Report

v. 1.0

Customer

1inch

Smart Contract Audit

Limit Order Settlement

28th February 2023

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	27.02.23	A. Zveryanskaya	Initial Draft
0.2	27.02.23	A. Zveryanskaya	Minor revision
1.0	28.02.23	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.

The 1inch Network unites decentralized protocols whose synergy enables the most lucrative, fastest and protected operations in the DeFi space.

3 Project scope

We were asked to review:

- Original Code
- Code with Fixes 1
- Code with Fixes 2
- Code with Fixes 3

Files:

delegating/		
BasicDelegationPod.sol	DelegatedShare.sol	Rewardable DelegationPod.sol
delegating/interfaces/		
IDelegatedShare.sol	IDelegationPod.sol	
erc20-pods/		
ERC20Pods.sol	Pod.sol	
erc20-pods/interfaces/		
IERC20Pods.sol	IPod.sol	
limit-order-settlement/		
BasicDelegationPod WithVotingPower.sol	FeeBank.sol	FeeBankCharger.sol
RewardableDelegation PodWithVotingPower.sol	Settlement.sol	St1inch.sol
WhitelistRegistry.sol		
limit-order-settlement/helpers		
VotingPower Calculator.sol		
limit-order-settlement/interfaces		
IFeeBank.sol	IFeeBankCharger.sol	IResolver.sol
ISettlement.sol	IVotable.sol	IWhitelistRegistry.sol
limit-order-settlement/libraries		
DynamicSuffix.sol	OrderSaltParser.sol	
EXTRA LINES REVIEW 327 lines		

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.
- **Minor issues** contain code style, best practices and other recommendations.

5 Our findings

We found 5 major, and a few less important issues. All identified Major issues have been fixed or otherwise addressed in collaboration with the client.

	Active	Fixed
Major	3	2
Moderate	10	5
Minor	36	29

Fixed 36 out of 85 issues

6 Major Issues

CVF-1. INFO

- **Category** Suboptimal
- **Source** WhitelistRegistry.sol

Recommendation This function has complexity O(n^2) and could be optimized to O(n) complexity by implementing the quickselect algorithm: https://en.wikipedia.org/wiki/Quickselect

Client Comment *Instead we opted for optimstic 2-step offchain approach.*

```
125 function _shrinkPoorest(AddressSet.Data storage set, uint256 size)
       ↪ private {
```

CVF-2. INFO

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

Description Overflow is possible here.

Client Comment *Won't fix.*

```
131 mstore(add(add(ptr, interactionLengthOffset), 4), add(
       ↪ interactionLength, suffixLength))
```

```
134 let offset := add(add(ptr, interactionOffset), interactionLength)
    mstore(add(offset, 0x04), totalFee)
    mstore(add(offset, 0x24), resolver)
    mstore(add(offset, 0x44), orderToken)
    mstore(add(offset, 0x64), orderSalt)
```

CVF-3. FIXED

- **Category** Suboptimal
- **Source** St1inch.sol

Description This function should emit some event.

```
51 function setEmergencyExit(bool _emergencyExit) external onlyOwner {
```

CVF-4. FIXED

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

Description There is no range check for ”origin” argument.

Recommendation Consider checking that it is in the past.

```
38 constructor(uint256 expBase, uint256 origin) {
```

CVF-5. INFO

- **Category** Suboptimal
- **Source** OrderSaltParser.sol

Recommendation Applying a mask after shift would make bytecode smaller.

Client Comment *Won't fix. It makes bytecode smaller but loses the beauty of visible masks.*

```
19 return (salt & _TIME_START_MASK) >> _TIME_START_SHIFT;
23 return (salt & _DURATION_MASK) >> _DURATION_SHIFT;
27 return (salt & _INITIAL_RATE_BUMP_MASK) >> _INITIAL_RATE_BUMP_SHIFT;
31 return (salt & _FEE_MASK) >> _FEE_SHIFT;
35 return salt & _SALT_MASK;
```

7 Moderate Issues

CVF-6. INFO

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

Description Overflow is possible here.

