio.sol

Description Linear search is inefficient.

Recommendation Pass token index instead of token address.

Client Comment *Passing index.*

```
116 for (uint256 i = 0; i < length; i++) {
        if (address(tokens[i]) == _token) {
```

CVF-16 FIXED

- **Category** Procedural
- **Source** ERC20Portfolio.sol

Description Here all token and all shares are read from the storage just to emit an event.

Recommendation Emit a special token removeal event that includes only the removed token.

Client Comment *emit TokenRemoved(tokens[_index]);*

```
142 emit PortfolioUpdated(tokens, shares);
```

CVF-17 FIXED

- **Category** Flaw
- **Source** PortfolioManager.sol

Recommendation This field shouldn't be public.

Client Comment *uint256[50] private __gap;*

```
28 uint256[] public __gap;
```

CVF-18 FIXED

- **Category** Overflow/Underflow
- **Source** PortfolioManager.sol

Description Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type, while certain intermediary calculation overflows.

Recommendation Use the "mulDiv" function.

Client Comment *Used mulDiv to avoid phantom overflow.*

```
200 uint256 totalShares = _initialSupply * _shares[i] / 1e18;
```

CVF-19 FIXED

- **Category** Procedural
- **Source** PortfolioManager.sol

Description This value is rounded down, i.e. towards the user. A good practice is to always round towards the protocol.

Recommendation Round up here.

Client Comment *Ok. Used Math.Rounding.Ceil*

```
200 uint256 totalShares = _initialSupply * _shares[i] / 1e18;
```

CVF-20 FIXED

- **Category** Flaw
- **Source** TokenManager.sol

Recommendation This field shouldn't be public.

Client Comment *uint256[50] private __gap;*

```
19 uint256[] public __gap;
```

CVF-21 FIXED

- **Category** Flaw
- **Source** NexaCoin.sol

Recommendation This field shouldn't be public.

Client Comment *uint256[50] private __gap;*

```
20 uint256[] public __gap;
```

CVF-22 FIXED

- **Category** Procedural
- **Source** NexaCoin.sol

Recommendation Unchained initializers should be called here.

Client Comment *Ok. Used the unchained initializers.*

```
23 __ERC20_init(_name, _symbol);
   __ERC20Permit_init(_name);
   __ReentrancyGuard_init();
   __Ownable_init(msg.sender);
```

CVF-23 FIXED

- **Category** Flaw
- **Source** NexaManager.sol

Recommendation This field shouldn't be public.

Client Comment *uint256[50] private __gap;*

```
52 uint256[] public __gap;
```

CVF-24 FIXED

- **Category** Procedural
- **Source** NexaManager.sol

Recommendation Unchained initializers should be called here.

Client Comment *Ok. Used the unchained initializers.*

```
76 __AccessControl_init();
   __Pausable_init();
   __ReentrancyGuard_init();
   __Admin_init(_admin);
```

CVF-25 FIXED

- **Category** Procedural
- **Source** NexaManager.sol

Description User shares are accounted twice: once inside the “sharesByUser’ mapping and another time as user’s balance of the asset. These two accountings could easily get out of sync.

Recommendation Account user shares in one place.

Client Comment *Removed sharesByUser, using only the ERC20 token balance.*

```
157 sharesByUser[_user][_asset] += _shares;
```

```
169 ERC20Base(_asset).safeTransfer(_user, _shares);
```

```
204 sharesByUser[_user][_asset] -= _shares;
```

```
215 ERC20Base(_asset).safeTransferFrom(_user, address(this), _shares);
```

```
256 sharesByUser[_user][_asset] -= _shares;
```

```
272 ERC20Base(_asset).burn(_user, _shares);
```

```
409 _portfolio.burn(_user, _shares);
```

```
422 sharesByUser[_user][address(_portfolio)] -= _shares;
```

CVF-26 FIXED

- **Category** Unclear behavior
- **Source** NexaManager.sol

Description This check makes the “_shared” argument redundant, as its values could be derived from the user’s balance.

Recommendation Remove the argument or explain why it is still needed.

Client Comment *_shares removed.*

```
307 if (ERC20Portfolio(_portfolio).balanceOf(_user) != _shares) revert
    ↪ Errors.NotEnoughShares();
```

CVF-27 FIXED

- **Category** Suboptimal
- **Source** NexaManager.sol

Description Transferring from “this” using approve/transferFrom is weird.

Recommendation Use plain transfer or explain why approve/transferFrom is needed here.

Client Comment *Agree. Using plain transfer.*

```
382 stablecoin.safeIncreaseAllowance(address(this), _stablecoinAmount);
    stablecoin.safeTransferFrom(address(this), _destination,
       ↪ _stablecoinAmount);
```

```
673 stablecoin.approve(address(this), _stablecoinAmount);
    stablecoin.transferFrom(address(this), _user, _stablecoinAmount);
```

CVF-28 FIXED

- **Category** Procedural
- **Source** NexaManager.sol

Description This value is rounded down, i.e. towards the user. A good practice is to always round towards the protocol.

Recommendation Round up here.

Client Comment *Ok. Used mulDiv and Math.Rounding.Ceil*

```
460 uint256 total = _shares * underlayingShares[i] / SHARE_PRECISION;
```

CVF-29 FIXED

- **Category** Documentation
- **Source** NexaManager.sol

Description This comment is wrong. The function actually returns assets of the user provided as an argument, rather then the assets of the caller.

Recommendation Fix the comment.

Client Comment *Fixed.*

```
520 /// @notice Gets the assets of the caller
```

CVF-30 FIXED

- **Category** Procedural
- **Source** NexaManager.sol

Recommendation The events should be emitted by the token contracts, not by the manager.

