1inch EXCHANGE

1INCH FUSION MODE V1

SMART CONTRACTS SECURITY AUDIT REPORT

0XORIO

DECEMBER 28, 2022

CONTENTS

1 INTRO

1.1 DISCLAIMER

The audit makes no statements or warranties about the utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about the fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

1.2 ABOUT OXORIO

Oxorio is a young but rapidly growing audit and consulting company in the field of the blockchain industry, providing consulting and security audits for organizations from all over the world. Oxorio has participated in multiple blockchain projects during which smart contract systems were designed and deployed by the company.

Oxorio is the creator, maintainer, and major contributor of several blockchain projects and employs more than 5 blockchain specialists to analyze and develop smart contracts.

Our contacts:

- oxor.io
- ping@oxor.io
- Github
- Linkedin
- Twitter

1.3 SECURITY ASSESSMENT METHODOLOGY

A group of auditors is involved in the work on this audit. Each of them checks the provided source code independently of each other in accordance with the security assessment methodology described below:

1. Project architecture review

Study the source code manually to find errors and bugs.

2. Check the code for known vulnerabilities from the list

Conduct a verification process of the code against the constantly updated list of already known vulnerabilities maintained by the company.

3. Architecture and structure check of the security model

Study the project documentation and its comparison against the code including the study of the comments and other technical papers.

4. Result's cross-check by different auditors

Normally the research of the project is done by more than two auditors. This is followed by a step of mutual cross-check process of the audit results between different task performers.

5. Report consolidation

Consolidation of the audited report from multiple auditors.

6. Reaudit of new editions

After the provided review and fixes from the client, the found issues are being double-checked. The results are provided in the new version of the audit.

7. Final audit report publication

The final audit version is provided to the client and also published on the official website of the company.

1.4 FINDINGS CLASSIFICATION

1.4.1 Severity Level Reference

The following severity levels were assigned to the issues described in the report:

- **CRITICAL**: A bug leading to assets theft, locked fund access, or any other loss of funds due to transfer to unauthorized parties.
- **MAJOR**: A bug that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
- **WARNING**: A bug that can break the intended contract logic or expose it to DDoS attacks.
- **INFO**: Minor issue or recommendation reported to / acknowledged by the client's team.

1.4.2 Status Level Reference

Based on the feedback received from the client's team regarding the list of findings discovered by the contractor, the following statuses were assigned to the findings:

