Report

v. 1.0

Customer

Ondo

Smart Contract Audit

Ondo Protocol

26th October 2022

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	25.10.22	A. Zveryanskaya	Initial Draft
0.2	26.10.22	A. Zveryanskaya	Minor revision
1.0	26.10.22	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.

Ondo Finance is an open, permissionless, decentralized investment bank. Ondo's core business is to service and connect various stakeholders in the emerging DeFi ecosystem - including DAOs and increasingly institutional and mainstream retail investors - through fully on-chain services.

3 Project scope

We were asked to review:

- Original Repository
- Fix Repository

Files:

/			
	OndoRegistryClient Initializable.sol	OndoRegistryClient.sol	Registry.sol
	Multiex.sol		
interfaces/			
	IMultiex.sol	IPairVault.sol	IRegistry.sol
	IRollover.sol	ISAStrategy.sol	ISingleAssetVault.sol
	IStrategy.sol.sol	ITrancheToken.sol	IWETH.sol
libraries/			
	OndoLibrary.sol		
single/			
	SAStrategy AllPairVault.sol	SAStrategyConvex.sol	SAStrategyRollover.sol
	SingleAssetVault.sol		
strategies/			
	AConvex Autocompounder.sol	BalancerStrategy.sol	BasePairLPStrategy.sol
	Convex Autocom-pounderStrategy.sol		
vendor/balancer/			
	IBalancerVault.sol		
vendor/convex/			
	IBaseRewardPool.sol	IConvexBooster.sol	
vendor/curve/			
	ICurve_2.sol	ICurve_3.sol	

4 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 used technologies, as well as on what the client is expecting from the audit.

- **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.

5 Our findings

We found 1 critical, 4 major, and a few less important issues. All identified Critical issues have been fixed.

Severity	Active	Fixed
Critical	0	1
Major	2	2
Moderate	0	7
Minor	117	62

Fixed 72 out of 191 issues

6 Critical Issues

CVF-121. FIXED

- **Category** Flaw
- **Source** SingleAssetVault.sol

Description It is not checked that the pool is not yet redeemed, so it is possible to redeem a pool several times.

```
pool.redeemed = true;
```

7 Major Issues

CVF-17. FIXED

- **Category** Documentation
- **Source** Registry.sol

Description Despite the name, this function is not only able to add a new strategist, but also change the name of an existing strategist.

Recommendation Consider either mentioning this fact in the documentation comment, or explicitly forbid adding the same strategist twice.

```
74 function addStrategist(address _strategist, string calldata _name)
     ↪ external {
```

CVF-31. INFO

- **Category** Flaw
- **Source** OndoRegistryClientInitializable.sol

Description There is no check that the lengrths of the "_tokens" and "_amounts" arrays are the same. In case the "_amounts" array is longer, extra element are silently ignored.

Recommendation Consider adding an appropriate check.

Client Comment *Checked in the external function which implements the internal _rescueTokens() function, won't fix.*

```
60 function _rescueTokens(address[] calldata _tokens, uint256[] memory
     ↪ _amounts)
```

CVF-68. FIXED

- **Category** Flaw
- **Source** AConvexAutocompounder.sol

Description It is not checked that the “_assets” and “_paths” arrays have the same length. If the “_paths” array is longer than the “_assets” array, extra elements are silently ignored.

Recommendation Consider adding appropriate length check.

```
function setSwapPaths(address[] memory _assets, SwapPath[] memory
    ↪ _paths)
```

CVF-149. INFO

- **Category** Suboptimal
- **Source** OndoLibrary.sol

Description This performs three calls to the token contract instead of one, which is overkill in most cases and just wastes gas. Usually, smart contracts just overwrite the current allowance before calling a contract that needs to take tokens. In case the whole allowance is used, there is no need to set to to zero explicitly.

Recommendation Consider simplifying.

Client Comment *Gas cost is absorbed by us, won’t fix.*

```
uint256 newAllowance = token.allowance(address(this), spender) +
    ↪ value;
token.safeApprove(spender, 0);
token.safeApprove(spender, newAllowance);
```

8 Moderate Issues

CVF-7. FIXED

- **Category** Flaw
- **Source** ConvexAutocompounderStrategy.sol

Description This function is callable by anyone.

Recommendation Consider restricting access to it.

```
function invest(
```

CVF-90. FIXED

- **Category** Flaw
- **Source** SAStrategyRollover.sol

Description These functions are not marked with the “nonReentrant” modifier, while they do update state after calling potentially untrusted external contracts and thus are potentially vulnerable to reentrancy attacks.

Recommendation Consider ether using the “nonReentrant” modifier for these functions or refactoring their code to do follow check→update→call execution order.

```
function invest(
```

```
function redeem(uint256 poolId, bytes memory data)
```

CVF-91. FIXED

- **Category** Flaw
- **Source** SAStrategyRollover.sol

Description The state is checked before calling external contracts, while updated afterwards. This makes the function potentially vulnerable to reentrancy attack.

Recommendation Consider refactoring the code or using the “nonReentrant” modifier.

```
require(!rolloverInvested[param.rolloverId], "rollover invested");
```

```
asset.safeApprove(address(rolloverManager), amount);
rolloverManager.deposit(param.rolloverId, param.tranche, amount);
rolloverInvested[param.rolloverId] = true;
```

CVF-92. FIXED

- **Category** Flaw
- **Source** SAStrategyRollover.sol

Description The state is checked before calling external contracts, while updated afterwards. This makes the function potentially vulnerable to reentrancy attack.

Recommendation Consider refactoring the code or using the “nonReentrant” modifier.

```
require(rolloverInvested[investParam.rolloverId], "rollover not
   ↪ invested");
```

```
  vaultManager.withdraw(investParam.rolloverId, investParam.tranche)
     ↪ ;
```

```
  asset.safeTransfer(msg.sender, amount);
```

```
  rolloverInvested[investParam.rolloverId] = false;
```

CVF-104. FIXED

- **Category** Suboptimal
- **Source** SAStrategyConvex.sol

Description The function actually doesn't return anything for the first return value.

Recommendation Consider either removing this return value from the function declaration or actually returning something for it.

```
returns (bool, uint256 amount)
```

CVF-122. FIXED

- **Category** Suboptimal
- **Source** SingleAssetVault.sol

Description This reads all the deposits into the memory, even those before "fromDepositIndex[_user]".

Recommendation Consider loading only relevant deposits.

```
UserDeposit[] memory deposits = userDeposits[_user];
```

CVF-133. FIXED

- **Category** Flaw
- **Source** SAStrategyAllPairVault.sol

Description The state is updated after calling untrusted external contracts, which could make a reentrancy attack possible.

Recommendation Consider refactoring the code to do state updates before calling untrusted external contracts, i.e. before performing token transfers.

```
vaultInvested[param.vaultId] = true;
```

9 Minor Issues

CVF-1. FIXED

- **Category** Suboptimal
- **Source** ConvexAutocompounderStrategy.sol

Description This is equivalent to: pragma solidity ^0.8.0;. Also relevant for the next files: AConvexAutocompounder.sol, SAStrategyRollover.sol, SAStrategyConvex.sol, SingleAssetVault.sol, SAStrategyAllPairVault.sol, BaseRewardPool.sol, IConvexBooster.sol, ICurve_2.sol, ISingleAssetVault.sol, ISAStrategy.sol, IRollover.sol, IMultiex.sol.

```
pragma solidity >=0.8.0 <0.9.0;
```

CVF-2. INFO

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

Recommendation The type of this argument should be "IERC20".