Client Comment *Event removed from manager.*

```
580 ERC20Base(_token).setDueDate(_dueDate);
    emit DueDateSet(_token, _dueDate);

598 ERC20Base(_token).setStrategy(_strategy);
    emit StrategySet(_token, _strategy);

606 ERC20Base(_token).addClass(_class);
    emit ClassAdded(_token, _class);

614 ERC20Base(_token).removeClass(_class);
    emit ClassRemoved(_token, _class);

623 ERC20Base(_token).setMinExposure(_strategy, _minExposure);
    emit MinExposureSet(_token, _strategy, _minExposure);

632 ERC20Base(_token).setMaxExposure(_strategy, _maxExposure);
    emit MaxExposureSet(_token, _strategy, _maxExposure);
```

CVF-31 FIXED

- **Category** Procedural
- **Source** NexaManager.sol

Recommendation The event should be emitted by the asset contract, not by the manager.

Client Comment *Event removed from manager.*

```
590 ERC20Base(_asset).setMaxStablecoinAmount(_stablecoin,
       ↪ _maxStablecoinAmount);
    emit MaxStablecoinAmountSet(_asset, _stablecoin,
       ↪ _maxStablecoinAmount);
```

8 Recommendations

CVF-32 FIXED

- **Category** Procedural
- **Source** AccessManager.sol

Description Consider specifying as "^0.8.0" unless there is something special regarding this particular version.

Recommendation Also relevant for: INexaManager.sol, ERC20Base.sol, ERC20Token.sol, ERC20Portfolio.sol, PortfolioManager.sol, TokenManager.sol, NexaCoin.sol, NexaManager.sol.

Client Comment *Changed to ^0.8.0.*

```
2 pragma solidity ^0.8.21;
```

CVF-33 FIXED

- **Category** Unclear behavior
- **Source** AccessManager.sol

Recommendation These functions should emit some events.

Client Comment *Added events.*

```
42 function addToWhitelist(address _account) public onlyRole(
   ↪ ACCESS_CONTROLLER_ROLE) {

48 function removeFromWhitelist(address _account) public onlyRole(
   ↪ ACCESS_CONTROLLER_ROLE) {

61 function addToBlacklist(address _account) public onlyRole(
   ↪ ACCESS_CONTROLLER_ROLE) {

67 function removeFromBlacklist(address _account) public onlyRole(
   ↪ ACCESS_CONTROLLER_ROLE) {
```

CVF-34 FIXED

- **Category** Procedural
- **Source** Errors.sol

Description This version requirement is inconsistent with other files in the same code base.

Recommendation Use consistent version requirements across code base.

Client Comment *All contracts are ^0.8.0 now.*

```
2 pragma solidity ^0.8.0;
```

CVF-35 FIXED

- **Category** Procedural
- **Source** Errors.sol

Description This library only contains errors.

Recommendation Consider moving errors into to top level and removing this library.

Client Comment *Ok, moved to the root folder.*

```
4 library Errors {
```

CVF-36 FIXED

- **Category** Procedural
- **Source** Errors.sol

Recommendation These errors could be made more useful by adding certain parameters into them.

Client Comment *Improved error messages with some parameters.*

```
7  error AssetNotFound();
   error Blacklisted();
```

```
11 error InvalidAmount();
   error InvalidArrayLength();
   error InvalidDueDate();
   error InvalidInitialPrice();
   error InvalidInitialSupply();
   error InvalidLength();
   error InvalidMaxExposure();
   error InvalidMinExposure();
   error InvalidToken();
20 error MaxStablecoinAmountReached();
   error NotAPortfolio();
   error NotEnoughBalance();
   error NotEnoughShares();
   error NotEnoughTokens();
   error NotManager();
   error NotWhitelisted();
   error NotController();
   error NotAdmin();
```

CVF-37 FIXED

- **Category** Documentation
- **Source** Errors.sol

Description It is unclear what is an empty address.

Recommendation Rename or clarify in a documentation comment.

Client Comment *Renamed to ZeroAddressNotAllowed().*

```
9  error EmptyAddress();
```

CVF-38 FIXED

- **Category** Bad datatype
- **Source** INexaManager.sol

Recommendation The type for this field should be more specific.

Client Comment *Field removed.*

```
14 address tokenAddress;
```

CVF-39 FIXED

- **Category** Suboptimal
- **Source** INexaManager.sol

Recommendation It would be more efficient to have a single array of structs with two fields, rather than two parallel arrays.

Client Comment *Created 2 new structs to group the fields.*

```
32 ERC20Token[] tokens;
```

```
34 uint256[] shares;
```

```
47 string[] underlyingTokens;
```

```
49 uint256[] underlyingAmounts;
```

CVF-40 FIXED

- **Category** Bad datatype
- **Source** INexaManager.sol

Recommendation The type for this field should be more specific.

Client Comment *Just improved the comment to say that it can be a token or portfolio address. It's not specific to be flexible.*

```
56 address asset;
```

CVF-41 FIXED

- **Category** Bad naming
- **Source** INexaManager.sol

Recommendation Events are usually named via nouns, such as “Token” or “Portfolio”.