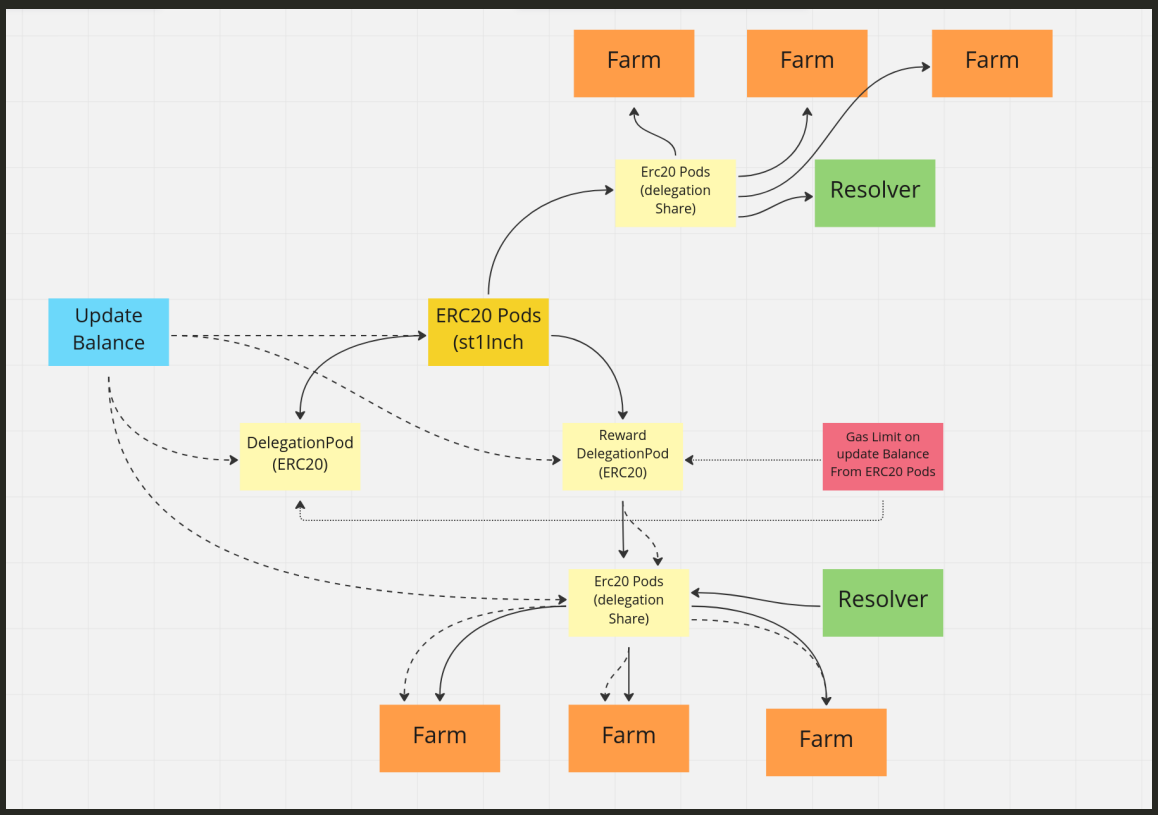
- **NEW**: Waiting for the project team's feedback.
- **FIXED**: Recommended fixes have been applied to the project code and the identified issue no longer affects the project's security.
- **ACKNOWLEDGED**: The project team is aware of this finding. Recommended fixes for this finding are planned to be made. This finding does not affect the overall security of the project.
- **NO ISSUE**: Finding does not affect the overall security of the project and does not violate the logic of its work.
- **DISMISSED**: The issue or recommendation was dismissed by the client.

1.5 PROJECT OVERVIEW

1inch is an exchange aggregator that scans decentralized exchanges to find the lowest cryptocurrency prices for traders, and is powered by its 1INCH utility and governance token.

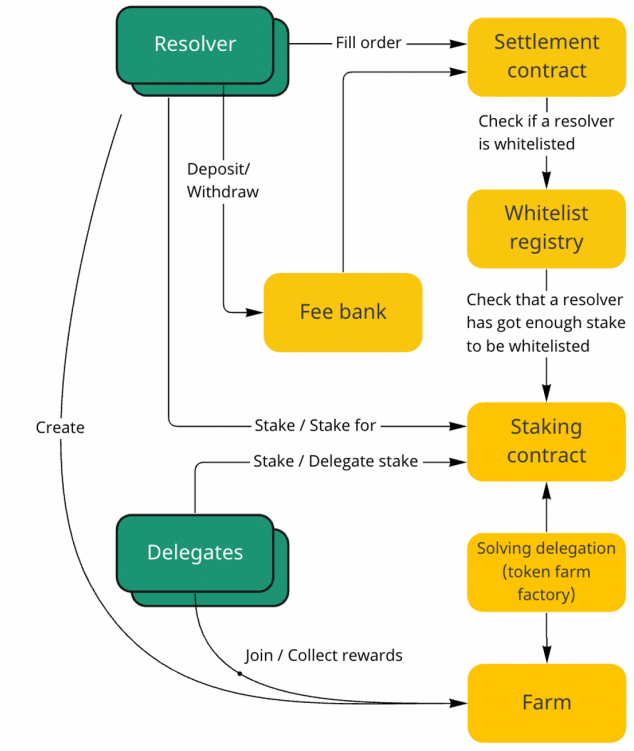
ERC20Pods contract is ERC20 extension enabling external smart contract based Pods to track balances of those users who opted-in to these Pods. For example, st1inch is ERC20Pods contract.
There are BasicDelegationPod and RewardableDelegationPod contracts in this scope.

