Quantstamp

Sperax - Farms

Executive Summary

This audit report was prepared by Quantstamp, the leader in blockchain security.

Type	Yield Farm
Timeline	2024-06-05 through 2024-06-12
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Demeter Protocol
Source Code	• Sperax/Demeter-Protocol #e2e7888
Auditors	• Jonathan Mevs Auditing Engineer • Roman Rohleder Senior Auditing Engineer • Gereon Mendler Auditing Engineer

Documentation quality	Medium
Test quality	High
Total Findings	17 Fixed: 14 Acknowledged: 2 Mitigated: 1
High severity findings	4 Fixed: 3 Acknowledged: 1
Medium severity findings	3 Fixed: 2 Mitigated: 1
Low severity findings	6 Fixed: 6
Undetermined severity findings	0
Informational findings	4 Fixed: 3 Acknowledged: 1

Summary of Findings

In this audit, Quantstamp reviewed Sperax's Demeter Protocol, which will be deployed on Arbitrum. The Demeter Protocol allows for users to create their own Yield Farms for a flat fee. Creators of Farms specify RewardToken managers which can configure the rate at which Reward Token rewards accumulate for depositors. Optionally, these Reward Token managers can be assigned to an instance of the Rewarder contract, which contains logic for maintaining rewards at a constant APR rate, based on market conditions. Farms will eventually expire, although Expirable Farms can be extended for a fee. Farm creators also have the opportunity to allow users to lock up funds for a specified cooldown period, to earn additional rewards. After initiating a cooldown, users must wait for that extra time in order to withdraw. The purpose of including the lock up period is to ensure locked tokens for the protocol while allowing depositors to earn extra rewards. Farms are intended to support liquidity deposits from protocols representing liquidity as an ERC20 instance of the token pair, in the style of Uniswap V2, or as an ERC721 instance that represents a liquidity position, in the style of Uniswap V3.

We identified 18 issues and include 4 suggestions for the general improvement of the codebase and adherence to best practices. SPE-1 and SPE-2 can both be addressed with further input validation. We recommend disallowing deposits and withdrawals to occur in the same block, in order to address the risk described in SPE-3. Use of OpenZeppelin's SafeCast library can address SPE-4. In addition to these high severity issues, we identified a number of issues that should be considered to ensure the accuracy of reward distribution and other accounting. Further, there is a significant centralization aspect that users should be aware of in which Rewards are only available if they are accurately maintained and supplied to the farm by its creator.

Their test suite is extensive, although, documentation could be improved by including more technical documentation and contract specifications. The Spearx team was very collaborative and helpful throughout the audit.

Fix Review Update

The team has addressed all the issues by either fixing or acknowledging them. We appreciate the team's responsiveness and commitment to security.

ID	DESCRIPTION	SEVERITY	STATUS
SPE-1	Underflow in `Farm._getRewardAccrualTimeElapsed()`	● High	Fixed
SPE-2	Oracles Are Not Checked for Validity or Staleness	● High	Acknowledged

ID	DESCRIPTION	SEVERITY	STATUS
SPE-3	Reward Inflation Through a Flash Loan	● High	Fixed
SPE-4	Use of Unsafe Cast Operation May Lead to Incorrect Accounting	● High	Fixed
SPE-5	`Rewarder` Ignores Accumulated Rewards of a Farm	● Medium	Fixed
SPE-6	Missing Reward Calculation Update at `farmEndTime`	● Medium	Mitigated
SPE-7	Loss of Precision in `Rewarder._calibrateReward()`	● Medium	Fixed
SPE-8	Rewarder Implementation Cannot Be Updated	● Low	Fixed
SPE-9	`FarmRegistry.updatePrivilege()` Does Not Validate Deployer Address	● Low	Fixed
SPE-10	Implementation Contracts Maintain Meaningful State	● Low	Fixed
SPE-11	Missing Input Validation	● Low	Fixed
SPE-12	Adding Rewards Doesn't Always Update Reward Data	● Low	Fixed
SPE-13	`recoverRewardFunds()` Not Callable when Using `Rewarder`	● Low	Fixed
SPE-14	`E20Farm` Has Repetitive Inheritance	● Informational	Fixed
SPE-15	Farm Duration Can Be Extended Indefinitely Past `MAX_EXTENSION`	● Informational	Fixed
SPE-16	5000 Is Not Equal to 50% in 1e8 Precision	● Informational	Acknowledged
SPE-17	`Rewarder._normalizeAmount()` Reverts for Tokens with More than 18 Decimals	● Informational	Fixed

Assessment Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

Disclaimer
Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

1. Code review that includes the following
   1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   3. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   1. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   2. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Scope