Client Comment *Improved the name of the events to be nouns focused.*

```
59 event TokenCreated(address indexed token, string name, string symbol
   ↪ , uint256 initialSupply, uint256 initialPrice, uint256 _days);
60 event PortfolioCreated(address indexed portfolio, string name,
   ↪ string symbol, uint256 initialSupply, uint256 initialPrice,
   ↪ uint256 _days);
   event Minted(address indexed asset, uint256 shares);
   event Burned(address indexed asset, uint256 shares);
   event Bought(address indexed user, address indexed asset, address
   ↪ indexed stablecoin, uint256 stablecoinAmount, uint256 shares);
   event Sold(address indexed user, address indexed asset, address
   ↪ indexed stablecoin, uint256 stablecoinAmount, uint256 shares);
   event SoldBack(address indexed user, address indexed asset, address
   ↪ indexed stablecoin, uint256 stablecoinAmount, uint256 shares);
   event Withdrawn(address indexed user, address indexed stablecoin,
   ↪ uint256 stablecoinAmount);
```

```
70 event PortfolioUpdated(address indexed portfolio, ERC20Token[]
   ↪ tokens, uint256[] shares, uint256 adjustedPrice);
   event NexaManagerInitialized(address indexed nexa, address indexed
   ↪ admin);
   event PortfolioManagerSet(address indexed portfolioManager);
   event DueDateSet(address indexed token, uint256 dueDate);
   event MaxStablecoinAmountSet(address indexed asset, address
   ↪ stablecoin, uint256 maxStablecoinAmount);
   event StrategySet(address indexed token, ERC20Token.Strategy
   ↪ strategy);
   event ClassAdded(address indexed token, string class);
   event ClassRemoved(address indexed token, string class);
   event MinExposureSet(address indexed token, ERC20Token.Strategy
   ↪ strategy, uint256 minExposure);
   event MaxExposureSet(address indexed token, ERC20Token.Strategy
   ↪ strategy, uint256 maxExposure);
```

CVF-42 FIXED

- **Category** Bad datatype
- **Source** INexaManager.sol

Recommendation The type for the “token” parameters should be more specific.

Client Comment *Instead of changing the type I just changed the name to tokenAddress because that’s what it mean, the address of the ER20 token.*

```
59 event TokenCreated(address indexed token, string name, string symbol
      ↪ , uint256 initialSupply, uint256 initialPrice, uint256 _days);
```

```
73 event DueDateSet(address indexed token, uint256 dueDate);
```

```
75 event StrategySet(address indexed token, ERC20Token.Strategy
      ↪ strategy);
   event ClassAdded(address indexed token, string class);
   event ClassRemoved(address indexed token, string class);
   event MinExposureSet(address indexed token, ERC20Token.Strategy
      ↪ strategy, uint256 minExposure);
   event MaxExposureSet(address indexed token, ERC20Token.Strategy
      ↪ strategy, uint256 maxExposure);
```

CVF-43 FIXED

- **Category** Bad datatype
- **Source** INexaManager.sol

Recommendation The type for the “portfolio” parameters should be more specific.

Client Comment *Instead of changing the type I just changed the name to portfolioAddress because that’s what it mean, the address of the ER20 portfolio.*

```
60 event PortfolioCreated(address indexed portfolio, string name,
   ↪ string symbol, uint256 initialSupply, uint256 initialPrice,
   ↪ uint256 _days);
```

```
67 event UnmountedPortfolio(address indexed user, address indexed
   ↪ portfolio, uint256 shares);
   event MountedPortfolio(address indexed user, address indexed
   ↪ portfolio, uint256 shares);
   event SoldBackUnderlyingToken(address indexed user, address indexed
   ↪ portfolio, address token, address stablecoin, uint256
   ↪ stablecoinAmount, uint256 shares);
70 event PortfolioUpdated(address indexed portfolio, ERC20Token[]
   ↪ tokens, uint256[] shares, uint256 adjustedPrice);
```

CVF-44 INFO

- **Category** Bad datatype
- **Source** INexaManager.sol

Recommendation The type for the “asset” parameter should be more specific.

Client Comment *I’ll leave it as it is because I want it to be flexible, so that the function can handle any ERC20 address.*

```
61 event Minted(address indexed asset, uint256 shares);
   event Burned(address indexed asset, uint256 shares);
   event Bought(address indexed user, address indexed asset, address
   ↪ indexed stablecoin, uint256 stablecoinAmount, uint256 shares);
   event Sold(address indexed user, address indexed asset, address
   ↪ indexed stablecoin, uint256 stablecoinAmount, uint256 shares);
   event SoldBack(address indexed user, address indexed asset, address
   ↪ indexed stablecoin, uint256 stablecoinAmount, uint256 shares);
```

```
74 event MaxStablecoinAmountSet(address indexed asset, address
   ↪ stablecoin, uint256 maxStablecoinAmount);
```

CVF-45 FIXED

- **Category** Bad datatype
- **Source** INexaManager.sol

Recommendation The type for the “stablecoin” parameters should be more specific.

Client Comment *Instead of changing the type I just changed the name to stablecointAddress because that’s what it mean, the address of the ER20 stablecoin.*

```
63 event Bought(address indexed user, address indexed asset, address
      ↪ indexed stablecoin, uint256 stablecoinAmount, uint256 shares);
   event Sold(address indexed user, address indexed asset, address
      ↪ indexed stablecoin, uint256 stablecoinAmount, uint256 shares);
   event SoldBack(address indexed user, address indexed asset, address
      ↪ indexed stablecoin, uint256 stablecoinAmount, uint256 shares);
   event Withdrawn(address indexed user, address indexed stablecoin,
      ↪ uint256 stablecoinAmount);
```

```
69 event SoldBackUnderlyingToken(address indexed user, address indexed
      ↪ portfolio, address token, address stablecoin, uint256
      ↪ stablecoinAmount, uint256 shares);
```

```
74 event MaxStablecoinAmountSet(address indexed asset, address
      ↪ stablecoin, uint256 maxStablecoinAmount);
```

CVF-46 FIXED

- **Category** Bad datatype
- **Source** INexaManager.sol

Recommendation The type for the “nexa” parameter should be more specific.

Client Comment *Changed the field name to nexaAddress.*

```
71 event NexaManagerInitialized(address indexed nexa, address indexed
      ↪ admin);
```

CVF-47 FIXED

- **Category** Bad datatype
- **Source** INexaManager.sol

Recommendation The type for the “portfolioManager” parameter should be more specific.

