Fjord Token Staking

Version 1.0

Shikhar Agarwal

September 16, 2024

Fjord Token Staking

Shikhar Agarwal

Sept 16, 2024

Prepared by: Shikhar Agarwal (https://www.codehawks.com/profile/clk3yh639002emf08ywok1hzf)

Lead Auditors: - Shikhar Agarwal

Table of Contents

About the Project

Fjord connects innovative projects and engaged backers through a community-focused platform, offering fair and transparent LBPs and token sale events. This repository is the Staking contract for

the Fjord ecosystem. Users who gets some ERC20 emitted by Fjord Foundry can stake them to get rewards.

See more contest details here - https://codehawks.cyfrin.io/c/2024-08-fjord

Disclaimer

As the sole auditor all efforts have been made to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. A security audit is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the Solidity implementation of the contracts.

Risk Classification

		Impact		
		High	Medium	Low
	High	H	H/M	M
Likelihood	Medium	H/M	M	M/L
	Low	M	M/L	L

We use the CodeHawks severity matrix to determine severity. See the documentation for more details.

Audit Details

Timeline

Aug 20th, 2024 → Aug 27th, 2024

Sponsor

Fjord

```
Commit Hash: 0312fa9dca29fa7ed9fc432fdcd05545b736575d
```

Scope

```
src/
#-- FjordAuction.sol
#-- FjordAuctionFactory.sol
#-- FjordPoints.sol
#-- FjordStaking.sol
#-- FjordToken.sol
#-- interfaces
    #-- IFjordPoints.sol
```

Roles

- **AuthorizedSender**: Address of the owner whose cancellable Sablier streams will be accepted.
- **Buyer**: User who aquire some ERC20 FJO token.
- **Vested Buyer**: User who get some ERC721 vested FJO on Sablier created by Fjord.
- **FJO-Staker**: Buyer who staked his FJO token on the Fjord Staking contract.
- **vFJO-Staker**: Vested Buyer who staked his vested FJO on Sablier created by Fjord, on the Fjord Staking contract.
- **Penalised Staker**: a Staker that claim rewards before 3 epochs or 21 days.
- **Rewarded Staker**: Any kind of Stakers who got rewarded with Fjord's reward or with ERC20 BJB.
- **Auction Creator**: Only the owner of the AuctionFactory contract can create an auction and offer a valid project token earn by a "Fjord LBP event" as an auctionToken to bid on.
- **Bidder**: Any Rewarded Staker that bid his BJB token inside a Fjord's auctions contract.

Issues found

Severity	Count of Findings
Medium	1
Low	1

Findings

Medium Risk Findings

M-01. Incorrect argument passed while calling `FjordPoints::onUnstaked` inside `FjordStaking::_unstakeVested`

Low Risk Findings

L-01. Incorrect event emission in `FjordPoints::distributePoints`

Medium Risk Findings

M-01. Incorrect argument passed while calling `FjordPoints::onUnstaked` inside `FjordStaking::_unstakeVested`

Relevant Github Links

https://github.com/Cyfrin/2024-08-fjord/blob/main/src/FjordStaking.sol#L561

Summary

- `FjordStaking::_unstakeVested` is an internal function, which can be called via `unstakeVested` and `onStreamCanceled` function.
- `unstakeVested` function allows users to unstake their whole NFT, while `onStreamCanceled` is called when the stream creator cancels the stream and the function either unstakes the NFT or removes the staked amount for the NFT.
- But when `onStreamCanceled` function calls `_unstakeVested`, where `_unstakeVested` function will unstake the amount for the NFT and also updates the staked amount on `FjordPoints` contract via `FjordPoints::onUnstaked`.
- But instead of passing the sablier NFT owner it passes `msg.sender` i.e., Sablier contract to the `onUnstaked` function and as Sablier contract has 0 staked amount on fjord points, it will result in a revert which thus reverts the `onStreamCanceled` function but Sablier contract will ignore this revert.
- Thus the stream was cancelled but staked data remains unupdated on `FjordStaking` contract and the sablier NFT owner will be able to receive rewards on the whole value including the value which was supposed to be unstaked.

Vulnerability Details

- The vulnerability is present in the `FjordStaking::_unstakeVested` where it passes `msg.sender` as user for which unstaking is done to `onUnstaked` function instead of `streamOwner`.
- Here, `msg.sender` will not be the sablier NFT owner (stream owner) in all cases.
- For the case when `_unstakeVested` is called by `onStreamCanceled` function, then `msg.sender` is going to be the Sablier contract and not the stream owner as a result of which it tries to reduce the staked amount data of Sablier contract on `FjordPoints` contract, thus leading to revert.
- This revert would thus cause the `onStreamCanceled` function to revert and all the updations related to unstaking of amount of user will be undone, but the stream creator cancelling of the stream would be successful as the revert caused by `onStreamCanceled` is ignored by Sablier contract.

Impact

- When stream is cancelled by creator, the NFT holder's will have no ownership of the unstreamed amount. But as `onStreamCanceled` faced a revert, the unstreamed amount is still in the accounting of the `FjordStaking` and `FjordPoints` contract.
- Thus, user will be able to receive rewards on the amount which they do not own as it is still in their staked amount.

Tools Used

Manual Review

Recommendations

In `FjordStaking::_unstakeVested`, instead of passing `msg.sender` to `onUnstaked` function, pass `streamOwner`

```
- points.onUnstaked(msg.sender, amount);
+ points.onUnstaked(streamOwner, amount);
```

Low Risk Findings

L-01. Incorrect event emission in `FjordPoints::distributePoints`

Github Link

https://github.com/Cyfrin/2024-08-fjord/blob/main/src/FjordPoints.sol#L247

Summary

- `distributePoints` function emits the event `PointsDistributed`.
- The first parameter of the event is the total amount of points that are distributed.
- But for the case when `weeksPending` has a value >1, then total points distributed will be `weeksPending * pointsPerEpoch`.
- But `pointsPerEpoch` is passed to the event for every case even if `weeksPending` has a value >1.

Vulnerability Details

- The vulnerability is present in the `distributePoints` function where it passes incorrect argument while emitting the `PointsDistributed` event.
- The event expects 2 parameters - total points distributed and points per token.
- But for the total points distributed it passes `pointsPerEpoch` which denotes the total points for a single epoch and not total points distributed.
- As for the case when `weeksPending` > 1, the total points distributed will be the product of `pointsPerEpoch` and `weeksPending`, but instead `pointsPerEpoch` is passed for every case.

Impact

Incorrect event data emitted leads to incorrect off-chain updation.

Tools Used

Manual Review

Recommendations

Correct the event emission as below:

```
- emit PointsDistributed(pointsPerEpoch, pointsPerToken);
+ emit PointsDistributed(pointsPerEpoch * weeksPending, pointsPerToken)
    ;
```