Recommendation Consider using checked arithmetics or the ”muldiv” function..

Client Comment *Won’t fix. Types are small and overflow is not possible.*

```
101 ? _BASE_POINTS + (initialRateBump * (duration - timePassed)) /
      ↪ duration
```

CVF-7. INFO

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

Description Overflow is possible here.

Recommendation Consider using checked arithmetics.

Client Comment *Won’t fix. Types are small and overflow is not possible.*

```
104 return _BASE_POINTS + initialRateBump;
```

CVF-8. INFO

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

Description There is no length check for ”data” to guarantee that these operations don’t read beyond the end of ”data”.

Recommendation Consider adding appropriate check.

Client Comment *Won’t fix. If there is not enough calldata for order, it will later fail on fillOrderTo.*

```
113 let orderOffset := add(data.offset, calldataload(data.offset))
    orderSalt := calldataload(orderOffset)
    orderToken := calldataload(add(orderOffset, 0x40))
```

```
123 let interactionLengthOffset := calldataload(add(data.offset, 0x40))
```

```
125 let interactionLength := calldataload(add(data.offset,
       ↪ interactionLengthOffset))
```

CVF-9. INFO

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

Description Overflow is possible here.

Recommendation Consider using checked arithmetics.

Client Comment *Won’t fix.*

```
115 orderToken := calldataload(add(orderOffset, 0x40))
```

```
124 let interactionOffset := add(interactionLengthOffset, 0x20)
    let interactionLength := calldataload(add(data.offset,
       ↪ interactionLengthOffset))
```

CVF-10. FIXED

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

Description There is no length check for ”cd” to ensure that ”cd” is at least 32 bytes long.

Recommendation Consider adding such check.

```
150 target := shr(96, calldataload(cd.offset))
```

CVF-11. FIXED

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

Description Underflow is possible here.

```
152 data.length := sub(cd.length, 20)
```

CVF-12. INFO

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

Description Underflow is possible here.

Client Comment *Won't fix. Suffix is constructed by the same contract so it will not overflow.*

```
159 suffix := sub(add(cd.offset, cd.length), suffixSize)
```

CVF-13. FIXED

- **Category** Unclear behavior
- **Source** St1inch.sol

Description This call may return incorrect value in case timestamp is less than the contract creation time.

Recommendation Consider either reverting or returning a correct value in such a case.

```
78 return _votingPowerAt(balance, timestamp);
```

CVF-14. INFO

- **Category** Suboptimal
- **Source** BasicDelegationPod.sol

Description This is executed even if ”to” is a zero address.

Recommendation Consider doing nothing or reverting in such a case.

Client Comment *Won't fix. This is an impossible scenario of transferring from zero address to zero address.*

```
34 _mint(delegated[to], amount);
```

CVF-15. INFO

- **Category** Suboptimal
- **Source** BasicDelegationPod.sol

Description This is executed even if ”from” is a zero address.

Recommendation Consider doing nothing or reverting in such a case.

Client Comment *Won't fix. This is an impossible scenario of transferring from zero address to zero address.*

```
39 _burn(delegated[from], amount);
```

CVF-16. FIXED

- **Category** Procedural
- **Source** BasicDelegationPod.sol

Recommendation Once the ”approve” function is disabled, the ”increaseAllowance” and ”decreaseAllowance” functions should also be disabled.

```
78 function approve(address /* spender */, uint256 /* amount */) public
   ↪  virtual override(IERC20, ERC20) returns (bool) {
```

CVF-17. FIXED

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

Description Underflow is possible here.

```
74  uint256 t = timestamp - _origin;
```

```
172 uint256 t = timestamp - _origin;
```

CVF-18. INFO

- **Category** Suboptimal
- **Source** VotingPowerCalculator.sol

Description Bits of "t" higher than 0×20000000 are silently ignored.

Recommendation Consider adding an explicit "require" statement to ensure that t doesn't exceed 0×3FFFFFFF. Alternatively, consider calculating missing exponents on the fly.