Client Comment *We have adopted in our team's solidity style guide to generally prefer the more generic 'address' datatype.*

```
address _stableAsset,
```

CVF-3. INFO

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

Recommendation The type of this argument should be "IRegistry".

Client Comment *We have adopted in our team's solidity style guide to generally prefer the more generic 'address' datatype.*

```
address _registry,
```

CVF-4. INFO

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

Recommendation The type of this argument should be “IERC20[]”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address[] memory _rewardTokens
```

CVF-5. FIXED

- **Category** Procedural
- **Source** ConvexAutocompounderStrategy.sol

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

```
{}
```

CVF-6. INFO

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

Description This function should probably emit some event

Client Comment *Its a function called by AllPairVault to add a vault in the strategy, an event is not needed.*

```
function addVault(
```

CVF-8. FIXED

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

Description The semantics of the returned values is unclear.

Recommendation Consider giving them descriptive names.

```
64  ) external override nonReentrant returns (uint256, uint256) {
```

```
111 ) external override nonReentrant returns (uint256, uint256) {
```

CVF-9. FIXED

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

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

Recommendation Consider using the “mulDiv” function as described here: https://xn–2-mb.com/21/muldiv/index.html or some other approach resistant to phantom overflow.

```
91  shares = (newLpAmounts * totalSupply) / prevLpAmounts;
```

```
119 amount += _withdrawLP(i, (lpAmounts[i] * shares) / totalSupply);
```

CVF-10. INFO

- **Category** Procedural
- **Source** ConvexAutocompounderStrategy.sol

Description Brackets are redundant.

Recommendation Consider removing them.

Client Comment *Brackets help readability.*

```
91  shares = (newLpAmounts * totalSupply) / prevLpAmounts;
```

CVF-11. INFO

- **Category** Procedural
- **Source** Registry.sol

Description Specifying a particular compiler version makes it harder to migrate to newer versions. Also revent for the next files:IBalancerVault.sol, BasePairLPStrategy.sol, OndoRegistryClientInitializable.sol, OndoRegistryClient.sol, Multiex.sol, BalancerStrategy.sol, OndoLibrary.sol, IWETH.sol, IRegistry.sol, IStrategy.sol, IPairVault.sol, ITrancheToken.sol.

Recommendation Consider specifying "^0.8.0".

Client Comment *The assumptions we have current are based on 0.8.3 and when we want to upgrade, we would have to consider the changes and make them explicit.*

```
pragma solidity 0.8.3;
```

CVF-12. FIXED

- **Category** Suboptimal
- **Source** Registry.sol

Description This variable is never read.

Recommendation Consider removing it.

```
address payable public fallbackRecipient;
```

CVF-13. INFO

- **Category** Procedural
- **Source** Registry.sol

Description Modifier with the same name and semantics i already defined in the "Access-Control" base contract.

Recommendation Consider removing duplicate definition.

Client Comment *Not duplicate as we are on old version of openzeppelin contracts.*

```
modifier onlyRole(bytes32 _role) {
  require(hasRole(_role, msg.sender), "Unauthorized: Invalid role");
  _;
}
```

CVF-14. INFO

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

Recommendation The type of this argument should be "IWETH".

Client Comment *We have adopted in our team's solidity style guide to generally prefer the more generic 'address' datatype.*

```
address _weth
```

CVF-15. INFO

- **Category** Suboptimal
- **Source** Registry.sol

Description Checks against zero address are redundant, as it is anyway possible to to pass a dead address as an argument.

Client Comment *Style guide conflict, won't fix.*

```
require(
  _fallbackRecipient != address(0) && _fallbackRecipient != address(
    ↪ this),
  "Invalid address"
);
require(_governance != address(0), "Invalid governance address");
require(_weth != address(0), "Invalid weth address");
```

CVF-16. INFO

- **Category** Suboptimal
- **Source** Registry.sol

Description This function is an alias for the "hasRole" function.

Recommendation Consider removing it.

Client Comment *Style guide conflict, won't fix.*

```
function authorized(bytes32 _role, address _account)
```

CVF-18. FIXED

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

Description These functions should emit some events.

```
82  function enableFeatureFlag(bytes32 _featureFlag)
```

```
93  function disableFeatureFlag(bytes32 _featureFlag)
```

```
116 function deleteFeatureFlag(bytes32 _featureFlag)
```

CVF-19. FIXED

- **Category** Suboptimal
- **Source** Registry.sol

Description These two functions basically do the same thing.

Recommendation Consider removing one of them.

```
93  function disableFeatureFlag(bytes32 _featureFlag)
```

```
116 function deleteFeatureFlag(bytes32 _featureFlag)
```

CVF-20. FIXED

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

Description These events are emitted even if nothing actually changed.

```
146 emit Paused(msg.sender);
```

```
154 emit Unpaused(msg.sender);
```

CVF-21. FIXED

- **Category** Documentation
- **Source** IBalancerVault.sol

Description The “payable” modifier was removed from structure fields but not from from functions.

Recommendation Consider explaining this.

```
) external payable;
```

```
) external payable returns (uint256);
```

```
) external payable returns (int256[] memory);
```

CVF-22. FIXED

- **Category** Documentation
- **Source** IBalancerVault.sol

Description This comment is confusing.

Recommendation Consider elaborating a bit more regarding why the “payable” modifier was removed.

```
address recipient, // NOTE: payable -> non payable
```

CVF-23. INFO

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

Recommendation The argument type should be “IRegistry”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
constructor(address _registry) OndoRegistryClient(_registry) {}
```

CVF-24. INFO

- **Category** Procedural
- **Source** BasePairLPStrategy.sol

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

Client Comment *Style guide conflict, won't fix.*

```
constructor(address _registry) OndoRegistryClient(_registry) {}
```

CVF-25. INFO

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

Description The semantics of the returned values is unclear.

Recommendation Consider giving them descriptive names and or describing in the documentation comment.

Client Comment *Already documented, won't fix.*

```
returns (IERC20, uint256)
```

CVF-26. INFO

- **Category** Suboptimal
- **Source** BasePairLPStrategy.sol

Description This implicitly assumes the tranche is junior. } else revert ();

Recommendation Consider making this assumption explicit like this: else if (tranche == OLib.Tranche.Junior) { ...

Client Comment *Won't fix, only have 2 tranches and slight gas savings with current code.*

```
} else {
```

CVF-27. INFO

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

Recommendation The argument type should be “IRegistry”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
function __OndoRegistryClient__initialize(address _registry)
```

CVF-28. INFO

- **Category** Suboptimal
- **Source** OndoRegistryClientInitializable.sol

Description This check is redundant. It is anyway possible to pass a dead registry address.

Recommendation Consider removing this check.

Client Comment *Won’t fix.*

```
require(_registry != address(0), "Invalid registry address");
```

CVF-29. INFO

- **Category** Suboptimal
- **Source** OndoRegistryClientInitializable.sol

Recommendation It would be more efficient to check “super.paused()” first as it is cheaper that checking “registry.paused()”.

Client Comment *We pause from Registry not RegistryClientInitializable, won’t fix.*

```
return registry.paused() || super.paused();
```

CVF-30. INFO

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

Recommendation The type of the “_tokens” argument should be “IERC20[]”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
60 function _rescueTokens(address[] calldata _tokens, uint256[] memory
    ↪ _amounts)
```

CVF-32. INFO

- **Category** Suboptimal
- **Source** OndoRegistryClientInitializable.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 *Leaving as is, won’t fix.*

```
60 function _rescueTokens(address[] calldata _tokens, uint256[] memory
    ↪ _amounts)
```

CVF-33. INFO

- **Category** Suboptimal
- **Source** OndoRegistryClientInitializable.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 *Leaving as is, won’t fix.*

```
73 function rescueTokens(address[] calldata _tokens, uint256[] memory
    ↪ _amounts)
```

CVF-34. INFO

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

Recommendation The argument type should be “IRegistry”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
constructor(address _registry) {
```

CVF-35. INFO

- **Category** Procedural
- **Source** Multiex.sol

Description This comment seems irrelevant.

Recommendation Consider removing it.

Client Comment *Keeping comment, won’t fix.*

```
* @notice Send all fee directly to creator
```

CVF-36. FIXED

- **Category** Procedural
- **Source** BalancerStrategy.sol

Recommendation These variables should be declared as immutable.