Files Included

Repo: https://github.com/Sperax/Demeter-Protocol/tree/dev/contracts(e2e78880da480af494bcc1c28105c37b76a361fd)

// Base Contracts
contracts/Farm.sol
contracts/FarmStorage.sol
contracts/FarmDeployer.sol
contracts/FarmRegistry.sol

// Features or plugins
contracts/features/ExpirableFarm.sol
contracts/features/OperableDeposit.sol

// Demeter rewarder
contracts/rewarder/Rewarder.sol
contracts/rewarder/RewarderFactory.sol
contracts/utils/TokenUtils.sol

// Demeter ERC-20 Farms
contracts/e20-farms/E20Farm.sol

// Demeter ERC-721 Farms
contracts/e721-farms/E721Farm.sol
contracts/e721-farms/camelotV2/CamelotV2Farm.sol
contracts/e721-farms/camelotV3/CamelotV3Farm.sol
contracts/e721-farms/uniswapV3/UniV3ActiveLiquidityFarm.sol
contracts/e721-farms/uniswapV3/UniV3Farm.sol

Files Excluded

Repo: https://github.com/Sperax/Demeter-Protocol/tree/dev/contracts(e2e78880da480af494bcc1c28105c37b76a361fd)

All other files

Operational Considerations

- In Farms that rely on a `Rewarder` as an implementation for maintaining an APR rate emission for a given Reward Token, the rewards must be actively calibrated by the team members, or the community members. Whether rewards are available to depositors is totally dependent on the RewardToken Managers configured by Farm owners.
- Reward funds must be actively kept in the farm contract, otherwise users won't get any rewards and rewards won't accumulate until they are deposited again.
- Maintenance of a configured APR for Farm rewards through the `Rewarder` contract are dependent on a single oracle that can be reassigned to any address at any time. The Oracle was out of scope of this audit.
- For E20 farms, if there are any fee-on-transfer tokens supported, there will be inaccurate accounting done when depositing. the contract will consider that there were more funds deposited than what actually transferred into the contract.
- `FarmRegistry` is upgradeable.

Key Actors And Their Capabilities

`FarmDeployer`

- `owner`, as in, the team account that deployed the Deployer contract

    - `renounceOwnership()`
    - `transferOwnership()`
    - `updateFarmImplementation()`, assigning a new address to the implementation contract from which new Farms will be cloned

`Farm`

- `owner`, as in, the farm admin specified during creation
    - `renounceOwnership()`
    - `transferOwnership()`
    - `updateCooldownPeriod()`
    - reassign the length of cooldown for the lockup fund
    - `farmPauseSwitch()`
    - if the farm is open, the owner can toggle the pause state of the farm
    - a paused farm:
        - does not enforce cooldown period, nor does it support a cooldown to be initiated
        - does not accrue rewards
        - does not support deposits
    - `closeFarm()`
    - pauses, and closes farm, capturing the final reward data for all holders.
    - the configured token manager withdraws all funds that are not currently allocated for rewards
    - the reward rate for this token is set to zero
    - `recoverERC20()`
    - withdraw any other token besides the reward token from the farm contract
    - in the case of an `E20Farm`, the `farmToken` cannot be withdrawn
    - `updateFarmStartTime()`
    - if the farm start time has not yet come, then the owner can reassign the farm start time
- `Token Manager`
    - `recoverRewardFunds()`
    - withdraw funds in excess of what is accumulated in rewards to the Token Manager address
    - `setRewardRate()`
    - defining the rewards per second of the Reward Token accumulating
    - `updateRewardData()`
    - assigns the token manager to a new address
    - can technically only be invoked by the owner of the Rewarder contract

`FarmRegsitry`

- `owner`, as in the account that invoked `initialize()` on the contract
    - `renounceOwnership()`
    - `transferOwnership()`
    - `registerFarmDeployer()`
    - assigning a farm deployer contract that is allowed to invoke `registerFarms()` to register user created farms with the registry
    - `removeDeployer()`
    - remove a previously registered deployer address
    - `updatePrivilege()`
    - assign a deployer address to be considered privileged or not. privileged deployers don't pay fees when creating new farms
    - `updateFeeParams()`
    - assign the feeReceiver, feeToken, and feeAmount paid when creating new farms

`ExpirableFarm`

- `owner`, as in the Farm Admin assigned on the creation of the farm
    - `extendFarmDuration()`
    - pay a fee to extend the end time by some amount of days defined, inclusively, between 100-300 days
    - `updateFarmStartTime()`
    - update the start time of the Farm to a valid time in the future

`E20Farm.sol`

- `owner`, as in the Farm Admin assigned on the creation of the farm
    - All privileges listed under `ExpirableFarm.sol`
    - `recoverERC20()`