Contracts interaction scheme:



Limit orders Settlement Contracts is part of new Fusion Mode V1.

Smart contracts



- Checks which resolvers are eligible for filling orders
- Executes the order filling on behalf of a resolver
- Implements dutch auction logic for orders (positive slippage)
- Transfers maker funds to resolver's implementation contract
- Collect fees from resolvers

- Manages whitelist

- Manages fee deposit/credit

- Holds resolvers' stakes
- Provides current staking weight
- Prolongs stake locks
- Receives stakes

- Provides staking for delegates
- Distributes staking rewards

- Provides additional rewards for delegates for staking

1.6 AUDIT SCOPE

- https://github.com/1inch/erc20-pods/
  Commit: f975e2eaec9714e66163c1826044f722660a14a2
- https://github.com/1inch/delegating
  Commit: 9a243d64422c41b0631617466deeb85dcd92e57c
- https://github.com/1inch/limit-order-settlement
  Commit: a6a811ac1eb76f545551e32c1756c52de03b9113

2 FINDINGS REPORT

OXORIO

2.1 CRITICAL

No critical issues found.

2.2 MAJOR

2.2.1 There is no validation of `podCallGasLimit_` in `ERC20Pods`

SEVERITY	MAJOR
STATUS	NO_ISSUE

Description

There is no validation of `podCallGasLimit_` in the constructor of the `ERC20Pods` contract. If the value of `podCallGasLimit_` ia zero or too low, work with `Pod` contracts will be blocked.

Recommendation

We recommend adding validation that `podCallGasLimit_` is greater than or equal to the `DEFAULT_CALL_GAS_LIMIT` value.

Update

1inch's response

Won't fix. The exact gas limit should be the design decision of the token creator.

2.2.2 There is no possibility to customize `_POD_CALL_GAS_LIMIT` in `DelegatedShare`

SEVERITY	MAJOR
STATUS	NO_ISSUE

Description

In the `DelegatedShare` contract, there is no possibility to customize `_POD_CALL_GAS_LIMIT`. `DelegationShare` is an extension of the `ERC20Pods` contract, and pods may require more gas depending on the purpose of use. If there is not enough gas, work with them will be blocked.

Recommendation

We recommend adding `_POD_CALL_GAS_LIMIT` value to the constructor in the `DelegatedShare` contract.

Update

1inch's response

Won't fix. This is by design. Some security aspects of the system a based on the assumption that those gas limits are constant.

2.2.3 There is no validation of `maxUserFarms` in `RewardableDelegationPod`

SEVERITY	**MAJOR**
STATUS	**ACKNOWLEDGED**

Description

In the `RewardableDelegationPod` contract there is no validation for `maxUserFarms`. If the value is high and the user adds all farms, it will cause the work with the pod to be blocked, since balance updates may require more than the parameter set in `ERC20Pods`.

For example, if `podCallGasLimit_` has the value of `300,000` and `DelegationShare` has the value of `100,000`, `maxUserFarms` should be less than 3. But there is no validation of that, and the resolver can set the value to 4.

Recommendation

We recommend adding validation of `maxUserFarms` depending on the value of `podCallGasLimit_` of the calling the `ERC20Pods` contract.

Update

1inch's response

This is a good suggestion, but it turns out to be difficult to calculate what specific assert value to do. While taking time to think.

2.2.4 There is no value for `feeReceiver` in the constructor in `St1Inch`

SEVERITY	MAJOR
STATUS	FIXED

Description

In the `St1Inch` contract, the value of `feeReceiver` is not set in the constructor and will be zero address after the deployment. In this case, depending on the implementation of the `ERC20` token, `loss` will be transferred to zero address when `earlyWithdraw` is called or `earlyWithdraw` will be reverted.