```
IBalancerVault public balVault;
```

```
address public balPool;
```

CVF-37. INFO

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

Recommendation The type of this variable should be “IERC20”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address public balPool;
```

CVF-38. INFO

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

Recommendation The type of this argument should be “IRegistry”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address _registry,
```

CVF-39. INFO

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

Recommendation The type of this argument should be “IBalancerVault”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address _balVault,
```

CVF-40. FIXED

- **Category** Suboptimal
- **Source** BalancerStrategy.sol

Recommendation Should be “!=” instead of “>”, as using “>” for addresses looks weird.

```
require(balPool > address(0), "Invalid Pool Id"); // Balancer pool address is a part of pool id, so, should check here.
```

CVF-41. INFO

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

Description These functions should emit some events.

Client Comment *Won’t fix.*

```
function addVault(
```

```
function addLp(uint256 _vaultId, uint256 _amount)
```

```
function removeLp(
```

```
function rebalanceIncomes(
```

CVF-42. FIXED

- **Category** Suboptimal
- **Source** BalancerStrategy.sol

Description The expression “vaults[_vaultId]” is calculated twice.

Recommendation Consider calculating once and reusing.

```
  address(vaults[_vaultId].origin) == address(0),
```

```
Vault storage vault = vaults[_vaultId];
```

CVF-43. INFO

- **Category** Suboptimal
- **Source** BalancerStrategy.sol

Recommendation This check makes the "_seniorToken" and the "_juniorToken" arguments redundant. It would be enough to just pass a boolean argument telling whether token order is straight or reversed.

Client Comment *Won't fix*

```
require(
  (tokenA == _seniorToken && tokenB == _juniorToken) ||
    (tokenB == _seniorToken && tokenA == _juniorToken),
  "Unsupported vault"
);
```

CVF-44. INFO

- **Category** Readability
- **Source** BalancerStrategy.sol

Recommendation Consider giving descriptive names to the returned values for readability.

Client Comment *Style guide conflict, won't fix*

```
  uint256,
  uint256,
  IERC20
```

```
returns (uint256, uint256)
```

CVF-45. INFO

- **Category** Procedural
- **Source** BalancerStrategy.sol

Description Usually, contracts just overwrite the new allowance, rather than increase it.

Recommendation Consider doing so.

Client Comment *Won't fix.*

203
```
tokenSenior.ondoSafeIncreaseAllowance(address(balVault),
   ↪ _totalSenior);
tokenJunior.ondoSafeIncreaseAllowance(address(balVault),
   ↪ _totalJunior);
```

300
```
IERC20(balPool).ondoSafeIncreaseAllowance(address(balVault), bptIn);
```

507
```
_tokenIn.ondoSafeIncreaseAllowance(address(balVault), _amountIn);
```

550
```
_tokenIn.ondoSafeIncreaseAllowance(address(balVault), _maxAmountIn);
```

CVF-46. FIXED

- **Category** Suboptimal
- **Source** BalancerStrategy.sol

Recommendation These lines could be simplified using the "-=" operator.

242
```
seniorInvested = seniorInvested - tokenSenior.balanceOf(address(this
   ↪ ));
juniorInvested = juniorInvested - tokenJunior.balanceOf(address(this
   ↪ ));
```

CVF-47. INFO

- **Category** Suboptimal
- **Source** BalancerStrategy.sol

Description There is no check that the vault is not yet invested, so this could overwrite the current shares value.

Recommendation Consider adding an appropriate check.

Client Comment *Incorrect suggestion, won't fix.*

```
vault.shares = bptOut;
```

CVF-48. FIXED

- **Category** Procedural
- **Source** BalancerStrategy.sol

Description The explicit initialization is redundant. According to the documentation, elements of arrays, allocated in the memory, are initialized with default value: https://docs.soliditylang.org/en/v0.8.12/types.html#allocating-memory-arrays For the "uint256" type, default value is zero.

```
/// @NOTE need to initialize??? yes, the memory may or may not be zeroed out.
(amountsOut[0], amountsOut[1]) = (0, 0);
```

CVF-49. INFO

- **Category** Suboptimal
- **Source** BalancerStrategy.sol

Description Increasing allowance instead of transferring tokens looks weird. Also, it consumes more gas than normal token transfer.

Recommendation Consider just transferring tokens, instead of increasing allowance.

Client Comment *Won't fix.*

```
tokenSenior.ondoSafeIncreaseAllowance(
  msg.sender,
  seniorReceived + vault.seniorExcess
);
tokenJunior.ondoSafeIncreaseAllowance(
  msg.sender,
  juniorReceived + vault.juniorExcess
);
```

CVF-50. INFO

- **Category** Suboptimal
- **Source** BalancerStrategy.sol

Recommendation Consider also logging the number of newly accrued tokens

Client Comment *Relevant info already logged from AllPair redeem event, won't fix.*

```
emit Redeem(_vaultId);
```

CVF-51. INFO

- **Category** Readability
- **Source** BalancerStrategy.sol

Recommendation Should be "else if' for readability.

Client Comment *Style guide conflict, won't fix.*

```
if (_seniorReceived > _seniorExpected) {
```

CVF-52. INFO

- **Category** Readability
- **Source** BalancerStrategy.sol

Description The code below looks like it is always executed, while it is executed only when _seniorReceiver < _seniorExpected.

Recommendation Consider putting the code below into an explicit “else” branch for readability.

Client Comment *Style guide conflict, won’t fix.*

```
}
```

CVF-53. INFO

- **Category** Readability
- **Source** BalancerStrategy.sol

Description The code below looks like it is always executed, while it is executed only when the received junior amount is enough to fulfill the senior expectation.

Recommendation Consider putting the code below into an explicit “else” branch for readability.

Client Comment *Style guide conflict, won’t fix.*

```
}
```

CVF-54. FIXED

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

Description This conversion looks like underflow is possible here, and the code relies on knowledge about how balancer works.

Recommendation Consider using safe conversion to not rely on external code.

```
amountOut = uint256(
```

CVF-55. INFO

- **Category** Suboptimal
- **Source** BalancerStrategy.sol

Description These functions are very similar.

Recommendation Consider merging them into one function or extracting common code to a utility function.

Client Comment *Style guide conflict, won't fix.*

```
function swapExactIn(
```

```
function swapExactOut(
```

CVF-56. FIXED

- **Category** Suboptimal
- **Source** AConvexAutocompounder.sol

Description This import is not used.

```
import "contracts/interfaces/IRollover.sol";
```

CVF-57. INFO

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

Recommendation The type of this field should be "IUniswapV2Router".