Client Comment *Changed the field name to portfolioManagerAddress.*

```
72 event PortfolioManagerSet(address indexed portfolioManager);
```

CVF-48 FIXED

- **Category** Documentation
- **Source** ERC20Base.sol

Description The number format of this constant is unclear.

Recommendation Explain in a documentation comment.

Client Comment *Added comments to MAX_EXPOSURE.*

```
13 uint256 public constant MAX_EXPOSURE = 10000;
```

CVF-49 FIXED

- **Category** Bad naming
- **Source** ERC20Base.sol

Recommendation Events are usually named via nouns, such as ”DueDate” or “Adjusted-Price”.

Client Comment *Improved the name of the events to be nouns focused.*

```
28 event DueDateSet(uint256 oldDueDate, uint256 newDueDate);
   event AdjustedPriceSet(uint256 timestamp, uint256 oldAdjustedPrice,
      ↪ uint256 newAdjustedPrice);
30 event MaxStablecoinAmountSet(address stablecoin, uint256 maxAmount);
   event StrategySet(Strategy strategy);
   event ClassAdded(string class);
   event ClassRemoved(string class);
   event MinExposureSet(Strategy strategy, uint256 minExposure);
   event MaxExposureSet(Strategy strategy, uint256 maxExposure);
```

CVF-50 FIXED

- **Category** Procedural
- **Source** ERC20Base.sol

Description The old value parameters are redundant, as they could be derived from the previous events.

Recommendation Remove the old value parameters.

Client Comment *Removed old values.*

```
28 event DueDateSet(uint256 oldDueDate, uint256 newDueDate);
   event AdjustedPriceSet(uint256 timestamp, uint256 oldAdjustedPrice,
      ↪ uint256 newAdjustedPrice);
```

CVF-51 FIXED

- **Category** Procedural
- **Source** ERC20Base.sol

Recommendation The “stablecoin” parameter should be indexed.

Client Comment *Indexed.*

```
30 event MaxStablecoinAmountSet(address stablecoin, uint256 maxAmount);
```

CVF-52 FIXED

- **Category** Procedural
- **Source** ERC20Base.sol

Recommendation The “strategy” parameter should be indexed.

Client Comment *Indexed.*

```
31 event StrategySet(Strategy strategy);
```

```
34 event MinExposureSet(Strategy strategy, uint256 minExposure);
   event MaxExposureSet(Strategy strategy, uint256 maxExposure);
```

CVF-53 FIXED

- **Category** Procedural
- **Source** ERC20Base.sol

Recommendation The “class” parameters should be indexed.

Client Comment *Indexed.*

```
32 event ClassAdded(string class);
   event ClassRemoved(string class);
```

CVF-54 FIXED

- **Category** Documentation
- **Source** ERC20Base.sol

Description The number format of the exposure parameters is unclear.

Recommendation Consider explaining in a documentation comment.

Client Comment *Added comments to exposure.*

```
34 event MinExposureSet(Strategy strategy, uint256 minExposure);
   event MaxExposureSet(Strategy strategy, uint256 maxExposure);
```

CVF-55 FIXED

- **Category** Documentation
- **Source** ERC20Base.sol

Description It is unclear, what terms this prices are denominated it.

Recommendation Explain in a documentation comment.

Client Comment *Added comments.*

```
40 /// @notice The initial price of the token (usually 100.00)
   uint256 public initialPrice;
```

```
43 /// @notice The adjusted price of the token
   uint256 public adjustedPrice;
```

CVF-56 FIXED

- **Category** Documentation
- **Source** ERC20Base.sol

Description The number format of the values for these mappings is unclear.

Recommendation Explain in a documentation comment.

Client Comment *Added comments.*

```
60 mapping(Strategy => uint256) public minExposure;
```

```
64 mapping(Strategy => uint256) public maxExposure;
```

CVF-57 INFO

- **Category** Flaw
- **Source** ERC20Base.sol

Description This function seems very dangerous, as it allows the owner to burn other people's tokens.

Recommendation Remove this function or restrict owner's rights.

Client Comment *Unfortunatelly it's true, it's dangerous and we need to keep it while our platform does not support auto custody.*

```
112 /// @notice Burns tokens from an address
    /// @param _from The address to burn tokens from
    /// @param _shares The number of shares to burn
    function burn(address _from, uint256 _shares) external onlyOwner {
```

CVF-58 FIXED

- **Category** Unclear behavior
- **Source** ERC20Base.sol

Recommendation These events are emitted even if nothing actually changed.

Client Comment *Only emit events if the value changed.*

```
124 emit DueDateSet(oldDueDate, dueDate);
```

```
132 emit AdjustedPriceSet(block.timestamp, oldAdjustedPrice,
    ↪ adjustedPrice);
```

```
141 emit MaxStablecoinAmountSet(_stablecoin, _maxStablecoinAmount);
```

```
148 emit StrategySet(_strategy);
```

```
180 emit MinExposureSet(_strategy, _minExposure);
```

```
191 emit MaxExposureSet(_strategy, _maxExposure);
```

CVF-59 FIXED

- **Category** Documentation
- **Source** ERC20Base.sol

Description The number format for the exposure arguments is unclear.

Recommendation Explain in a documentation comment.

Client Comment *Added comments to exposure.*

```
175 function setMinExposure(Strategy _strategy, uint256 _minExposure)
    ↪ external onlyOwner {
```

```
186 function setMaxExposure(Strategy _strategy, uint256 _maxExposure)
    ↪ external onlyOwner {
```

CVF-60 FIXED

- **Category** Procedural
- **Source** ERC20Token.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

Client Comment *Added comment to explain why the constructor body is empty.*

```
32 ) {}
```

CVF-61 FIXED

- **Category** Bad naming
- **Source** ERC20Portfolio.sol

Recommendation Events are usually named via nouns, such as “PortfolioUpdate” and “TokenRemoval”.