Client Comment *t can exceed 0×3FFFFFFF but there is no need to compute exponents further because they become zero.*

```
74  uint256 t = timestamp - _origin;
```

```
172 uint256 t = timestamp - _origin;
```

CVF-19. INFO

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

Description Overflow is possible here.

Client Comment *Won't fix.*

```
77  votingPower = (votingPower * _expTable0) / 1e18;
80  votingPower = (votingPower * _expTable1) / 1e18;
83  votingPower = (votingPower * _expTable2) / 1e18;
86  votingPower = (votingPower * _expTable3) / 1e18;
89  votingPower = (votingPower * _expTable4) / 1e18;
92  votingPower = (votingPower * _expTable5) / 1e18;
95  votingPower = (votingPower * _expTable6) / 1e18;
98  votingPower = (votingPower * _expTable7) / 1e18;
101 votingPower = (votingPower * _expTable8) / 1e18;
104 votingPower = (votingPower * _expTable9) / 1e18;
107 votingPower = (votingPower * _expTable10) / 1e18;
110 votingPower = (votingPower * _expTable11) / 1e18;
113 votingPower = (votingPower * _expTable12) / 1e18;
116 votingPower = (votingPower * _expTable13) / 1e18;
119 votingPower = (votingPower * _expTable14) / 1e18;
122 votingPower = (votingPower * _expTable15) / 1e18;
```

(125, 128, 131, 134, 137, 140, 143, 146, 149, 152, 155, 158, 161,164, 175, 178, 181, 184, 187, 190, 193, 196, 199, 202, 205, 208, 211, 214, 217, 220, 223, 226, 229, 232, 235, 238, 241, 244, 247, 250, 253, 256, 259, 262)

CVF-20. INFO

- **Category** Suboptimal
- **Source** ERC20Pods.sol

Recommendation The function now has O(n^2) complexity that could be optimized to O(n ln n) by sorting there two arrays.

Client Comment *Won't fix as we expect those arrays to be small.*

```
127 address[] memory a = _pods[from].items.get();
    address[] memory b = _pods[to].items.get();
```

8 Minor Issues

CVF-21. INFO

- **Category** Procedural
- **Source** WhitelistRegistry.sol

Description We didn't review this file.

```
6 import "@1inch/solidity-utils/contracts/libraries/AddressSet.sol";
```

CVF-22. INFO

- **Category** Suboptimal
- **Source** WhitelistRegistry.sol

Description These errors could be made not useful by adding parameters to them.

Client Comment *Won't fix.*

```
18 error BalanceLessThanThreshold();
   error NotEnoughBalance();
20 error AlreadyRegistered();
   error NotWhitelisted();
```

CVF-23. INFO

- **Category** Documentation
- **Source** WhitelistRegistry.sol

Description The semantics of this mapping is unclear.

Recommendation Consider documenting.

Client Comment *We'll fix this later.*

```
32 mapping(address => address) public promotion;
```

CVF-24. INFO

- **Category** Documentation
- **Source** WhitelistRegistry.sol

Description The semantics of keys in this mapping is unclear.

Recommendation Consider documenting.

Client Comment *This mapping was removed.*

```
37 mapping(address => uint256) private _promotingsCount;
```

CVF-25. INFO

- **Category** Suboptimal
- **Source** WhitelistRegistry.sol

Description These events are emitted even if nothing actually changed.

Client Comment *Won't fix.*

```
161 emit SetResolverThreshold(resolverThreshold_);
```

```
166 emit SetWhitelistLimit(whitelistLimit_);
```

CVF-26. FIXED

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

Recommendation The type of this constant should be "bytes1".

```
24 bytes32 private constant _FINALIZE_INTERACTION = bytes1(0x01);
```

CVF-27. INFO

- **Category** Readability
- **Source** Settlement.sol

Recommendation This value could be rendered as ”0.1e4” for readability.

Client Comment *Won’t fix.*

```
27 uint256 private constant _DEFAULT_INITIAL_RATE_BUMP = 1000; // 10%
```

CVF-28. INFO

- **Category** Documentation
- **Source** Settlement.sol

Description The semantics of the returned value is unclear.

Recommendation Consider documenting.