Client Comment *We have adopted in our team's solidity style guide to generally prefer the more generic 'address' datatype*

```
address router;
```

CVF-58. INFO

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

Recommendation The type of this field should be “IERC20[]”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype*

```
address[] path;
```

CVF-59. INFO

- **Category** Suboptimal
- **Source** AConvexAutocompounder.sol

Description Hardcoding Mainnet addresses is a bad practice as it makes it harder to test contracts.

Recommendation Consider passing the addresses as constructor arguments and storing in immutable variables.

Client Comment *Style guide conflict, won’t fix*

```
  IERC20(0x6c3F90f043a72FA612cbac8115EE7e52BDe6E490);
```

```
  ICurve_3(0xbEbc44782C7dB0a1A60Cb6fe97d0b483032FF1C7);
```

```
  IConvexBooster(0xF403C135812408BFbE8713b5A23a04b3D48AAE31);
address public constant DAI = 0
   ↪ x6B175474E89094C44Da98b954EedeAC495271d0F;
address public constant USDC = 0
   ↪ xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48;
address public constant USDT = 0
   ↪ xdAC17F958D2ee523a2206206994597C13D831ec7;
```

CVF-60. INFO

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

Recommendation The type of these constants should be “IERC20”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address public constant DAI = 0
   ↪ x6B175474E89094C44Da98b954EedeAC495271d0F;
address public constant USDC = 0
   ↪ xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48;
address public constant USDT = 0
   ↪ xdAC17F958D2ee523a2206206994597C13D831ec7;
```

CVF-61. INFO

- **Category** Documentation
- **Source** AConvexAutocompounder.sol

Description The semantics of the keys in this mapping is unclear.

Recommendation Consider documenting.

Client Comment *Style guide conflict, won’t fix.*

```
mapping(uint256 => uint256) public balanceOf;
```

```
mapping(address => SwapPath) public swapPaths;
```

CVF-62. INFO

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

Recommendation The type of this variable should be “IERC20”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address[] public rewardTokens;
```

CVF-63. INFO

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

Recommendation The key type for this mapping should be “IERC20”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
mapping(address => SwapPath) public swapPaths;
```

CVF-64. INFO

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

Recommendation The type of this argument should be “IERC20”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address _stableAsset,
```

CVF-65. INFO

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

Recommendation The type of this argument should be “IRegistry”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address _registry,
```

CVF-66. INFO

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

Recommendation The type of this argument should be “IERC20[]”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address[] memory _rewardTokens
```

CVF-67. INFO

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

Recommendation These indexes should be named constants.

Client Comment *Style guide conflict, won’t fix.*

```
stableAssetCurveIndex = 0;
```

```
stableAssetCurveIndex = 1;
```

```
stableAssetCurveIndex = 2;
```

CVF-69. INFO

- **Category** Suboptimal
- **Source** AConvexAutocompounder.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 *Style guide conflict, won’t fix*

```
function setSwapPaths(address[] memory _assets, SwapPath[] memory
   ↪ _paths)
```

CVF-70. INFO

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

Recommendation The type of the “_assets” array should be “IERC20”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
114 function setSwapPaths(address[] memory _assets, SwapPath[] memory
      ↪ _paths)
```

CVF-71. INFO

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

Recommendation The argument type should be “IERC20[]”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
127 function setRewardTokens(address[] memory _rewardTokens)
```

CVF-72. INFO

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

Description These functions should probably emit some events

Client Comment *Fine as is, won't fix.*

```
function setRewardTokens(address[] memory _rewardTokens)
```

```
function compound() external isAuthorized(OLib.STRATEGIST_ROLE) {
```

```
function claim() public isAuthorized(OLib.STRATEGIST_ROLE) {
```

```
function swapRewardTokensToStableAsset()
```

```
function splitAndDepositConvex()
```

CVF-73. FIXED

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

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

Recommendation Consider using the "mulDiv" function as described here: https://xn--2-mb.com/21/muldiv/index.html

```
(total3CRVAmount * setting.allocPoints) / totalAllocPoints
```

CVF-74. INFO

- **Category** Procedural
- **Source** AConvexAutocompounder.sol

Recommendation It is a good practice when passing a boolean literal as an argument to put the argument name as a comment for readability.

Client Comment *Style guide conflict, won’t fix.*

```
203 CONVEX_BOOSTER.deposit(setting.cvxPID, newLpAmount, true);
```

CVF-75. FIXED

- **Category** Suboptimal
- **Source** AConvexAutocompounder.sol

Description Approving twice before each swap is waste of gas.

Recommendation Consider approving the maximum amount only once.

```
215 IERC20(rewardAsset).safeApprove(data.router, 0);
    IERC20(rewardAsset).safeApprove(data.router, amount);
```

CVF-76. FIXED

- **Category** Suboptimal
- **Source** AConvexAutocompounder.sol

Description This is redundant for most of the tokens, but does consume extra gas.

Recommendation Consider not doing this when not needed. For example. consider adding a per-token flag telling whether this additional “approve” call is needed for the token.

```
215 IERC20(rewardAsset).safeApprove(data.router, 0);
```

CVF-77. INFO

- **Category** Suboptimal
- **Source** AConvexAutocompounder.sol

Recommendation This code could be simplified using an array literal.

Client Comment *Won't fix.*

```
uint256[3] memory amounts;
amounts[0] = IERC20(DAI).balanceOf(address(this));
amounts[1] = IERC20(USDC).balanceOf(address(this));
amounts[2] = IERC20(USDT).balanceOf(address(this));
```

CVF-78. FIXED

- **Category** Flaw
- **Source** AConvexAutocompounder.sol

Description It is not checked that the second token is "THREE_CRV_LP".

Recommendation Consider adding such check.

```
} else {
```

```
  } else {
```

CVF-79. FIXED

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

Description This conversion looks like it could cause overflow.

Recommendation Consider using a smaller type for "stableAssetCurveIndex", such as "uint8" to make this conversion safe.

```
int128(stableAssetCurveIndex),
```

CVF-80. FIXED

- **Category** Suboptimal
- **Source** AConvexAutocompounder.sol

Description Here implicit underflow checks are used to enforce business-level constraints. This is a bad practice as it makes code harder to read and more fragile.

Recommendation Consider explicitly checking that the balance is sufficient.

```
balanceOf[poolId] -= amount;
totalSupply -= amount;
```

CVF-81. FIXED

- **Category** Procedural
- **Source** SAStrategyRollover.sol

Recommendation The type of this field should be a enum.

```
uint256 redeemType; // 0: redeem from rollover, 1: redeem from
    ↪ allPairVault
```

CVF-82. FIXED

- **Category** Documentation
- **Source** SAStrategyRollover.sol

Description The semantics of the keys in these mappings is unclear.

Recommendation Consider documenting.

```
mapping(uint256 => bool) public rolloverInvested;
mapping(uint256 => uint256) public rolloverExcees;
mapping(uint256 => uint256) public rolloverRedeemVaultId;
mapping(uint256 => InvestParam) public investParams;
```

CVF-83. FIXED

- **Category** Suboptimal
- **Source** SAStrategyRollover.sol

Recommendation It would be more efficient to merge these three mappings into a single mapping whose keys are rollover IDs and values are structs of three fields encapsulating the values of the original mappings.

31
```
mapping(uint256 => bool) public rolloverInvested;
mapping(uint256 => uint256) public rolloverExcees;
mapping(uint256 => uint256) public rolloverRedeemVaultId;
```

CVF-84. FIXED

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

Recommendation The “rolloverId” parameter should be indexed.