Client Comment *Improved the name of the events to be nouns focused.*

```
14 event PortfolioUpdated(ERC20Token[] tokens, uint256[] shares);
   event TokenRemoved(ERC20Token token);
```

CVF-62 FIXED

- **Category** Procedural
- **Source** ERC20Portfolio.sol

Recommendation It would be more efficient to use a single array of structs with two fields, instead of two parallel arrays.

Client Comment *Used a single array of struct instead of 2 arrays.*

```
14 event PortfolioUpdated(ERC20Token[] tokens, uint256[] shares);
```

CVF-63 FIXED

- **Category** Procedural
- **Source** ERC20Portfolio.sol

Recommendation It would be more efficient to use a single array of structs with two fields instead of two parallel arrays,.

Client Comment *Used a single array of struct instead of 2 arrays.*

```
20 ERC20Token[] public tokens;
```

```
23 uint256[] public shares;
```

CVF-64 INFO

- **Category** Procedural
- **Source** ERC20Portfolio.sol

Recommendation It would be more efficient to pass a single array of structs with two fields instead of two parallel arrays. This would also make the length check unnecessary.

Client Comment *It's easier for the UI to pass both arrays intead of grouping in 1 array of structs.*

```
41     ERC20Token[] memory _tokens,
       uint256[] memory _shares
```

```
98 function updateUnderlyingTokens(ERC20Token[] memory _tokens, uint256
   ↪ [] memory _shares) external onlyOwner {
```

CVF-65 FIXED

- **Category** Documentation
- **Source** ERC20Portfolio.sol

Description It is unclear what exact checks are performed by this function.

Recommendation Elaborate more.

Client Comment *Elaborated more on the _checkTokens comments.*

```
63 /// @notice Checks the tokens in the portfolio
```

CVF-66 INFO

- **Category** Suboptimal
- **Source** ERC20Portfolio.sol

Description This check is redundant, as it is anyway possible to pass a dead token address.

Recommendation Remove this check.

Client Comment *This check at least mitigates that the UI sends an empty token address. We prefer to keep it.*

```
67 if (address(_tokens[i]) == address(0)) {
```

CVF-67 INFO

- **Category** Suboptimal
- **Source** ERC20Portfolio.sol

Description These functions are redundant as “tokens” and “shares” fields are public.

Recommendation Remove these functions, or make the corresponding fields non-public, or explain, why these functions are indeed necessary.

Client Comment *Now we have ”TokensAndShares[] private tokensAndShares” so it’s not public and it doesn’t exist anymore. We need to keep it.*

```
85 function getTokens() external view returns (ERC20Token[] memory) {
```

```
91 function getShares() external view returns (uint256[] memory) {
```

CVF-68 INFO

- **Category** Bad datatype
- **Source** ERC20Portfolio.sol

Recommendation The type for the “_token” argument should be “ERC20Token”.

Client Comment *Now we use the index: ”function updateQuantity(uint256 _tokenIndex, ...” so nothing to change anymore.*

```
114 function updateQuantity(address _token, uint256 _quantity) external
    ↪ onlyOwner {
```

CVF-69 FIXED

- **Category** Bad naming
- **Source** PortfolioManager.sol

Recommendation Events are usually named via nouns, such as “SharesLock” or “SharesUnlock”.

Client Comment *Improved the name of the events to be nouns focused.*

```
13 event SharesUnlocked(address indexed token, uint256 shares);
   event SharesLocked(address indexed token, uint256 shares);
```

CVF-70 FIXED

- **Category** Bad datatype
- **Source** PortfolioManager.sol

Recommendation The type for the “token” parameters should be “ERC20Token”.

Client Comment *Changed type to ERC20Token.*

```
13 event SharesUnlocked(address indexed token, uint256 shares);
   event SharesLocked(address indexed token, uint256 shares);
   event Approval(address indexed token, address indexed spender,
      ↪ uint256 amount);
```

CVF-71 FIXED

- **Category** Bad datatype
- **Source** PortfolioManager.sol

Recommendation The key type for this mapping should be “ERC20Token”.

Client Comment *Changed the mapping type to ERC20Token.*

```
25 mapping(address token => uint256 shares) public lockedShares;
```

CVF-72 INFO

- **Category** Suboptimal
- **Source** PortfolioManager.sol

Description This check is redundant, as it is anyway possible to pass a dead manager address.

Recommendation Remove this check.

Client Comment *With this test at least we mitigate that it's passed as argument the empty address as the manager of the contract.*

```
47 if (_manager == address(0)) revert Errors.EmptyAddress();
```

CVF-73 INFO

- **Category** Suboptimal
- **Source** PortfolioManager.sol

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

Client Comment *Internally at the ERC20Portfolio we are using an array of struct TokensAndShares. But here it's easier to the UI to send different arrays.*

```
70 ERC20Token[] memory _tokens,
   uint256[] memory _shares
```

CVF-74 FIXED

- **Category** Bad datatype
- **Source** PortfolioManager.sol

Recommendation The return type should be "ERC20Portfolio".

Client Comment *Returned ERC20Portfolio.*

```
75 returns (address)
```

CVF-75 FIXED

- **Category** Procedural
- **Source** PortfolioManager.sol

Description The value “portfolios.length” is calculated twice.

Recommendation Calculate once and reuse.

Client Comment *Avoided calculating twice.*

```
102 Asset[] memory assets = new Asset[](portfolios.length);
    uint256 length = portfolios.length;
```

CVF-76 FIXED

- **Category** Bad datatype
- **Source** PortfolioManager.sol

Recommendation The argument type should be “ERC20Portfolio”.

