Protocol Audit Report

Version 1.0

Tim Sigl

Protocol Audit Report

Tim Sigl

2024/08

Prepared by: Lead Security Researcher Tim Sigl

Table of Contents

Protocol Summary

Fjord connects innovative projects and engaged backers through a community-focused platform, offering fair and transparent LBPs and token sale events.

Contest Summary

Sponsor: Fjord

Dates: Aug 20th, 2024 - Aug 27th, 2024

See more contest details here

Disclaimer

The Tim-team makes all effort 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 by the team 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

The findings described in this document correspond with the following commit hash:

```
0312fa9dca29fa7ed9fc432fdcd05545b736575d
```

Scope

nSLOC: 662

```
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	Number of Findings
High	1
Medium	0
Low	0
Info	0
Total	1

Findings

High

[H-1] Funds Locked in `AuctionFactory` Contract When Auctions End Without Bids

Summary When an auction ends without any bids, all ERC20 auction tokens are returned to the owner. However, since an auction is created through the `AuctionFactory` contract, it will be the owner. Consequently, all tokens are transferred to the `AuctionFactory`, which lacks a withdrawal mechanism. This results in the tokens becoming permanently locked within the `AuctionFactory` contract.

Vulnerability Details `FjordAuction::auctionEnd` can be called once an auction ends. In the case no bids were placed, all auction tokens are transferred to the owner (`FjordAuction` line 192-195):

```
if (totalBids == 0) {
    auctionToken.transfer(owner, totalTokens);
    return;
}
```

In the constructor of the `FjordAuction` contract the `owner` is set to the `msg.sender` (line 134). The issue here is that the auction is created through the `AuctionFactory` contract (`AuctionFactory` line 52-66):

```
function createAuction(
        address auctionToken,
        uint256 biddingTime,
        uint256 totalTokens,
        bytes32 salt
    ) external onlyOwner {
        address auctionAddress = address(
@>          new FjordAuction{ salt: salt }(fjordPoints, auctionToken,
    biddingTime, totalTokens)
        );

        // Transfer the auction tokens from the msg.sender to the new
            auction contract
        IERC20(auctionToken).transferFrom(msg.sender, auctionAddress,
            totalTokens);

        emit AuctionCreated(auctionAddress);
    }
```

As a result, the `AuctionFactory` contract becomes the owner of each `FjordAuction` contract, not the caller of `AuctionFactory::createAuction`. Consequently, all auction tokens from unsuccessful auctions are sent to the `AuctionFactory` contract, where they become stuck due to the lack of a withdrawal mechanism.

Proof of Concept A forge test demonstrating this vulnerability has been provided. The test creates an auction, allows it to end without bids, and verifies that the tokens are indeed transferred to the `AuctionFactory` contract. Copy the code below into a solidity file in the `test` directory and run the test.

Actors:

- **Deployer**: Deployer of the `FjordAuctionFactory` contract who should receive the auction tokens.
- **User**: The user who ends the auction without any bids.

Working Test Case:

```
// SPDX-License-Identifier: AGPL-3.0-only

pragma solidity =0.8.21;

import {Test} from "forge-std/Test.sol";
import {ERC20} from "lib/openzeppelin-contracts/contracts/token/ERC20/
    ERC20.sol";
import {FjordAuction} from "../src/FjordAuction.sol";
import {AuctionFactory} from "../src/FjordAuctionFactory.sol";
import {FjordPoints} from "../src/FjordPoints.sol";

contract AuctionERC20 is ERC20 {
    constructor() ERC20("Auction Token", "AT") {
        _mint(msg.sender, 1_000_000e18);
    }
}

contract AuditTest is Test {
    FjordAuction public fjordAuction;
    AuctionFactory public auctionFactory;
    FjordPoints public fjordPoints;
    AuctionERC20 public auctionToken;
    FjordAuction public auction;

    address deployer = makeAddr("deployer");
    address user = makeAddr("user");

    function setUp() public {
        vm.startPrank(deployer);
        fjordPoints = new FjordPoints();
        auctionFactory = new AuctionFactory(address(fjordPoints));
        auctionToken = new AuctionERC20();
        auctionToken.approve(address(auctionFactory), 100_000 * 10 **
            18);
        // Create an auction with 100_000 tokens of the auctionToken
        auctionFactory.createAuction(
            address(auctionToken),
            block.timestamp + 100,
            100_000e18,
            bytes32(0)
        );
        auction = FjordAuction(0
            xF50d4eC7549ce8C9B75C0b89B6F784B8F5c8aEFA);
        vm.stopPrank();
    }

    function test_auction_end_without_bids_will_lock_funds() public {
        vm.startPrank(user);
        // Move time past the auction end
        vm.warp(block.timestamp + 101);
        auction.auctionEnd();
        vm.stopPrank();
        // Funds will be stuck in the auction factory which doesn't
            have a way to withdraw them
        assertEq(auctionToken.balanceOf(address(auctionFactory)), 100
            _000e18);
        // The auction owner is the auction factory
        assertEq(auction.owner(), address(auctionFactory));
    }
}
```

Impact

- All auction tokens without any bids will be stuck in the `AuctionFactory` contract.
- Impact: High
- Likelihood: Medium (depends on the number of auctions without bids)

-> Severity: **High**

Tools Used

- Manual code review
- Forge unit test

Recommendations There are two options to fix this issue:

1. Implement a withdrawal mechanism in the `AuctionFactory` contract to allow the owner to withdraw the auction tokens.
2. Change the owner of the `FjordAuction` contract to the same owner of the `AuctionFactory` .