36
```
event Invest(uint256 rolloverId, OLib.Tranche tranche, uint256
    ↪ amount);
event Redeem(uint256 rolloverId, OLib.Tranche tranche, uint256
    ↪ amount);
```

CVF-85. INFO

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

Recommendation The type of this argument should be “ISingleAssetVault”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

47
```
address _singleAssetVault,
```

CVF-86. INFO

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

Recommendation The type of this argument should be “IPairVault”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address _vault,
```

CVF-87. INFO

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

Recommendation The type of this argument should be “IRollover”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address _rollover,
```

CVF-88. INFO

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

Recommendation The type of this argument should be “IRegistry”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address _registry
```

CVF-89. INFO

- **Category** Suboptimal
- **Source** SAStrategyRollover.sol

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

Client Comment *Keeping checks, won't fix.*

```
require(_singleAssetVault != address(0), "Invalid asset vault");
require(_vault != address(0), "Invalid AllPairVault");
require(_rollover != address(0), "Invalid RolloverVault");
```

CVF-93. FIXED

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

Recommendation "0" and "1" here should be named constants.

```
if (redeemParam.redeemType == 0) {
```

```
} else if (redeemParam.redeemType == 1) {
```

CVF-94. FIXED

- **Category** Suboptimal
- **Source** SAStrategyRollover.sol

Description The withdrawn balance is returned from the "withdraw" function call. No need to calculate it from the token balances.

Recommendation Consider either using the returned value or explaining in a comment why the returned value cannot be trusted.

```
uint256 balanceBeforeWithdraw = asset.balanceOf(address(this));
```

```
  asset.balanceOf(address(this)) -
  balanceBeforeWithdraw;
```

CVF-95. FIXED

- **Category** Suboptimal
- **Source** SAStrategyRollover.sol

Recommendation The withdrawn balance is returned from the “withdraw” function call. No need to calculate it from the token valances.

```
amount = asset.balanceOf(address(this));
```

```
amount = asset.balanceOf(address(this)) - amount;
```

CVF-96. INFO

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

Recommendation The type of this argument should be “IERC20”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address _stableAsset,
```

CVF-97. INFO

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

Recommendation The type of this argument should be “ISingleAssetVault”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address _singleAssetVault,
```

CVF-98. INFO

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

Recommendation The type of this argument should be “IRegistry”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address _registry,
```

CVF-99. INFO

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

Recommendation The type of this argument should be “IERC20[]”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address[] memory _rewardTokens
```

CVF-100. INFO

- **Category** Suboptimal
- **Source** SAStrategyConvex.sol

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

Recommendation Consider removing this check.

Client Comment *Style guide conflict, won’t fix.*

```
require(_singleAssetVault != address(0), "Invalid asset vault");
```

CVF-101. INFO

- **Category** Readability
- **Source** SAStrategyConvex.sol

Recommendation Consider initializing to 0 for readability.

Client Comment *Style guide conflict, won't fix.*

```
46 uint256 prevLpAmounts;
```

CVF-102. INFO

- **Category** Suboptimal
- **Source** SAStrategyConvex.sol

Description The "totalSupply" value is read from the storage twice (and one again inside the "_mint" function.

Recommendation Consider refactoring the code to read it only once.

Client Comment *Won't fix.*

```
55 if (totalSupply == 0) {
```

```
58   shares = (newLpAmounts * totalSupply) / prevLpAmounts;
```

CVF-103. FIXED

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

Description Phantom overflow is possible here.

Recommendation Consider using a safe mulDiv function.

```
58 shares = (newLpAmounts * totalSupply) / prevLpAmounts;
```

```
80 amount += _withdrawLP(i, (lpAmounts[i] * shares) / totalSupply);
```

CVF-105. INFO

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

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

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
mapping(address => bool) public isStrategy;
```

CVF-106. INFO

- **Category** Documentation
- **Source** SingleAssetVault.sol

Description The semantics of the key and value for this mapping in unclear.

Recommendation Consider documenting.

Client Comment *Style guide conflict, won’t fix.*

```
mapping(address => uint256) internal fromDepositIndex; // calculate
    ↪ token balance from this deposit index
```

CVF-107. INFO

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

Description The name is confusing.

Recommendation Consider renaming.

Client Comment *Style guide conflict, won’t fix.*

```
mapping(address => uint256) internal fromDepositIndex; // calculate
    ↪ token balance from this deposit index
```

CVF-108. FIXED

- **Category** Suboptimal
- **Source** SingleAssetVault.sol

Description These mappings are not used.

Recommendation Consider removing them.

```
mapping(uint256 => bool) internal vaultInvested;
mapping(uint256 => bool) internal rolloverInvested;
```

CVF-109. INFO

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

Recommendation The type of the “_asset” argument should be “IERC20”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
constructor(address _asset, address _registry) OndoRegistryClient(
   ↪ _registry) {
```

CVF-110. INFO

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

Recommendation The type of the “_registry” argument should be “IRegistry”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
constructor(address _asset, address _registry) OndoRegistryClient(
   ↪ _registry) {
```

CVF-111. INFO

- **Category** Suboptimal
- **Source** SingleAssetVault.sol

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

Client Comment *Won't fix.*

```
require(_asset != address(0), "Invalid asset");
```

CVF-112. INFO

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

Recommendation The type of the "_strategy" argument should be "IStrategy".

Client Comment *We have adopted in our team's solidity style guide to generally prefer the more generic 'address' datatype.*

```
function setStrategy(address _strategy, bool _flag)
```

CVF-113. INFO

- **Category** Documentation
- **Source** SingleAssetVault.sol

Description The semantics of the "_flag" argument is unclear from its name.

Recommendation Consider documenting.

Client Comment *Style guide conflict, won't fix.*

```
function setStrategy(address _strategy, bool _flag)
```

CVF-114. INFO

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

Description These functions should emit some events.

Client Comment *Won't fix.*

```
function setStrategy(address _strategy, bool _flag)
```

```
function setWithdrawEnabled(bool _withdrawEnabled)
```

CVF-115. INFO

- **Category** Suboptimal
- **Source** SingleAssetVault.sol

Description This logic is coded several times.

Recommendation Consider extracting to a function to reduce code duplication.

Client Comment *Style guide conflict, won't fix.*

```
userDeposits[msg.sender].push(
  UserDeposit({amount: _amount, firstActionId: actions.length})
);
```

```
  userDeposits[msg.sender].push(
    UserDeposit({
      amount: remainAmount - _amount,
      firstActionId: actions.length
    })
  );
```

```
  userDeposits[msg.sender].push(
    UserDeposit({amount: remainAmount, firstActionId: actions.length
      ↪ })
  );
```

CVF-116. INFO

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

Description Overflow is possible here that would revert the transaction.

Recommendation Consider using bitwise “or” instead of “+”.

Client Comment *Won’t fix*

```
require(activeInvestAmount + passiveInvestAmount == 0, "invested!");
```

```
require(activeInvestAmount + passiveInvestAmount == 0, "invested!");
```

CVF-117. INFO

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

Recommendation The type of this argument should be “IStrategy”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address _strategy,
```

CVF-118. FIXED

- **Category** Suboptimal
- **Source** SingleAssetVault.sol

Recommendation This could be simplified as: if (_isPassive) { ... } else { ... }

```
if (_isPassive == false) {
```

```
} else {
```

CVF-119. FIXED

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

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

Recommendation Consider using the “mulDiv” function as described here: https://xn–2-mb.com/21/muldiv/index.html or some other approach, resistant to phantom overflows.