Client Comment *Argument of type ERC20Portfolio.*

```
113 function getPortfolioInfo(address _asset) external view returns (
       ↪ PortfolioInfo memory) {
```

CVF-77 FIXED

- **Category** Bad datatype
- **Source** PortfolioManager.sol

Recommendation The type for the "_token" arguments should be "ERC20Token".

Client Comment *Changed casts.*

```
133 function unlockShares(address _token, uint256 _shares) external
    ↪ onlyManager {
```

```
141 function lockShares(address _token, uint256 _shares) external
    ↪ onlyManager {
```

```
150 function approve(address _token, address _spender, uint256 _amount)
    ↪ external onlyManager {
```

```
160 function allowance(address _token, address _owner, address _spender)
    ↪  external view returns (uint256) {
```

```
168 function balanceOf(address _token, address _owner) external view
    ↪ returns (uint256) {
```

```
175 function getLockedShares(address _token) external view returns (
    ↪ uint256) {
```

CVF-78 FIXED

- **Category** Bad datatype
- **Source** PortfolioManager.sol

Recommendation The "_token" argument should be casted to "ERC20Token" rather then to "ERC20Portfolio".

Client Comment *Changed casts.*

```
151 ERC20Portfolio(_token).approve(_spender, _amount);
```

```
161 return ERC20Portfolio(_token).allowance(_owner, _spender);
```

```
169 return ERC20Portfolio(_token).balanceOf(_owner);
```

CVF-79 INFO

- **Category** Suboptimal
- **Source** PortfolioManager.sol

Recommendation It would be more efficient to pass a single array of structs with two fields, rather than two parallel arrays.

Client Comment *Internally at the ERC20Portfolio we are using an array of struct TokensAndShares. But here it's easier to the UI to send different arrays.*

```
185 ERC20Token[] memory _tokens,
    uint256[] memory _shares,
```

CVF-80 INFO

- **Category** Procedural
- **Source** TokenManager.sol

Description These fields are redundant, as it is anyway possible to pass a dead admin or manager address.

Recommendation Remove these checks.

Client Comment *With this test at least we mitigate that it's passed as argument the empty address as the admin and manager of the contract.*

```
31 if (_admin == address(0)) revert Errors.EmptyAddress();
   if (_manager == address(0)) revert Errors.EmptyAddress();
```

CVF-81 FIXED

- **Category** Bad datatype
- **Source** TokenManager.sol

Recommendation The return type should be "ERC20Token".

Client Comment *Returned ERC20Token.*

```
54 returns (address)
```

CVF-82 FIXED

- **Category** Bad datatype
- **Source** TokenManager.sol

Recommendation The type for the "_token" arguments should be "ERC20Token".

Client Comment *Changed casts.*

```
76 function approve(address _token, address _spender, uint256 _shares)
   ↪ external onlyRole(DEFAULT_ADMIN_ROLE) {
```

```
85 function allowance(address _token, address _owner, address _spender)
   ↪  external view returns (uint256) {
```

```
93 function balanceOf(address _token, address _owner) external view
   ↪ returns (uint256) {
```

CVF-83 FIXED

- **Category** Bad datatype
- **Source** TokenManager.sol

Recommendation The argument type should be "ERC20Token".

Client Comment *Changed to ERC20Token.*

```
111 function getTokenInfo(address _asset) external view returns (
    ↪ TokenInfo memory) {
```

CVF-84 FIXED

- **Category** Documentation
- **Source** NexaManager.sol

Description It is unclear how this address is used to withdraw funds.

Recommendation Elaborate more.

Client Comment *Improved the comments.*

```
39 /// @notice The Nexa address (used to withdraw funds)
```

CVF-85 FIXED

- **Category** Suboptimal
- **Source** NexaManager.sol

Recommendation It would be more efficient otherwise merge these two mappings into a single mapping whose keys are users and values are structs encapsulating the values of the original mappings.

Client Comment *Removed sharesByUser, using only the ERC20 token balance (item #26).*

```
46 mapping(address user => address[] assets) public assetsByUser;
```

```
49 mapping(address user => mapping(address asset => uint256 shares))
   ↪ public sharesByUser;
```

CVF-86 INFO

- **Category** Procedural
- **Source** NexaManager.sol

Description This check is redundant as it is anyway possible to pass a dead Nexa address.

Recommendation Remove this check.

Client Comment *With this test at least we mitigate that it's passed as argument the empty address as the nexa address of the contract.*

```
74 if (_nexa == address(0)) revert Errors.EmptyAddress();
```

CVF-87 INFO

- **Category** Procedural
- **Source** NexaManager.sol

Recommendation Consider merging these two events into one event.

Client Comment *No, Initialized is from OpenZeppelin Initializable contract.*

```
85 emit Initialized(1); // version 1
   emit NexaManagerInitialized(_nexa, _admin);
```

CVF-88 FIXED

- **Category** Bad datatype
- **Source** NexaManager.sol

Recommendation The type for the "_asset" argument should be "ERC20Base" or some interface derived from it.

Client Comment *Changed casts.*

```
94  function mint(address _asset, uint256 _shares)

112 function burn(address _asset, uint256 _shares)

133 function buy(address _user, address _asset, address _stablecoin,
    ↪ uint256 _stablecoinAmount, uint256 _shares)

180 function sell(address _user, address _asset, address _stablecoin,
    ↪ uint256 _stablecoinAmount, uint256 _shares)

228     address _asset,

500 function getAssetByUser(address _user, address _asset) public view
    ↪ returns (bool, uint256) {

546 function getBalanceForAsset(address _asset) external view returns (
    ↪ uint256) {

562 function getMaxStablecoinAmount(address _asset, address _stablecoin)
    ↪  public view returns (uint256) {

589 function setMaxStablecoinAmount(address _asset, address _stablecoin,
    ↪  uint256 _maxStablecoinAmount) external onlyController {

689 function _processUnlocking(address _asset, uint256 _shares) internal
    ↪  {
```

CVF-89 INFO

- **Category** Procedural
- **Source** NexaManager.sol

Description This check is redundant, as it is anyway possible to pass a dead asset address.