Client Comment *We’ll fix this later.*

```
64 ) external onlyThis(taker) onlyLimitOrderProtocol returns (uint256
     ↪ result) {
```

CVF-29. INFO

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

Description Phantom overflow is possible here.

Recommendation Consider using the ”muldiv” function.

Client Comment *Won’t fix.*

```
79 result = (takingAmount * _calculateRateBump(suffix.salt)) /
     ↪ _BASE_POINTS;
```

CVF-30. FIXED

- **Category** Readability
- **Source** Settlement.sol

Recommendation Brackets around the multiplication are redundant.

```
101 ? _BASE_POINTS + (initialRateBump * (duration - timePassed)) /
      ↪ duration
```

CVF-31. INFO

- **Category** Procedural
- **Source** St1inch.sol

Description Specifying a particular compiler version makes it harder to migrate to newer versions.

Recommendation Consider specifying as "^0.8.0". Also relevant for: RewardableDelegationPodWithVotingPower.sol, FeeBankCharger.sol, FeeBank.sol, VotingPowerCalculator.sol, DynamicSuffix.sol, OrderSaltParser.sol, BasicDelegationPodWithVotingPower.sol, IWhitelistRegistry.sol, IVotable.sol, ISettlement.sol, IResolver.sol, IFeeBank.sol, IFeeBankCharger.sol.

Client Comment *Won't fix.*

```
3 pragma solidity 0.8.17;
```

CVF-32. INFO

- **Category** Suboptimal
- **Source** St1inch.sol

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

Client Comment *Won't fix.*

```
18 error LockTimeMoreMaxLock();
   error LockTimeLessMinLock();
20 error ChangeAmountAndUnlockTimeForExistingAccount();
   error UnlockTimeWasNotCome();
```

CVF-33. FIXED

- **Category** Suboptimal
- **Source** St1inch.sol

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

```
29 mapping(address => uint256) private _unlockTime;
30 mapping(address => uint256) private _deposits;
```

CVF-34. FIXED

- **Category** Documentation
- **Source** St1inch.sol

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

Recommendation Consider documenting.

```
29 mapping(address => uint256) private _unlockTime;
30 mapping(address => uint256) private _deposits;
```

CVF-35. FIXED

- **Category** Suboptimal
- **Source** St1inch.sol

Description There is no range check for this arguments.

Recommendation Consider adding an appropriate check.

```
40 uint256 _expBase,
```

CVF-36. INFO

- **Category** Suboptimal
- **Source** St1inch.sol

Description These functions should emit some deposit-specific events.

Client Comment *Won't fix. They emit Transfer in _mint.*

```
81  function deposit(uint256 amount, uint256 duration) external {
```

```
85  function depositWithPermit(
```

```
94  function depositFor(
```

```
101 function depositForWithPermit(
```

```
110 function increaseLockDuration(uint256 duration) external {
```

```
114 function increaseAmount(uint256 amount) external {
```

CVF-37. FIXED

- **Category** Suboptimal
- **Source** St1inch.sol

Description The value "deposits[amount]" is read from the storage twice times.

Recommendation Consider reading once and reusing.

```
124 if (_deposits[account] > 0 && amount > 0 && duration > 0) revert
    ↪ ChangeAmountAndUnlockTimeForExistingAccount();
```

```
128     _deposits[account] += amount;
```

```
138 _mint(account, _balanceAt(_deposits[account], lockedTill) /
    ↪ _VOTING_POWER_DIVIDER - balanceOf(account));
```

CVF-38. INFO

- **Category** Suboptimal
- **Source** St1inch.sol

Description This functions should emit some withdraw-specific events.

Client Comment *Won't fix. They emit Transfer in _burn.*

```
143 function withdraw() external {
```

```
147 function withdrawTo(address to) public {
```

CVF-39. INFO

- **Category** Documentation
- **Source** RewardableDelegationPod-WithVotingPower.sol

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

Client Comment *Won't fix.*

```
14 {} // solhint-disable-line no-empty-blocks
```

CVF-40. INFO

- **Category** Suboptimal
- **Source** FeeBankCharger.sol

Description This contract is not self-contained, but rather supposed to be inherited by other contracts.