```
134     depositMultiplier: (_amount * MULTIPLIER_DENOMINATOR) /
          (totalFundAmount + totalPassivePoolAmount),
```

```
185 (redeemAmount * MULTIPLIER_DENOMINATOR) /
    pool.investAmount;
```

```
190   (pool.investAmount * MULTIPLIER_DENOMINATOR) /
      (totalFundAmount + totalPassivePoolAmount);
```

```
246     (remainAmount * pool.depositMultiplier) /
        MULTIPLIER_DENOMINATOR;
```

```
252     (userPoolDeposits[action.poolId] * pool.redemptionMultiplier) /
```

```
262   (remainAmount * totalPassivePoolAmount) /
      (totalFundAmount + totalPassivePoolAmount);
```

CVF-120. INFO

- **Category** Suboptimal
- **Source** SingleAssetVault.sol

Recommendation This logic is executed in both branches and should be placed after the conditional statement.

Client Comment *Current version is more readable, won’t fix.*

```
143 totalActivePoolAmount += _amount;
```

```
156 totalPassivePoolAmount += _amount;
```

CVF-123. FIXED

- **Category** Suboptimal
- **Source** SingleAssetVault.sol

Description The “actions.length” value is read from the storage on every loop iteration.

Recommendation Consider reading once before the loop.

```
while (depositIndex < deposits.length || currentActionId < actions.
    ↪ length) {
```

CVF-124. INFO

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

Description Action type “Redeem” is implicitly assumed here.

Recommendation Consider making this assumption explicit like: else if (action.actionType == ActionType.Redeem) { ... } else revert ();

Client Comment *Won’t fix.*

```
} else if (userPoolDeposits[action.poolId] > 0) {
```

CVF-125. INFO

- **Category** Documentation
- **Source** SAStrategyAllPairVault.sol

Description The semantics of the keys for these mappings is unclear.

Recommendation Consider documenting.

Client Comment *Style guide conflict, won’t fix.*

```
mapping(uint256 => bool) public vaultInvested;
mapping(uint256 => InvestParam) public investParams;
```

CVF-126. FIXED

- **Category** Readability
- **Source** SAStrategyAllPairVault.sol

Recommendation The “vaultId” parameter should be indexed.

```
event Invest(uint256 vaultId, OLib.Tranche tranche, uint256 amount);
event Redeem(uint256 vaultId, OLib.Tranche tranche, uint256 amount);
```

CVF-127. INFO

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

Recommendation The type of this argument should be “ISingleAssetVault”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address _singleAssetVault,
```

CVF-128. INFO

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

Recommendation The type of this argument should be “IPairVault”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address _vault,
```

CVF-129. INFO

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

Recommendation The type of this argument should be “IRegistry”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address _registry
```

CVF-130. INFO

- **Category** Suboptimal
- **Source** SAStrategyAllPairVault.sol

Description These checks are redundant. It is anyway possible to pass a dead address as an argument.

Client Comment *Won’t fix.*

```
require(_singleAssetVault != address(0), "Invalid asset vault");
require(_vault != address(0), "Invalid AllPairVault");
```

CVF-131. FIXED

- **Category** Procedural
- **Source** SAStrategyAllPairVault.sol

Recommendation This transfer should be postponed until all the checks are made.

```
asset.safeTransferFrom(msg.sender, address(this), amount);
```

CVF-132. FIXED

- **Category** Procedural
- **Source** SAStrategyAllPairVault.sol

Recommendation This check should be performed earlier, right after decoding “params”.

```
require(!vaultInvested[param.vaultId], "vault invested");
```

CVF-134. FIXED

- **Category** Flaw
- **Source** SAStrategyAllPairVault.sol

Description The state is updated after calling external contracts, which could make a reentrancy attack possible.

Recommendation Consider refactoring the code to do state updates before calling external contracts.

96
```
vaultInvested[investParam.vaultId] = false;
```

CVF-135. FIXED

- **Category** Flaw
- **Source** SAStrategyAllPairVault.sol

Description The “withdraw” function returns the withdrawn amount.

Recommendation Consider using a returned value instead of a calculated one.

103
```
// calculate withdrawn balance
amount = asset.balanceOf(address(this)) - amount;
```

CVF-136. FIXED

- **Category** Suboptimal
- **Source** SAStrategyAllPairVault.sol

Recommendation Consider doing this only when amount>0.

106
```
asset.safeTransfer(msg.sender, amount);

emit Redeem(investParam.vaultId, investParam.tranche, amount);
```

CVF-137. FIXED

- **Category** Procedural
- **Source** OndoLibrary.sol

Recommendation This commented out import should be removed.

8
```
//import "@openzeppelin/contracts/utils/Address.sol";
```

CVF-138. INFO

- **Category** Procedural
- **Source** OndoLibrary.sol

Description The library name ("OLib") is inconsistent with the file name ("OndoLibrary.sol"). This makes it harder to navigate through the code.

Recommendation Consider naming consistently.

Client Comment *Style guide conflict, won't fix.*

```
library OLib {
```

CVF-139. INFO

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

Recommendation The type of these fields should be "IERC20".

Client Comment *We have adopted in our team's solidity style guide to generally prefer the more generic 'address' datatype.*

```
address seniorAsset;
address juniorAsset;
```

CVF-140. INFO

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

Recommendation The type of this field should be "IStrategy".

Client Comment *We have adopted in our team's solidity style guide to generally prefer the more generic 'address' datatype.*

```
address strategy;
```

CVF-141. INFO

- **Category** Documentation
- **Source** OndoLibrary.sol

Description The number format of this field is unclear.

Recommendation Consider documenting.

Client Comment *Style guide conflict, won't fix.*

```
27 uint256 hurdleRate;
```

CVF-142. INFO

- **Category** Suboptimal
- **Source** OndoLibrary.sol

Description These two arrays seem to always have the same length. Using a single array of structs with two fields would be more efficient as the length will only be stored once.

Client Comment *Style guide conflict, won't fix.*

```
68 uint256[] userSums;
   uint256[] prefixSums;
```

CVF-143. INFO

- **Category** Documentation
- **Source** OndoLibrary.sol

Description The semantics of the returned values is unclear.

Recommendation Consider documenting.

Client Comment *Style guide conflict, won't fix.*

```
94 returns (uint256 userInvested, uint256 excess)
```

CVF-144. INFO

- **Category** Suboptimal
- **Source** OndoLibrary.sol

Recommendation It would be clear to say: return (0, 0);

Client Comment *Style guide conflict, won’t fix.*

100
```
return (userInvested, excess);
```

CVF-145. INFO

- **Category** Suboptimal
- **Source** OndoLibrary.sol

Recommendation It would be clearer to say: return (investor.userSums[length - 1], 0);

Client Comment *Style guide conflict, won’t fix.*

105
```
userInvested = investor.userSums[length - 1];
return (userInvested, excess);
```

CVF-146. INFO

- **Category** Suboptimal
- **Source** OndoLibrary.sol

Description In all these cases, “excess” actually equals to “investor.userSums[length - 1] - userInvested”.

Recommendation Consider calculating excess in one place after all the conditional operators.

Client Comment *Negligible savings, won’t fix.*

106
```
return (userInvested, excess);
```

112
```
excess = investor.userSums[length - 1] - userInvested;
```

123
```
  excess = investor.userSums[length - 1] - userInvested;
```

125
```
  excess = investor.userSums[length - 1] - userInvested;
```

CVF-147. INFO

- **Category** Suboptimal
- **Source** OndoLibrary.sol

Description The expression “prefixSum - depositAmount” is implicitly calculated twice.