Recommendation

We recommend adding value setting for `feeReceiver` in the constructor of the contract `St1Inch`.

Update

1inch's response

Commit 43c6c08791cbad5a9927a630e7973443409a02d8.

2.2.5 There is no value for `maxLoss` in the constructor of the `St1Inch` contract

SEVERITY	MAJOR
STATUS	NO_ISSUE

Description

In the `St1Inch` contract the value of `maxLoss` is not set in the constructor and will be 0. This will cause `earlyWithdraw` to be reverted.

Recommendation

We recommend adding value setting for `maxLoss` in the constructor of the contract `St1Inch`.

Update

1inch's response

Won't fix. This is by design. Default option is to have early withdrawals disabled.

2.2.6 `podCallGasLimit` can not be customized in `ERC20Pods`

SEVERITY	**MAJOR**
STATUS	**NO_ISSUE**

Description

In the `ERC20Pods` contract `podCallGasLimit` is `immutable`. Gas cost of the opcodes can be changed to mitigate transaction spam attacks. If the cost of the opcodes is increased, deployed `ERC20Pods` will stop updating balances in the `_updateBalances` function.
Example of opcode gas cost changes: EIP-150, EIP-2929.

Recommendation

We recommend changing `podCallGasLimit` from `immutable` to a variable and implement the function `updatePodCallGasLimit` for changing the maximum amount of gas for calls.

Update

1inch's response

Won't fix. This is by design. Some security aspects of the system a based on the assumption that those gas limits are constant.

2.3.1 There is no validation that `Pod` has a valid token in `ERC20Pods`.

SEVERITY	**WARNING**
STATUS	**FIXED**

Description

In the contract `ERC20Pods` in the function `_addPod` there is no validation that a pod has a valid token. `ERC20Pods` token address should be set as a token for `Pod`.

Recommendation

We recommend adding validation that `Pod` has a valid token address before adding a pod.

```
if (Pod(pod).token() != address(this)) revert InvalidPodAddress();
```

Update

1inch's response

Commit 4098798e36f272804b47b7db1aa27aa3f545c9df.

2.3.2 The `promote` function is not authorized in `WhitelistRegistry`

SEVERITY	**WARNING**
STATUS	**NO_ISSUE**

Description

In the contract `WhitelistRegistry` in the function `promote` any user can promote `promotee` for `chainId`. This makes it vulnerable for ddos and storage spam.

Recommendation

We recommend adding logic that only the whitelisted addresses or a candidate to the whitelist can promote an address.

Update

1inch's response

Won't fix. One can be temporarily excluded from whitelist but still want to manage promotees.

2.3.3 There is no validation of promoted addresses in `WhitelistRegistry`

SEVERITY	**WARNING**
STATUS	**FIXED**

Description

There is no validation of already promoted addresses in the contract `WhitelistRegistry` in the `promote` function. This makes it possible for users to be able to `promote` the same address for `chainId` again.

Recommendation

We recommend adding validation that address has not been promoted for chain yet.

Update

1inch's response

Commit 2737211f262ceac9bbfbaf108bcacf9b6b6ae5c7

2.3.4 `feeReceiver` can be zero address in `St1Inch`

SEVERITY	WARNING
STATUS	FIXED

Description

In the function `setFeeReceiver` in the contract `St1inch`, it is possible to set `feeReceiver` as zero address. In this case, depending on the implementation of the `ERC20` token, `loss` will be transferred to zero address when `earlyWithdraw` is called, or `earlyWithdraw` will be reverted.

Recommendation

We recommend adding validation that `feeReceiver` is not zero address.

Update

1inch's response

Commit 39d568da8574f5891bbdf1962fe50f988969643a

2.3.5 Pod is not removed after withdrawal of all tokens from `St1Inch`

SEVERITY	WARNING
STATUS	NO_ISSUE

Description

In the `_withdrawTo` function in the `St1inch` contract, `Pod` is not removed after withdrawal of all tokens from `st1Inch`. This causes `Pod` to stay in user's Pods list. In the case of subsequent updates to `defaultFarm`, the user may have an error PodsLimitReachedForAccount when depositing to `St1Inch` again and entering a new `defaultFarm`. For example, if `podsLimit = 1`.