Recommendation Consider declaring it as abstract.

Client Comment *Won't fix.*

```
9 contract FeeBankCharger is IFeeBankCharger {
```

CVF-41. FIXED

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

Recommendation The type of this variable should be "IFeeBank".

```
13 address public immutable feeBank;
```

CVF-42. INFO

- **Category** Documentation
- **Source** FeeBankCharger.sol

Description The semantics of the returned value is unclear.

Recommendation Consider documenting.

Client Comment *We'll fix this later.*

```
25 function availableCredit(address account) external view returns (
   ↪ uint256) {
```

CVF-43. INFO

- **Category** Suboptimal
- **Source** FeeBankCharger.sol

Description These functions should emit some events.

Client Comment *Won't fix.*

```
29 function increaseAvailableCredit(address account, uint256 amount)
   ↪ external onlyFeeBank returns (uint256 allowance) {
```

```
35 function decreaseAvailableCredit(address account, uint256 amount)
   ↪ external onlyFeeBank returns (uint256 allowance) {
```

CVF-44. FIXED

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

Description The returned value is actually unnamed in the code. Naming it in a comment is confusing.

Recommendation Consider naming the returned values in the code.

```
32 * @return totalAvailableCredit The total sender's availableCredit
      ↪ after deposit.
```

```
42 * @return totalAvailableCredit The total account's availableCredit
      ↪ after deposit.
```

```
52 * @return totalAvailableCredit The total sender's availableCredit
      ↪ after deposit.
```

```
73 * @return totalAvailableCredit The total sender's availableCredit
      ↪ after withdrawal.
```

```
83 * @return totalAvailableCredit The total sender's availableCredit
      ↪ after withdrawal.
```

CVF-45. INFO

- **Category** Suboptimal
- **Source** FeeBank.sol

Description These functions should emit some events specific to their use cases, not only generic Transfer events.

Client Comment *Won't fix.*

```
34 function deposit(uint256 amount) external returns (uint256) {
```

```
44 function depositFor(address account, uint256 amount) external
   ↪ returns (uint256) {
```

```
54 function depositWithPermit(uint256 amount, bytes calldata permit)
   ↪ external returns (uint256) {
```

```
61 function depositForWithPermit(
```

```
75 function withdraw(uint256 amount) external returns (uint256) {
```

```
85 function withdrawTo(address account, uint256 amount) external
   ↪ returns (uint256) {
```

```
94 function gatherFees(address[] memory accounts) external onlyOwner
   ↪ returns (uint256 totalAccountFees) {
```

CVF-46. FIXED

- **Category** Suboptimal
- **Source** FeeBank.sol

Description The expression "account[i]" is calculated several times.

Recommendation Consider calculating once and reusing.

```
97 uint256 accountFee = _accountDeposits[accounts[i]] - _charger.
   ↪ availableCredit(accounts[i]);
_accountDeposits[accounts[i]] -= accountFee;
```

CVF-47. FIXED

- **Category** Suboptimal
- **Source** FeeBank.sol

Description The value "_accountDeposits[accounts[i]]" is is read from the storage twice.

Recommendation Consider reading once and reusing.

```
97 uint256 accountFee = _accountDeposits[accounts[i]] - _charger.
      ↪ availableCredit(accounts[i]);
   _accountDeposits[accounts[i]] -= accountFee;
```

CVF-48. FIXED

- **Category** Suboptimal
- **Source** FeeBank.sol

Description The value assigned here is actually the result of the "_charger.availableCredit" call above.

Recommendation Consider caching and reusing that value.

```
98 _accountDeposits[accounts[i]] -= accountFee;
```

CVF-49. FIXED

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

Recommendation The type of the "token" argument should be "IERC20Pods".

```
16 constructor(string memory name_, string memory symbol_, address
      ↪ token)
```

CVF-50. FIXED

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

Description Some argument names have the underscore ("_") suffix, while other don't.