Recommendation Remove this check.

Client Comment *This check at least mitigates that the UI sends an empty token address. We prefer to keep it.*

```
100 if (_asset == address(0)) revert Errors.EmptyAddress();
```

```
118 if (_asset == address(0)) revert Errors.EmptyAddress();
```

CVF-90 FIXED

- **Category** Procedural
- **Source** NexaManager.sol

Description This check is redundant, as it would anyway be performed inside the “burn” call.

Recommendation Remove this check.

Client Comment *Removed.*

```
120 if (ERC20Base(_asset).balanceOf(address(this)) < _shares) revert
    ↪ Errors.NotEnoughShares();
```

CVF-91 FIXED

- **Category** Bad datatype
- **Source** NexaManager.sol

Recommendation The type for the "_stablecoin" arguments should be "IERC20".

Client Comment *Changed casts.*

```
133 function buy(address _user, address _asset, address _stablecoin,
    ↪ uint256 _stablecoinAmount, uint256 _shares)
```

```
180 function sell(address _user, address _asset, address _stablecoin,
    ↪ uint256 _stablecoinAmount, uint256 _shares)
```

```
229     address _stablecoin,
```

```
291     address _stablecoin,
```

```
367 function withdraw(address _destination, address _stablecoin, uint256
    ↪  _stablecoinAmount)
```

```
553 function getBalanceForStablecoin(address _stablecoin) external view
    ↪ returns (uint256) {
```

```
562 function getMaxStablecoinAmount(address _asset, address _stablecoin)
    ↪  public view returns (uint256) {
```

```
589 function setMaxStablecoinAmount(address _asset, address _stablecoin,
    ↪  uint256 _maxStablecoinAmount) external onlyController {
```

```
671 function _transferStablecoinToUser(address _user, address
    ↪ _stablecoin, uint256 _stablecoinAmount) internal {
```

```
681 function _transferStablecoinFromUser(address _user, address
    ↪ _stablecoin, uint256 _stablecoinAmount) internal {
```

CVF-92 INFO

- **Category** Procedural
- **Source** NexaManager.sol

Description These checks are redundant, as it is anyway possible to pass dead addresses.

Recommendation Remove these checks.

Client Comment *At least it checks for empty (zero) addresses. We prefer to keep it.*

```
139 if (_user == address(0)) revert Errors.EmptyAddress();
140 if (_asset == address(0)) revert Errors.EmptyAddress();
    if (_stablecoin == address(0)) revert Errors.EmptyAddress();
```

```
186 if (_user == address(0)) revert Errors.EmptyAddress();
    if (_asset == address(0)) revert Errors.EmptyAddress();
    if (_stablecoin == address(0)) revert Errors.EmptyAddress();
```

```
238 if (_user == address(0)) revert Errors.EmptyAddress();
    if (_asset == address(0)) revert Errors.EmptyAddress();
240 if (_stablecoin == address(0)) revert Errors.EmptyAddress();
```

```
301 if (_user == address(0)) revert Errors.EmptyAddress();
    if (_portfolio == address(0)) revert Errors.EmptyAddress();
    if (_token == address(0)) revert Errors.EmptyAddress();
    if (_stablecoin == address(0)) revert Errors.EmptyAddress();
```

CVF-93 FIXED

- **Category** Procedural
- **Source** NexaManager.sol

Description The error is misleading, as the same error is used for zero addresses provided as argument, while here the problem is that portfolio manager is not set.

Recommendation Use different error here.

Client Comment *Instead of changing the name of the event I pass it as argument to be more explicit: Errors.ZeroAddressNotAllowed("portfolioManager").*

```
144 if (address(portfolioManager) == address(0)) revert Errors.
    ↪ EmptyAddress();
```

```
191 if (address(portfolioManager) == address(0)) revert Errors.
    ↪ EmptyAddress();
```

```
246 if (address(portfolioManager) == address(0)) revert Errors.
    ↪ EmptyAddress();
```

```
402 if (address(portfolioManager) == address(0)) revert Errors.
    ↪ EmptyAddress();
```

CVF-94 FIXED

- **Category** Procedural
- **Source** NexaManager.sol

Recommendation These checks are redundant, as they will anyway be performed when transferring tokens.

Client Comment *Removed.*

```
146 if (ERC20Base(_asset).balanceOf(address(this)) < _shares) revert
    ↪ Errors.NotEnoughShares();
    if (IERC20(_stablecoin).balanceOf(_user) < _stablecoinAmount) revert
    ↪  Errors.NotEnoughBalance();
```

```
192 if (ERC20Base(_asset).balanceOf(_user) < _shares) revert Errors.
    ↪ NotEnoughShares();
    if (IERC20(_stablecoin).balanceOf(address(this)) < _stablecoinAmount
    ↪ ) revert Errors.NotEnoughBalance();
```

```
244 if (ERC20Base(_asset).balanceOf(_user) < _shares) revert Errors.
    ↪ NotEnoughShares();
    if (IERC20(_stablecoin).balanceOf(address(this)) < _stablecoinAmount
    ↪ ) revert Errors.NotEnoughBalance();
```

```
309 if (IERC20(_stablecoin).balanceOf(address(this)) < _stablecoinAmount
    ↪ ) revert Errors.NotEnoughBalance();
```

CVF-95 FIXED

- **Category** Procedural
- **Source** NexaManager.sol

Description Here a value just written into the storage is read back.