Recommendation

We recommend adding logic to remove `Pod` from account after withdrawal of all tokens from the `St1inch` contract.

Update

1inch's response

Won't fix. User might want to have those pods added even though their balance is temporarily zero.

2.3.6 Incorrect comparison in `WhitelistRegistry`

SEVERITY	**WARNING**
STATUS	**ACKNOWLEDGED**

Description

In the `WhitelistRegistry` contract the comparison of the `balances` in the `_shrinkPoorest` function is incorrect. When two whitelisted addresses have the same balance, this will result in the oldest address being removed as an address with the smallest balance.

Recommendation

We recommend changing the comparison to `balances[i] < balances[richestIndex]` instead. With this change the newest address will be removed after setting new limit of whitelisted addresses. We also recommend relying on the voting power rather than balances to determine the richest address.

Update

1inch's response

Won't fix. We'll replace this logic completely in future.

2.4.1 There is no validation of variables in `Settlement`

SEVERITY	INFO
STATUS	NO_ISSUE

Description

In the `Settlement` contract the `rateBumb` and `takingFee.ratio` variables are missing validation. This can cause loss of all funds from the `Settlement` contract by approving too much tokens to `_limitOrderProtocol` or by transferring `takingFee` to `takingFee.receiver()`. Loss of all the funds from the `Settlement` contract will lead to failed calls in the resolver contract, or it can be reverted in `fillOrderInteraction` call with a fee greater than 0.

Recommendation

We recommend adding off-chain and on-chain validation of the `rateBumb` and `takingFee.ratio` variables.

Update

1inch's response

Noted. Offchain validation is performed.

2.4.2 There is a possibility of overflow in `Settlement`

SEVERITY	INFO
STATUS	NO_ISSUE

Description

There is a possibility of overflow in the following cases: * Settlement.sol#L65 * Settlement.sol#L66 * Settlement.sol#L106

Tokens with big amount of decimals can lead to overflow and failed `fillOrderInteraction` calls. Large `rateBumb`, `salt.getFee()` and `takingFee.ratio` variables can also lead to overflow.

Recommendation

We recommend to simulate the call before the execution, or use `muldiv` to multiply elements safely.

Update

1inch's response

Noted. That's why the math there is checked. To make those orders invalid.

2.4.3 Orders with fee can be reverted in `Settlement`

SEVERITY	**INFO**
STATUS	**NO_ISSUE**

Description

In the `Settlement` contract the order can be set with fee. If the fee is greater than the amount of tokens on the balance of `Settlement`, the order will be reverted. The fees should already be on the contract address before the filling of the order since they are not substracted from the result value. It is obligatory to send tokens to the `Settlement` contract before the filling of the order with fee the same way as it is done in tests. The filling of the order with fee of the new token is very likely to fail.

Recommendation

We recommend substracting fees from the result value in `forceApprove` function, or adding a list of available tokens offchain or onchain and transfering tokens to the `Settlement` contract before the filling of the order with fee.

Update

1inch's response

Won't fix.

2.4.4 The `delegate` function does not revert when `prevDelegatee` equals new `delegatee` in `BasicDelegationPod`

SEVERITY	**INFO**
STATUS	**NO_ISSUE**

Description

In the `DelegationPod` contract user can limitlessly call `delegate` where `delegatee` variable equals `prevDelegatee`, this can be misleading for users looking for successful `delegate` transactions on the explorers with no state changes. It is also can be a problem while analyzing stats from the `DelegationPod` contract.

Recommendation

We recommend adding a revert condition and revert when the `prevDelegatee` equals `delegatee`.

Update

1inch's response

Won't fix.