Recommendation Consider using a consistent naming schema.

```
16 constructor(string memory name_, string memory symbol_, address
   ↪ token)
```

CVF-51. FIXED

- **Category** Readability
- **Source** BasicDelegationPod.sol

Recommendation Should be "else if" for readability.

```
38 if (to == address(0)) {
```

CVF-52. FIXED

- **Category** Readability
- **Source** BasicDelegationPod.sol

Description The code below looks like it is always execute, while it is executed only when neither "from" nor "to" is zero.

Recommendation Consider putting the rest of the function into an explicit "else" branch.

```
41 }
```

CVF-53. FIXED

- **Category** Suboptimal
- **Source** BasicDelegationPod.sol

Description This would emit two separate events making it harder to track balance transfers.

Recommendation Consider calling "_transfer" instead.

```
46      _burn(fromDelegatee, amount);
        _mint(toDelegatee, amount);
```

```
52  _burn(prevDelegatee, balance);
    _mint(delegatee, balance);
```

CVF-54. FIXED

- **Category** Suboptimal
- **Source** BasicDelegationPod.sol

Description These checks make the functions to do nothing in certain cases, which is error prone.

Recommendation Consider reverting on zero arguments and leaving the "do nothing" logic to be implemented by callers.

```
59  if (account != address(0)) {
```

```
65  if (account != address(0)) {
```

CVF-55. FIXED

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

Recommendation The value ”1e18” should be a named constant.

```
41  _expTable1 = (_expTable0 * _expTable0) / 1e18;
    _expTable2 = (_expTable1 * _expTable1) / 1e18;
    _expTable3 = (_expTable2 * _expTable2) / 1e18;
    _expTable4 = (_expTable3 * _expTable3) / 1e18;
    _expTable5 = (_expTable4 * _expTable4) / 1e18;
    _expTable6 = (_expTable5 * _expTable5) / 1e18;
    _expTable7 = (_expTable6 * _expTable6) / 1e18;
    _expTable8 = (_expTable7 * _expTable7) / 1e18;
    _expTable9 = (_expTable8 * _expTable8) / 1e18;
50  _expTable10 = (_expTable9 * _expTable9) / 1e18;
    _expTable11 = (_expTable10 * _expTable10) / 1e18;
    _expTable12 = (_expTable11 * _expTable11) / 1e18;
    _expTable13 = (_expTable12 * _expTable12) / 1e18;
    _expTable14 = (_expTable13 * _expTable13) / 1e18;
    _expTable15 = (_expTable14 * _expTable14) / 1e18;
    _expTable16 = (_expTable15 * _expTable15) / 1e18;
    _expTable17 = (_expTable16 * _expTable16) / 1e18;
    _expTable18 = (_expTable17 * _expTable17) / 1e18;
    _expTable19 = (_expTable18 * _expTable18) / 1e18;
60  _expTable20 = (_expTable19 * _expTable19) / 1e18;
    _expTable21 = (_expTable20 * _expTable20) / 1e18;
    _expTable22 = (_expTable21 * _expTable21) / 1e18;
    _expTable23 = (_expTable22 * _expTable22) / 1e18;
    _expTable24 = (_expTable23 * _expTable23) / 1e18;
    _expTable25 = (_expTable24 * _expTable24) / 1e18;
    _expTable26 = (_expTable25 * _expTable25) / 1e18;
    _expTable27 = (_expTable26 * _expTable26) / 1e18;
    _expTable28 = (_expTable27 * _expTable27) / 1e18;
    _expTable29 = (_expTable28 * _expTable28) / 1e18;
```

CVF-56. INFO

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

Description These conversions drop the higher bits of a value.

Recommendation Consider explicitly requiring the dropped bits to be zero.

Client Comment *Won't fix. The idea here is to pack data tightly and use unused address bits for flags etc.*

```
18 return address(uint160(self._resolver));
```

```
22 return IERC20(address(uint160(self._token)));
```

CVF-57. INFO

- **Category** Procedural
- **Source** ERC20Pods.sol

Description We didn't review this file.

```
6 import "@1inch/solidity-utils/contracts/libraries/AddressSet.sol";
```

CVF-58. INFO

- **Category** Suboptimal
- **Source** ERC20Pods.sol

Recommendation These errors could be made more useful by adding parameters to them, such as the pod address or reached limit.