`CamelotV2Farm.sol`

- `owner`, as in the Farm Admin assigned on the creation of the farm
    - All privileges listed under `ExpirableFarm.sol`
    - `updateFarmStartTime()`

`CamelotV3Farm.sol`

- `owner`, as in the Farm Admin assigned on the creation of the farm
    - All privileges listed under `ExpirableFarm.sol`
    - `updateFarmStartTime()`

`UniV3Farm.sol`

- `owner`, as in the Farm Admin assigned on the creation of the farm
    - All privileges listed under `ExpirableFarm.sol`
    - `updateFarmStartTime()`

`RewarderFactory`

- `owner`, as in the team owned account that deployed this contract
  - `transferOwnership()`
  - `renounceOwnership()`
  - `updateOracle()` to redefine the address used by the Rewarder contract when updating rewards

`Rewarder`

- `owner`, as in the account that deployed the Rewarder through the factory
  - `transferOwnership()`
  - `renounceOwnership()`
  - `toggleCalibrationRestriction()`, toggling whether or not the owner can be the only user to calibrate rewards
  - `updateTokenManagerOfFarm()`, invoking `IFarm.updateRewardData()` to reassign the token manager of the farm
  - `updateAPR()`, redefine the APR and recalibrate rewards accordingly
  - `recoverERC20()` withdraw any amount of any ERC20 tokens in the contract
  - `updateRewardConfig()` to assign a new reward configuration for a farm

Findings

SPE-1 Underflow in `Farm._getRewardAccrualTimeElapsed()`

● High ⓘ Fixed

Update

Marked as "Fixed" by the client.

Addressed in: `edb854e128420bb1723ef517f79c06b08bfbb6d7`.

The client provided the following explanation:

```
• The bug was introduced in Demeter UniswapV3 Farm upgrades #21.
• Replaced lastFundUpdateTime with farmStartTime.
    -_getRewardAccrualTimeElapsed now returns 0 if block.timestamp > farmStartTime or
  lastFundUpdateTime = 0.
• Updated the logic for when lastSecondsInside is updated to address issues in
  UniV3ActiveLiquidityFarm.
• Adjusted test cases to reflect these changes.
```

File(s) affected: `Farm.sol`

Description: When the Farm is set up using `_setupFarm()`, a `_farmStartTime` is assigned which marks the start of reward accumulation. This timestamp is stored in the `lastFundUpdateTime` variable, which is later meant to keep track of the last update time. After the setup is completed, the Farm is already considered active and open such that deposits can already be made. This in turn attempts to update the reward calculations using the `_updateFarmRewardData()` function. This function does not check the intended start time of the Farm, and instead retrieves the elapsed time since the last update from `_getRewardAccrualTimeElapsed()`, where an unchecked subtraction occurs. However, since `lastFundUpdateTime` stores the start time, which can be in the future, this calculation will underflow and result in massive time periods, for which rewards will be incorrectly accrued leading to the loss of all reward tokens already present in the contract. This exploit can occur at any time between the setup of the Farm and the specified start time.

Exploit Scenario:

1. A new Farm is created to have a Farm start time defined for 2 seconds in the future. `Farm._setupFarm()` updates the state variable, `lastFundUpdateTime` to be this configured start time, `lastFundUpdateTime = 2`.
2. 1 second later (`block.timestamp = 1`), a user deposits into this Farm. During the deposit, `_updateFarmRewardData()` is called, which in turn, calculates the time used for determining the accrued rewards to be `_getRewardAccrualTimeElapsed()`.
3. The time that has passed will calculated to be `1 - 2 = type(uint256).max` in `_getRewardAccrualTimeElapsed()` due to the subtraction happening in the `unchecked` block.
4. Now, in `_updateFarmRewardData()`, when calculating the `_getAccRewards()`, the accumulated rewards will be equivalent to the balance of Reward Tokens in the contract, which will be totally allocated to this sole depositor.
5. Immediately following this deposit, the user could claim their rewards to drain the contract of all Reward tokens.

Recommendation: In `_getRewardAccrualTimeElapsed()`, if `lastFundUpdateTime` is in the future, return zero.

SPE-2 Oracles Are Not Checked for Validity or Staleness

● High ⓘ Acknowledged

Update

Marked as "Acknowledged" by the client.

