Smart Contract Audit Report — LiquidRon Vault

2⃣ Project Overview

Project Name: LiquidRon

Contract Type: Staking / Yield Vault

Audit Type: Manual Review

Auditor: Hari (Blockchain Auditor)

Repository: https://github.com/code-423n4/2025-01-liquid-ron

Issue H-1: Operator fee withdrawal reduces totalAssets, causing users to receive fewer assets on redemption

Summary :

The vault allows users to deposit RON and receive shares representing their proportional ownership. However, when the operator withdraws their accumulated fee, the total assets of the vault decrease, while user shares remain unchanged. As a result, the share-to-asset ratio becomes unbalanced, causing users to receive fewer RON during redemption than they originally deposited.

Vulnerability Detail :

When the operator withdraws their accumulated fee, the vault’s total assets decrease, but user shares stay the same. This makes the share-to-asset ratio lower, so users get slightly less RON when they redeem

Impact :

Users may receive fewer assets than they deposited when redeeming their shares. This creates an unfair loss for users and can break trust in the vault’s accounting accuracy

Proof of code

```
   function test_ShowusergetreducedAmount() external{

       address user1 = makeAddr("user1");
       vm.deal(user1 , 100 ether);

       address user2 = makeAddr("user2");
       vm.deal(user2 , 100 ether);

       vm.prank(user1);
       liquidRon.deposit{value: 100 ether}();

       vm.prank(user2);
       liquidRon.deposit{value: 100 ether}();


       uint256 amount = 100000 ether + 100 ether + 100 ether;
       liquidRon.deposit{value: amount}();
       uint256 delegateAmount = amount / 7;
       uint256[] memory amounts = new uint256[](5);
       for (uint256 i = 0; i < 5; i++) {
           amounts[i] = delegateAmount;
       }

       liquidRon.delegateAmount(0, amounts, consensusAddrs);
       skip(86400 * 365 + 1);

       liquidRon.harvest(0 , consensusAddrs);

       console.log(liquidRon.operatorFeeAmount());

       uint256 user2shares = liquidRon.balanceOf(user2);

       console.log(user2shares);

       uint256 expctedassetsamount = liquidRon.previewRedeem(user2shares);
       console.log("Expected Asset Amount" , expctedassetsamount);

       vm.prank(address(this));
       liquidRon.fetchOperatorFee();

       vm.prank(user2);
       uint256 GetAmount = liquidRon.redeem(user2shares , user2 , user2);

       console.log("Actually Received Asset Amount" , GetAmount);

       assertGt(expctedassetsamount , GetAmount , " not true ");

   }
```

```
Run the Test code : forge test --mt test_ShowusergetreducedAmount -vvv
```

Recommendation :

```
function totalAssets() public view override returns (uint256) {
       return super.totalAssets() + getTotalStaked() + getTotalRewards() - operatorFeeAmount;
   }
```

Issue M-1: onlyOperator modifier incorrectly blocks legitimate operators from executing functions

Summary:

The `onlyOperator` modifier is intended to allow legitimate operators to perform specific actions, but due to an incorrect Check , it blocks them from executing these functions.

Vulnerability Detail:

The `onlyOperator` modifier is intended to allow the contract owner or authorized operators to execute certain functions. However, the condition `if (msg.sender != owner() || operator[msg.sender]) revert ErrInvalidOperator();` is incorrect. Currently, it reverts when `msg.sender` is a legitimate operator because `operator[msg.sender]` evaluates to true, blocking valid operators from performing actions.

Impact :

As a result of the incorrect check in the `onlyOperator` modifier, legitimate operators are unable to execute functions requiring operator privileges. This can block authorized operators from performing critical actions while potentially allowing unintended users to trigger certain functions, which may disrupt contract functionality or break expected workflows.

Vulnerble Code :

```
 modifier onlyOperator() {
      if (msg.sender != owner() || operator[msg.sender]) revert ErrInvalidOperator();
      _;
  }
```

Recommendation :

```
modifier onlyOperator() {
      if (msg.sender != owner() || ! operator[msg.sender]) revert ErrInvalidOperator();
      _;
  }
```