Client Comment *Won't fix.*

```
15 error PodAlreadyAdded();
   error PodNotFound();
   error InvalidPodAddress();
   error PodsLimitReachedForAccount();
```

CVF-59. INFO

- **Category** Suboptimal
- **Source** ERC20Pods.sol

Description Sets are usually considered unordered, so accessing a set element by index looks weird.

Recommendation Consider either removing such opportunity or explaining the index semantics and who stable the indexes are (i.e. whether an index of an element may change when another element was added or removed.

Client Comment *Won't fix.*

```
40 return _pods[account].at(index);
```

CVF-60. INFO

- **Category** Readability
- **Source** ERC20Pods.sol

Recommendation Should be "else return" for readability.

Client Comment *Won't fix.*

```
51 return 0;
```

CVF-61. INFO

- **Category** Bad datatype
- **Source** ERC20Pods.sol

Recommendation The type of the "pods" arguments should be "IPod".

Client Comment *Won't fix.*

```
66 function _addPod(address account, address pod) internal virtual {
```

```
77 function _removePod(address account, address pod) internal virtual {
```

CVF-62. INFO

- **Category** Suboptimal
- **Source** ERC20Pods.sol

Recommendation Checking the length before adding would save gas on failed cases.

Client Comment *Won't fix. It changes error in case of the same pod addition.*

```
69 if (_pods[account].length() > podsLimit) revert
      ↪ PodsLimitReachedForAccount();
```

CVF-63. INFO

- **Category** Suboptimal
- **Source** ERC20Pods.sol

Description Using decremented index looks odd.

Recommendation Consider using the "arrow" descending "for" loop: for (uint256 i = items.length; i −→ 0;) { ... }

Client Comment *Won't fix.*

```
92     _updateBalances(items[i - 1], account, address(0), balance);
```

```
94 _pods[account].remove(items[i - 1]);
```

CVF-64. INFO

- **Category** Suboptimal
- **Source** ERC20Pods.sol

Recommendation It would be more efficient to check whether: gas() * 63 < _POD_CALL_GAS_LIMIT * 64, where the right part could be precomputed.

Client Comment *Won't fix. With immutables precompilation will not work.*

```
112 if lt(div(mul(gas(), 63), 64), _POD_CALL_GAS_LIMIT) {
```

CVF-65. INFO

- **Category** Suboptimal
- **Source** ERC20Pods.sol

Recommendation This check should be performed at the very beginning of the assembly block.

Client Comment *Won't fix. We tried to do it as close to the call as possible.*

```
112 if lt(div(mul(gas(), 63), 64), _POD_CALL_GAS_LIMIT) {
```

CVF-66. FIXED

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

Recommendation The type of the "token" argument should be "IERC20".

```
32 constructor(string memory name_, string memory symbol_, address
   ↪ token) BasicDelegationPod(name_, symbol_, token) {}
```

CVF-67. FIXED

- **Category** Suboptimal
- **Source** RewardableDelegationPod.sol

Description The value "defaultFarms[delegatee]" is read from the storage twice.

Recommendation Consider reading once and reusing.

```
38 if (defaultFarms[delegatee] != address(0)) {
       delegatedShare.addDefaultFarmIfNeeded(msg.sender, defaultFarms[
          ↪ delegatee]);
```

CVF-68. FIXED

- **Category** Suboptimal
- **Source** RewardableDelegationPod.sol

Recommendation The type conversion is redundant, as ”shareFarms” is already ”IDelegatedShare”.

```
68 registration[msg.sender] = IDelegatedShare(shareToken);
```

CVF-69. FIXED

- **Category** Suboptimal
- **Source** RewardableDelegationPod.sol

Description This function should emit some event.

```
84 function setDefaultFarm(address farm) external onlyRegistered {
```

CVF-70. FIXED

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

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

```
10 address public immutable token;
```