2.4.5 The `withdraw` and `withdrawTo` functions do not revert when lock amount equals 0 in `St1inch`

SEVERITY	**INFO**
STATUS	**NO_ISSUE**

Description

In the `St1inch` contract user can limitlessly call the `withdraw` or `withdrawTo` functions where the `amount` of user is zero. This can be misleading for users looking for successful `withdraw` or `withdrawTo` transactions on the explorers with no state changes. It can also be a problem while analyzing statistics from the `St1inch` contract.

Recommendation

We recommend adding a revert condition and revert when the `amount` equals zero.

Update

1inch's response

Won't fix.

2.4.6 The `_DEFAULT_INITIAL_RATE_BUMP` and `_DEFAULT_DURATION` constants are not used in `Settlement`

SEVERITY	**INFO**
STATUS	**FIXED**

Description

The `_DEFAULT_INITIAL_RATE_BUMP` and `_DEFAULT_DURATION` constants in the `Settlement` contract are never used.

Recommendation

We recommend removing these constants to keep the codebase clean.

Update

1inch's response

Commit 9c9522c3b8c5eb37b6da6ee66b5fe28aa510fecd.

2.4.7 `rateBump` is not calculated in tests

SEVERITY	INFO
STATUS	ACKNOWLEDGED

Description

In the tests for `Settlement` contract the `rateBump` variable is not calculated. It results in wrong comparisons of balances of tokens on addresses.

Recommendation

We recommend calculating the `rateBump` variable and remove the `TODO:` comment.

Update

1inch's response

Noted. Will fix later.

2.4.8 Redundant allowance after resolve of the order in `Settlement`

SEVERITY	INFO
STATUS	NO_ISSUE

Description

In the `Settlement` contract after the `forceApprove` is executed, the order is being resolved. Some tokens, e.g. deflationary tokens, that where programmed poorly, can leave allowance on the `Settlement` address even after `transferFrom` with all allowed amount transferred.
It is also possible that the resolver hasn't used all available allowance.

Recommendation

We recommend adding a validation of the allowance after resolving the order and set it to zero if neccessary.

Update

1inch's response

Won't fix. forceApprove can handle those extra allowances.

2.4.9 Promoted address can not be removed in `WhitelistRegistry`

SEVERITY	**INFO**
STATUS	**NO_ISSUE**

Description

In the `WhitelistRegistry` contract promoted address can not be removed from `promotions` mapping. This can result in whitelisting of unwanted addresses passed there by mistake, or by whitelisting the address passed there long time ago.

Recommendation

We recommend implementing the `removePromotee` function to remove unwanted addresses from `promotions` mapping. We also recommend to remove all `promotee` when the whitelisted address is being removed from `_whitelist`.

Update

1inch's response

It is possible by calling promote(chainId, 0).

2.4.10 It is possible to call `_mint` with zero value in `St1Inch`

SEVERITY	INFO
STATUS	NO_ISSUE

Description

In the function `_deposit` in the contract `st1Inch` it is possible to do a call to `_mint` with the value of `balanceDiff` being zero. This causes the event `Transfer(address(0), account, amount)` to be emitted.

Recommendation

We recommend adding a validation that `balanceDiff != 0` before call `_mint`.

Update

1inch's response

Won't fix.

2.4.11 There is no validation of constructor parameters in `FeeBank`

SEVERITY	INFO
STATUS	FIXED

Description

In the constructor in the `FeeBank` contract there is no validation that `charger` and `inch` variables are not zero addresses.

Recommendation

We recommend adding a validation that `charger` and `inch` are not zero addresses.

Update

1inch's response

Commit fdecf80cce5f7ddc7b7d361b3d162b20e79df50b.

3 CONCLUSION

The following table contains the total number of issues that were found during audit:

Level	Amount
CRITICAL	0
MAJOR	6
WARNING	6
INFO	11
Total	**23**

THANK YOU FOR CHOOSING

OXORIO