OpenZeppelin | security

ERC-20 Pods and Delegating Pod Audit

March 01, 2023

This security assessment was prepared by OpenZeppelin.

Table of Contents

Summary

Type	DeFi
Timeline	From 2023-02-06 To 2023-02-13
Languages	Solidity

Total Issues	5 (3 resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	0 (0 resolved)
Low Severity Issues	1 (1 resolved)
Notes & Additional Information	4 (2 resolved)

Scope

We audited a diff of the 1inch/erc20-pods and 1inch/delegating repositories with `2b4545d435e159f08ddf80fd14e8f4efb665bcdb` being the base commit and `01535f1a84385fb78df27dd93f05e8d0b3d224a2` being the head for the former; and `9a243d64422c41b0631617466deeb85dcd92e57c` being the base commit and `da8872a034a2af16fa55f88f3953689a43678844` being the head for the latter.

In scope were the following contracts:

```
contracts
├── ERC20Pods.sol
├── Pod.sol
├── interfaces
│   ├── IERC20Pods.sol
│   └── IPod.sol
└── libs
    └── ReentrancyGuard.sol
```

```
contracts
├── DelegatedShare.sol
├── DelegationPod.sol
├── FarmingDelegationPod.sol
├── TokenizedDelegationPod.sol
└── interfaces
    ├── IDelegatedShare.sol
    ├── IDelegationPod.sol
    ├── IFarmingDelegationPod.sol
    └── ITokenizedDelegationPod.sol
```

Overview

The `erc20-pods` repository provides an extendable framework for ERC-20 tokens that allows users to opt into extra functionality carried out by third-party contracts known as pods. The `delegating` repository is an implementation of a pod that enables tracking of balance and its delegation.

ERC-20 Pods

Many protocols build parts of their code on top of ERC-20 balances. Some examples of balance-based systems are voting, staking and farming protocols. When implementing this style of system, developers commonly have issues tracking balances with every transfer, as ERC-20 transactions do not allow for code execution on transfers.

The `ERC20Pods` contract solves this problem by providing a generic hook into OpenZeppelin's ERC-20 contract during the `_afterTokenTransfer` call. ERC-20 users can register this hook by adding a pod to the ERC-20. Any pod added by a user only affects balance changes to that user's address; including transfers to and from, as well as minting and burning.

Since the `_afterTokenTransfer` hook provides code execution to any registered pods, it is very important to prevent pods from reverting transactions. The hook called has protections against reverts to ensure the underlying token does not behave differently when pods are enabled and used.

When balances change on a registered ERC-20 pods address, the `_updateBalances` function is called, pushing out balance updates to all registered pods. This allows pods to execute code after balance changes, resulting in accurate balance tracking with the ability to implement more complex logic.

Delegating

The `DelegationPod`, `TokenizedDelegationPod`, and `FarmingDelegationPod` contracts are implementations of an `erc20-pods` pod aimed at solving delegated balance tracking, as well as providing a simple mechanism to reward users for delegating their balances. This can easily be used for tracking credibility and voting power based on ERC-20 balances.

As the underlying ERC-20 balance changes, a `DelegationPod` will track delegated balances, increasing or decreasing the balance delegated by the user. If the contract is a `FarmingDelegationPod`, it is also possible for users to be rewarded for delegating to specific addresses.

It is important that proper accounting is carried out on the contracts. If delegated balances do not correctly represent underlying balances, users may have inaccurate delegation power and possibly receive the incorrect amount of rewards if `FarmingDelegationPod` was used.

Summary of Changes

The main change is splitting `RewardableDelegationPod` into two contracts: `FarmingDelegationPod` and `TokenizedDelegationPod`. `BasicDelegationPod` was renamed to `DelegationPod`. Several safety checks aimed for the users of the system were added, and small changes across the contracts were made.

Findings

Here we present our findings.

Low Severity

L-01 Default owner of the contract is set to the deployer

The `FarmingPod` contract is ownable and the owner is set to the deployer by default. However, the deployer and the actual owner are usually different contracts.

Consider transferring ownership during deployment for safety and consistency with other contracts.

Update: *Resolved. This is not an issue because all ownable contracts - except for* `FeeBank` *- are deployed by an EOA.* `FeeBank` *is an exception since it is deployed by the* `FeeBankCharger` *contract and transferring ownership this way consumes less gas.*

Notes & Additional Information

N-01 Missing docstrings

There are parts of the codebase that could benefit from having docstrings:

- Since `ERC20Pods` and other contracts are expected to be used by many developers, there should be warnings relating to possible failures due to implementation. For example, developers should be aware of how to properly configure the maximum number of pods and the amount of gas passed in order to prevent desync of chained `ERC20Pods` contracts. Such a desync may happen because contracts further down the chain might run out of gas and revert, while contracts earlier in the chain will ignore this revert.
- All the functions that are part of the contract's public API.

Consider documenting the aforementioned behavior and functions. When writing docstrings, consider following the Ethereum Natural Specification Format (NatSpec).

Update: *Acknowledged, not resolved. The 1inch team stated:*

> *Noted. We will fix the NatSpec docs in the future.*

N-02 Lack of public API for retrieving the owner of `DelegatedShare`

The `TokenizedDelegationPod` contract allows users to delegate their balance towards a registered delegatee. Users that would like to participate as delegatees register for an instance of the `DelegatedShare` contract.

Since the instance of the `DelegatedShare` contract was created by the `TokenizedDelegationPod`, the `_owner` will always be set to the `TokenizedDelegationPod`. Unfortunately, as the `_owner` variable is private, the only way to find the `TokenizedDelegationPod` that the `DelegatedShare` is associated with is by looking back at contract creation, or querying the memory slot associated with `_owner`.

In order to provide a convenient way to look up the `TokenizedDelegationPod` from the `DelegatedShare`, consider providing a method to retrieve the owner of the contract. Also, consider changing the `_owner` variable name to be more explicit, such as `_tokenizedDelegationPod`.

Update: *Resolved in pull request #16 at commit `e295252`.*

N-03 Lack of indexed parameters

On line 8 and line 9 of `IERC20Pods.sol`, both the `account` and `pod` parameters may benefit from indexing.

Consider indexing event parameters to improve the ability of off-chain services to search and filter for specific events.

Update: *Acknowledged, not resolved. The 1inch team stated:*

> *Will not fix. We will keep them not indexed for gas savings.*

N-04 Unused import

In `DelegatedShare.sol` the import `Ownable` is unused and could be removed.

Consider removing unused imports to improve the overall clarity and readability of the codebase.

Update: *Resolved at commit `f950437`.*

Conclusions

No critical or high-severity issues were found. Recommendations have been given to ensure production readiness of the code, improve quality, and minimize errors.

Appendix

Monitoring Recommendations

While audits help in identifying potential security risks, the 1inch team is encouraged to also incorporate automated monitoring of on-chain contract activity into their operations. Ongoing monitoring of deployed contracts helps in identifying potential threats and issues affecting the production environment.

As the `erc20-pods` and `delegating` repositories are frameworks for developers to implement into their protocols, any monitoring recommendations should be applied to the protocols that integrate these repositories.

One important aspect to monitor would be reverts in pods attached to the `erc20-pods` contract. Attached pods should not revert during normal operations. When they do revert, their balance tracking will not correctly reflect the true balances of the underlying ERC-20 pods contract. This could signal either malicious activity or oversight from pod developers and users.