Recommendation Consider refactoring like this to calculate it only once: uint256 prefix-SumBeforeDeposit = prefixSum - depositAmount; if (prefixSumBeforeDeposit < invested) { userInvested += invested - prefixSumBeforeDeposit;

Client Comment *Style guide conflict, won’t fix.*

```
if (prefixSum - depositAmount < invested) {
  userInvested += (depositAmount + invested - prefixSum);
```

CVF-148. INFO

- **Category** Procedural
- **Source** OndoLibrary.sol

Recommendation This library should be moved to a file named “OndoSaferERC20”.

Client Comment *Style guide conflict, won’t fix.*

```
library OndoSaferERC20 {
```

CVF-150. INFO

- **Category** Suboptimal
- **Source** OndoLibrary.sol

Description This need to be executed only in case the current allowance (obtained in the previous line) is not zero.

Client Comment *Won’t fix.*

```
token.safeApprove(spender, 0);
```

CVF-151. INFO

- **Category** Procedural
- **Source** ICurve_3.sol

Description In the curve code, this function is declared as “payable”, which means that it could accept ether, but this function is declared without the “payable” modifier, so it cannot accept ether: https://github.com/curvefi/deposit-nd-stake-zap/blob/2183cfa03d23b9a1e572d46332d73ad30b39845d/contracts/deposit_and_stake_zap.vy#L21 Probably not an issue.

Client Comment *Keeping functionaly as-is, won’t fix.*

```
function add_liquidity(uint256[3] calldata amounts, uint256
   ↪ min_mint_amount)
  external;
```

CVF-152. INFO

- **Category** Unclear behavior
- **Source** ICurve_3.sol

Description It is unclear which Curve functions these functions corresponds to.

Recommendation Consider providing references.

Client Comment *Style guide conflict, won’t fix.*

```
function remove_liquidity_one_coin(
  uint256 burn_amount,
  int128 i,
  uint256 mim_received
) external;
```

```
function coins(uint256) external view returns (address);
```

CVF-153. INFO

- **Category** Documentation
- **Source** ICurve_3.sol

Description The semantics of the returned value is unclear.

Recommendation Consider documenting.

Client Comment *Style guide conflict, won't fix.*

```
function coins(uint256) external view returns (address);
```

CVF-154. INFO

- **Category** Documentation
- **Source** IBaseRewardPool.sol

Description The semantics of the arguments are the returned values is unclear.

Recommendation Consider giving them descriptive names and/or adding documentation comments.

Client Comment *Style guide conflict, won't fix.*

```
function getReward() external returns (bool);
```

```
function balanceOf(address) external view returns (uint256);
```

```
function withdrawAndUnwrap(uint256, bool) external returns (bool);
```

CVF-155. INFO

- **Category** Documentation
- **Source** IConvexBooster.sol

Description The semantics of the arguments and returned values is unclear.

Recommendation Consider giving them descriptive names and/or adding documentation comments.

Client Comment *Style guide conflict, won't fix.*

```
function deposit(
  uint256,
  uint256,
  bool
) external returns (bool);
```

```
function withdraw(uint256, uint256) external returns (bool);
```

CVF-156. INFO

- **Category** Documentation
- **Source** IConvexBooster.sol

Description It is unclear what Convex functions these function correspond to.

Recommendation Consider documenting or using Convex interfaces directly from Convex repository.

Client Comment *Style guide conflict, won't fix.*

```
function deposit(
```

```
function withdraw(uint256, uint256) external returns (bool);
```

ABDK

CVF-157. FIXED

- **Category** Procedural
- **Source** ICurve_2.sol

Description In the curve code, this function is declared as "payable", which means that it could accept ether, but his function is declared without the "payable" modifier, so it cannot accept ether: https://github.com/curvefi/deposit-and-take-zap/blob/2183cfa03d23b9a1e572d46332d73ad30b39845d/contracts/deposit_and_stake_zap.vy#L18 Probably not an issue.

```
function add_liquidity(uint256[2] calldata amounts, uint256
   ↪ min_mint_amount)
  external;
```

CVF-158. INFO

- **Category** Unclear behavior
- **Source** ICurve_2.sol

Description It is unclear which Curve functions these functions corresponds to.

Recommendation Consider providing references.

Client Comment *Style guide conflict, won't fix.*

```
function remove_liquidity_one_coin(
  uint256 burn_amount,
  int128 i,
  uint256 mim_received
) external returns (uint256);
```

```
function coins(uint256) external view returns (address);
```

CVF-159. INFO

- **Category** Documentation
- **Source** ICurve_2.sol

Description The semantics of the returned value is unclear.

Recommendation Consider documenting.

Client Comment *Style guide conflict, won't fix.*

```
) external returns (uint256);
```

```
function coins(uint256) external view returns (address);
```

CVF-160. INFO

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

Recommendation The type of this field should be "IStrategy".

Client Comment *We have adopted in our team's solidity style guide to generally prefer the more generic 'address' datatype.*

```
address strategy;
```

CVF-161. FIXED

- **Category** Suboptimal
- **Source** ISAStrategy.sol

Description These functions should emit some events and these events should be declared in this interface.

```
function invest(
```

```
function redeem(uint256 poolId, bytes memory data)
```

CVF-162. INFO

- **Category** Documentation
- **Source** ISAStrategy.sol

Description The semantics of the returned values is unclear.

Recommendation Consider giving them descriptive names and/or adding a documentation comment.

Client Comment *Style guide conflict, won't fix.*

```
returns (bool, uint256);
```

CVF-163. FIXED

- **Category** Procedural
- **Source** IRollover.sol

Description This file is imported twice.

Recommendation Remove one import.

```
import "contracts/interfaces/ITrancheToken.sol";
```

```
import "contracts/interfaces/ITrancheToken.sol";
```

CVF-164. INFO

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

Recommendation Events are usually named via nouns, such as "Rollover", "Vault", "Migration", etc.

Client Comment *Style guide conflict, won't fix.*

```
event CreatedRollover(
```

```
event AddedVault(uint256 indexed rolloverId, uint256 indexed vaultId
    ↪ );
```

```
event MigratedRollover(
```

```
event Withdrew(
```

```
event Deposited(
```

```
event Claimed(
```

CVF-165. INFO

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

Recommendation The type of these parameters should be "IERC20".

Client Comment *We have adopted in our team's solidity style guide to generally prefer the more generic 'address' datatype.*

```
address seniorAsset,
address juniorAsset,
address seniorToken,
address juniorToken
```

CVF-166. INFO

- **Category** Suboptimal
- **Source** IRollover.sol

Description It seems that the length of these arrays is always 2.

Recommendation Consider declaring as “IERC20[2]” and “ITrancheToken[2]”.

Client Comment *Won’t fix.*

76
```
IERC20[] assets;
ITrancheToken[] rolloverTokens;
```

CVF-167. INFO

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

Recommendation These field could be replaced with a single field of type “TrancheRoundView”.

Client Comment *Won’t fix.*

82
```
uint256 deposited;
uint256 invested; // Total, if any, actually invested
uint256 redeemed; // After Vault is done, total tokens redeemed for
   ↪ LP
uint256 shares;
uint256 newDeposited;
uint256 newInvested;
```

CVF-168. INFO

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

Description The argument name “wad” is misleading. This argument is actually the amount to be withdrawn.