CVF-71. FIXED

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

Recommendation The type of the ”token_” argument should be ”IERC20”.

```
17 constructor(address token_) {
```

CVF-72. FIXED

- **Category** Documentation
- **Source** BasicDelegationPodWith-VotingPower.sol

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

Client Comment *This file was removed completely.*

```
14 {} // solhint-disable-line no-empty-blocks
```

CVF-73. INFO

- **Category** Procedural
- **Source** ISettlement.sol

Description We didn't review this file.

```
5 import "@1inch/limit-order-protocol-contract/contracts/interfaces/
    ↪ IInteractionNotificationReceiver.sol";
```

CVF-74. INFO

- **Category** Documentation
- **Source** ISettlement.sol

Description The semantics of this function is unclear.

Recommendation Consider adding a documentation comment.

Client Comment *We'll fix this later.*

```
9 function settleOrders(bytes calldata order) external;
```

CVF-75. INFO

- **Category** Documentation
- **Source** IResolver.sol

Description It is unclear how this function treats its argument and what it does with it.

Recommendation Consider adding a detailed documentation comment.

Client Comment *We'll fix this later.*

```
6 function resolveOrders(bytes calldata data) external;
```

CVF-76. FIXED

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

Description The semantics of the returned value is unclear.

Recommendation Consider either giving the returned value a descriptive name and/or adding a documentation comment.

```
6 function deposit(uint256 amount) external returns (uint256);
```

CVF-77. FIXED

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

Description The semantics of the returned values is unclear.

Recommendation Consider either giving descriptive names to the returned value and/or adding documentation comments.

```
7 function increaseAvailableCredit(address account, uint256 amount)
    ↪ external returns (uint256);
  function decreaseAvailableCredit(address account, uint256 amount)
    ↪ external returns (uint256);
```

CVF-78. INFO

- **Category** Bad datatype
- **Source** IERC20Pods.sol

Recommendation The type of the ”pod” arguments should be ”IPod”.

Client Comment *Won't fix.*

```
8  function hasPod(address account, address pod) external view returns(
       ↪ bool);

12 function podBalanceOf(address pod, address account) external view
       ↪ returns(uint256);

14 function addPod(address pod) external;
   function removePod(address pod) external;
```

CVF-79. INFO

- **Category** Bad datatype
- **Source** IERC20Pods.sol

Recommendation The returned type should be ”IPod”.

Client Comment *Won't fix.*

```
10 function podAt(address account, uint256 index) external view returns
       ↪ (address);
```

CVF-80. INFO

- **Category** Bad datatype
- **Source** IERC20Pods.sol

Recommendation The returned type should be ”IPod[]”.

Client Comment *Won't fix.*

```
11 function pods(address account) external view returns(address[]
       ↪ memory);
```

CVF-81. FIXED

- **Category** Suboptimal
- **Source** IERC20Pods.sol

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

```
14 function addPod(address pod) external;
   function removePod(address pod) external;
   function removeAllPods() external;
```

CVF-82. INFO

- **Category** Suboptimal
- **Source** IDelegationPod.sol

Recommendation The event parameters should be indexed.

Client Comment *Won't fix.*

```
9 event Delegate(address account, address delegatee);
```

CVF-83. FIXED

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

Recommendation A better name would be "delegatee".

```
11 function delegated(address account) external view returns(address);
```

CVF-84. INFO

- **Category** Procedural
- **Source** IDelegatedShare.sol

Description The function name suggests that this function may skip adding a default farm.

Recommendation Consider returning a boolean flag indicating whether a default farm was actually added.

Client Comment *Won't fix as it is used only in one place where this flag is not needed*

```
8 function addDefaultFarmIfNeeded(address account, address farm)
    ↪ external; // onlyOwner
```

CVF-85. INFO

- **Category** Suboptimal
- **Source** IDelegatedShare.sol

Description This function should emit some event and this event should be declared in this interface.

Client Comment *Won't fix. It emits PodAdded.*

```
8 function addDefaultFarmIfNeeded(address account, address farm)
    ↪ external; // onlyOwner
```

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