Recommendation Reuse the written value.

Client Comment *We don't have sharesByUser anymore (#26).*

```
207 if (sharesByUser[_user][_asset] == 0) {
```

```
259 if (sharesByUser[_user][_asset] == 0) {
```

CVF-96 FIXED

- **Category** Bad datatype
- **Source** NexaManager.sol

Recommendation The type for the "_portfolio" arguments should be "ERC20Portfolio".

Client Comment *Changed.*

```
289     address _portfolio,
```

```
337     address _portfolio,
```

```
392 function unmountPortfolio(address _user, ERC20Portfolio _portfolio,
    ↪ uint256 _shares)
```

```
439     ERC20Portfolio _portfolio,
```

CVF-97 INFO

- **Category** Suboptimal
- **Source** NexaManager.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 *Internally at the ERC20Portfolio we are using an array of struct TokensAndShares. But here it's easier to the UI to send different arrays.*

```
338 ERC20Token[] memory _tokens,
    uint256[] memory _shares,
```

CVF-98 INFO

- **Category** Procedural
- **Source** NexaManager.sol

Description This check is redundant, as it is anyway possible to pass a dead portfolio address.

Recommendation Remove this check.

Client Comment *At least it checks for empty (zero) addresses. We prefer to keep it.*

```
347 if (_portfolio == address(0)) revert Errors.EmptyAddress();
```

CVF-99 INFO

- **Category** Procedural
- **Source** NexaManager.sol

Description These checks are redundant, as it is anyway possible to pass a dead address.

Recommendation Remove these checks.

Client Comment *At least it checks for empty (zero) addresses. We prefer to keep it.*

```
373 if (_destination == address(0)) revert Errors.EmptyAddress();
    if (_stablecoin == address(0)) revert Errors.EmptyAddress();
```

CVF-100 FIXED

- **Category** Procedural
- **Source** NexaManager.sol

Description This check is redundant, as will anyway be performed when transferring tokens.

Recommendation Remove these checks.

Client Comment *Removed.*

```
376 if (IERC20(_stablecoin).balanceOf(address(this)) < _stablecoinAmount
      ↪ ) revert Errors.NotEnoughBalance();
```

CVF-101 INFO

- **Category** Procedural
- **Source** NexaManager.sol

Description These checks are redundant, as it is anyway possible to pass dead addresses.

Recommendation Remove these checks.

Client Comment *At least it checks for empty (zero) addresses. We prefer to keep it.*

```
398 if (_user == address(0)) revert Errors.EmptyAddress();
    if (address(_portfolio) == address(0)) revert Errors.EmptyAddress();
```

CVF-102 FIXED

- **Category** Procedural
- **Source** NexaManager.sol

Description This check is redundant, as it will anyway be performed when burning tokens.

Recommendation Remove this check.

Client Comment *Removed.*

```
401 if (ERC20Base(address(_portfolio)).balanceOf(_user) < _shares)
       ↪ revert Errors.NotEnoughShares();
```

CVF-103 INFO

- **Category** Procedural
- **Source** NexaManager.sol

Description These checks are redundant, as it is anyway possible to pass dead addresses.

Recommendation Remove these checks.

Client Comment *At least it checks for empty (zero) addresses. We prefer to keep it.*

```
447 if (_user == address(0)) revert Errors.EmptyAddress();
    if (address(_portfolio) == address(0)) revert Errors.EmptyAddress();
```

CVF-104 FIXED

- **Category** Procedural
- **Source** NexaManager.sol

Description This check is redundant as it will anyway be performed when transferring tokens.

Recommendation Remove this check.

Client Comment *Removed.*

```
450 if (ERC20Base(address(_portfolio)).balanceOf(address(this)) <
       ↪ _shares) revert Errors.NotEnoughShares();
```

```
461     if (ERC20Base(address(underlayingTokens[i])).balanceOf(_user) <
           ↪ total) revert Errors.NotEnoughShares();
```

CVF-105 FIXED

- **Category** Bad naming
- **Source** NexaManager.sol

Description The function name is misleading, as the function actually returns the index of an asset rather then the asset itself.

Recommendation Rename to “findUserAsset” or something similar.

Client Comment *Renamed.*

```
500 function getAssetByUser(address _user, address _asset) public view
       ↪ returns (bool, uint256) {
```

CVF-106 FIXED

- **Category** Bad naming
- **Source** NexaManager.sol

Description These names are too similar.

Recommendation Use more distinguishable names.

Client Comment *Renamed.*

```
516 function getAssetsByUser(address _user) public view returns (address
       ↪ [] memory) {
```

```
522 function getAssetsForUser(address _user) external view returns (
       ↪ UserAsset[] memory) {
```

CVF-107 FIXED

- **Category** Unclear behavior
- **Source** NexaManager.sol

Description This event is emitted even if nothing actually changed.

Client Comment *Only emit events if the value changed.*

```
573 emit PortfolioManagerSet(_portfolioManager);
```

CVF-108 FIXED

- **Category** Suboptimal
- **Source** NexaManager.sol

Description There is no range check for the “_dueDate” argument.

Recommendation Add an appropriate check.

Client Comment *Set a max due date of 100 years.*

```
579 function setDueDate(address _token, uint256 _dueDate) external
      ↪ onlyRole(DEFAULT_ADMIN_ROLE) {
```

ABDK
Consulting

About us

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.

Contact

Email

dmitry@abdkconsulting.com

Website

abdk.consulting

Twitter

twitter.com/ABDKconsulting

LinkedIn

linkedin.com/company/abdk-consulting