Client Comment *Won’t fix.*

9
```
function withdraw(uint256 wad) external;
```

CVF-169. FIXED

- **Category** Suboptimal
- **Source** IRegistry.sol

Description These functions should emit some events and these events should be declared in this interface.

```
function pause() external;
```

```
function unpause() external;
```

```
function enableFeatureFlag(bytes32 _featureFlag) external;
```

```
function disableFeatureFlag(bytes32 _featureFlag) external;
```

```
function deleteFeatureFlag(bytes32 _featureFlag) external;
```

CVF-170. FIXED

- **Category** Documentation
- **Source** IRegistry.sol

Description It is unclear how these two function do differ.

Recommendation Consider documenting.

```
function disableFeatureFlag(bytes32 _featureFlag) external;
```

```
function deleteFeatureFlag(bytes32 _featureFlag) external;
```

CVF-171. INFO

- **Category** Suboptimal
- **Source** IRegistry.sol

Description This function is an alias for the “hasRole” function declared in the “IAccess-Control” interface.

Recommendation Consider removing this function and using “hasRole” instead.

Client Comment *No difference, won’t fix.*

```
function authorized(bytes32 _role, address _account)
  external
  view
  returns (bool);
```

CVF-172. INFO

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

Description Despite the name, this function returns information about a single vault.

Recommendation Consider renaming to “vault” or “getVault”.

Client Comment *Style guide conflict, won’t fix.*

```
function vaults(uint256 vaultId)
```

CVF-173. INFO

- **Category** Suboptimal
- **Source** IStrategy.sol

Description This function should return an instance of the “Vault” structure.

Client Comment *Overriden by vault struct, won’t fix.*

```
returns (
  IPairVault origin,
  IERC20 pool,
  IERC20 senior,
  IERC20 junior,
  uint256 shares,
  uint256 seniorExcess,
  uint256 juniorExcess
);
```

CVF-174. FIXED

- **Category** Documentation
- **Source** IStrategy.sol

Description The semantics of this function is unclear.

Recommendation Consider documenting.

```
function addLp(uint256 _vaultId, uint256 _lpTokens) external;
```

```
function removeLp(
```

```
function withdrawExcess(
```

CVF-175. FIXED

- **Category** Procedural
- **Source** IStrategy.sol

Description An LP amount argument is named “_lpTlokens” in the former case and “_shared” in the latter case.

Recommendation Consider using consistent naming.

```
function addLp(uint256 _vaultId, uint256 _lpTokens) external;
```

```
  uint256 _shares,
```

CVF-176. FIXED

- **Category** Documentation
- **Source** IStrategy.sol

Description The semantics of the returned values is unclear.

Recommendation Consider documenting.

```
  returns (IERC20, uint256);
```

```
) external returns (uint256, uint256);
```

CVF-177. FIXED

- **Category** Documentation
- **Source** IStrategy.sol

Description The semantics of these amounts is unclear.

Recommendation Consider documenting.

```
uint256 _totalSenior,
uint256 _totalJunior,
uint256 _extraSenior,
uint256 _extraJunior,
uint256 _seniorMinOut,
uint256 _juniorMinOut
```

```
uint256 _seniorExpected,
uint256 _seniorMinOut,
uint256 _juniorMinOut
```

CVF-178. FIXED

- **Category** Documentation
- **Source** IStrategy.sol

Description The semantics of this function and its return values is unclear.

Recommendation Consider documenting.

```
function sharesFromLp(uint256 vaultId, uint256 lpTokens)
  external
  view
  returns (
    uint256 shares,
    uint256 vaultShares,
    IERC20 pool
  );
```

```
function lpFromShares(uint256 vaultId, uint256 shares)
  external
  view
  returns (uint256 lpTokens, uint256 vaultShares);
```

CVF-179. FIXED

- **Category** Documentation
- **Source** IPairVault.sol

Description The semantics of this field is unclear and is not documented.

Recommendation Consider adding a comment.

```
address rollover;
```

CVF-180. FIXED

- **Category** Documentation
- **Source** IPairVault.sol

Description The number format of this field is unclear.

Recommendation Consider documenting.

```
uint256 hurdleRate; // Return offered to senior tranche
```

CVF-181. INFO

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

Recommendation The type of this field should probably be “IRollover”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address rollover;
```

CVF-182. FIXED

- **Category** Documentation
- **Source** IPairVault.sol

Description The semantics of these amounts is unclear.

Recommendation Consider documenting.

```
uint256 originalInvested;
uint256 totalInvested; // not literal 1:1, originalInvested +
   ↪ proportional lp from mid-term
uint256 received;
uint256 rolloverDeposited;
```

CVF-183. FIXED

- **Category** Documentation
- **Source** IPairVault.sol

Description This comment is confusing.

Recommendation Consider adding more details.

```
uint256 totalInvested; // not literal 1:1, originalInvested +
   ↪ proportional lp from mid-term
```

CVF-184. FIXED

- **Category** Documentation
- **Source** IPairVault.sol

Description The difference between depositing and investing is unclear.

Recommendation Consider documenting.

```
function deposit(
```

```
function depositETH(uint256 _vaultId, OLib.Tranche _tranche)
   ↪ external payable;
```

```
function depositLp(uint256 _vaultId, uint256 _amount)
```

```
function invest(
```

CVF-185. FIXED

- **Category** Documentation
- **Source** IPairVault.sol

Description The semantics of the returned values is unclear.

Recommendation Consider documenting.

```
) external returns (uint256, uint256);
```

```
) external returns (uint256, uint256);
```

```
  returns (uint256);
```

```
  returns (uint256);
```

```
  returns (uint256, uint256);
```

```
  returns (uint256, uint256);
```

```
  returns (uint256, uint256);
```

```
  returns (uint256, uint256);
```

CVF-186. FIXED

- **Category** Documentation
- **Source** IPairVault.sol

Description The difference between redeeming, withdrawing, and claiming is unclear.

Recommendation Consider documenting.

```
function redeem(
```

```
function withdraw(uint256 _vaultId, OLib.Tranche _tranche)
```

```
function withdrawETH(uint256 _vaultId, OLib.Tranche _tranche)
```

```
function withdrawLp(uint256 _vaultId, uint256 _amount)
```

```
function claim(uint256 _vaultId, OLib.Tranche _tranche)
```

```
function claimETH(uint256 _vaultId, OLib.Tranche _tranche)
```

CVF-187. FIXED

- **Category** Documentation
- **Source** IPairVault.sol

Description The semantics of these functions is unclear.

Recommendation Consider documenting.

```
function depositFromRollover(
```

```
function rolloverClaim(uint256 _vaultId, uint256 _rolloverId)
```

```
function vaultInvestor(uint256 _vaultId, OLib.Tranche _tranche)
```

```
function seniorExpected(uint256 _vaultId) external view returns (uint256);
```

CVF-188. INFO

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

Recommendation The type of this argument should probably be “IRollover”.

Client Comment *We have adopted in our team’s solidity style guide to generally prefer the more generic ‘address‘ datatype.*

```
address _rollover,
```

CVF-189. FIXED

- **Category** Procedural
- **Source** IPairVault.sol

Recommendation This commented out function should be removed.

```
// function canTransition(uint256 _vaultId, OLib.State _state)
//   external
//   view
//   returns (bool);
```

CVF-190. FIXED

- **Category** Documentation
- **Source** ITrancheToken.sol

Description The semantics of this function is unclear.

Recommendation Consider documenting.

Client Comment *Function removed.*

```
function destroy(address payable _receiver) external;
```

CVF-191. INFO

- **Category** Procedural
- **Source** IMultiex.sol

Recommendation Consider naming the function in camelCase, i.e. “multiexCall”.

Client Comment *Style guide conflict, won’t fix.*

```
function multiexcall(Call[] calldata calls)
```

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