The client provided the following explanation:

```
• We are going to use USDs Master Oracle contract which is already audited.
• Deployed contract:
  https://arbiscan.io/address/0x14D99412dAB1878dC01Fe7a1664cdE85896e8E50#readContract
• Audit report: https://github.com/Sperax/USDs-v2/blob/main/audit/Quantstamp-USDsV2-Audit.pdf
• We will ensure all the future price feeds that are integrated on the masterOracle have a
  staleness proof.
```

File(s) affected: `Rewarder.sol`

Description: Oracles usually include a timestamp of the last update. If this timestamp is too old, the retrieved data is considered stale and should not be relied upon. Additionally, if the price returned is 0, the Oracle data should not be trusted.

Recommendation: Sufficiently validate oracle output.

SPE-3 Reward Inflation Through a Flash Loan

● High ⓘ Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `e1359d81959883d4485f09e48e28afa970627d89`.
The client provided the following explanation:

```
Added a depositTs in struct Deposit, it is updated in deposit() and increaseDeposit() and validated in
withdraw() and decreaseDeposit()
```

This has been fixed as users can no longer deposit or withdraw in the same block.

File(s) affected: `Rewarder.sol`, `Farm.sol`

Description: As deposits and withdrawals are allowed to occur within the same block, it is possible for a user to significantly inflate the rewards through a Flash Loan. The exploit scenario below describes specifics. This attack is more feasible for smaller farms.

Exploit Scenario:
1. The attacker makes an initial deposit, which will later be needed to claim rewards.
2. The attacker receives a flash loan, from which they receive a large amount of farm tokens.
3. The attacker deposits these farm tokens, calibrates the rewards, and immediately withdraws their deposit, and repays the flash loan.
4. Time passes.
5. The attacker claims rewards at a highly inflated rate due to the rewards being calibrated at a time when there were substantial funds in the farm.

Recommendation: To mitigate this risk, consider disallowing deposits and withdrawals to occur in the same block. This can be done by including a deposit timestamp in the `Deposit` struct, and ensuring that some time has passed when a withdrawal is processed, or a deposit is decreased.

SPE-4 Use of Unsafe Cast Operation May Lead to Incorrect Accounting

● High ⓘ Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `401c7378c197642e7eda8f75eb3b9da11ccc250b`.
The client provided the following explanation:

```
We have implemented OpenZeppelin SafeCast in all instances where we previously performed uint256 to
uint128 typecasting.
```

File(s) affected: `TokenUtils.sol`, `CamelotV3Farm.sol`, `UniV3Farm.sol`

Description: Function `getUniV3TokenAmounts()` downcasts parameter `_liquidity` from `uint256` down to `uint128` using the unsafe cast operation `uint128(...)`, which will truncate values bigger than `2**128-1` and thereby lead to incorrect accounting in all follow up operations.

The same applies to functions:
1. `TokenUtils.getCamelotV3TokenAmounts()`.
2. `CamelotV3Farm.decreaseDeposit()`.
3. `UniV3Farm.decreaseDeposit()`.

Recommendation: We recommend using a safe cast alternative, i.e. OpenZeppelins SafeCast, which reverts for values that would truncate, preventing unexpected follow up computations or otherwise ensuring that `_liquidity` may never exceed a value higher than `2**128-1`.

SPE-5 `Rewarder` Ignores Accumulated Rewards of a Farm

• Medium ⓘ Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `42b5b031e08849fab983589eb609e2d7ae42a452`.
The client provided the following explanation:

```
Implemented changes as recommended. Using farm.getRewardBalance(token) instead of token.balanceOf(farm)
in the Rewarder contract.
```

File(s) affected: `Rewarder.sol`

Description: The `Rewarder` contract does not consider the accumulated rewards stored in a farm when querying the balance of `REWARD_TOKEN` in the farm. Due to this, the function `rewardsEndTime()` can overestimate the time period in which there are rewards available for a farm, as described in the Exploit Scenario below. In `Rewarder._calibrateRewards()`, `_farmRwdBalance` is calculated to be the `REWARD_TOKEN` balance of the farm, without considering the accumulated rewards. Therefore, there is an overestimation that can lead to a situation when the `Rewarder` contract is not transferring rewards to the farm, when the farm is in need of more tokens.

Exploit Scenario:

1. `rewardsEndTime()` is called at time = 50
    1. The farm in consideration has a balance of 100 reward tokens, and a reward rate of 1 reward per second.
        1. As 50 secs have passed, the accumulated rewards = 50.
    2. No rewards have been claimed.
    3. The rewarder has a balance of 100 tokens, and a totalReward Rate of 1 reward per second.
2. The local variable `rewardsEndingOn` is calculated to be `100 + 100 = 200`, which returns `block.timestamp + 200`.
3. This is inaccurate, as there are only 150 seconds of reward time remaining, as 50 seconds of rewards have already accumulated.

Recommendation: Instead of directly querying the balance of the `REWARD_TOKEN` in the farm, invoke `Farm.getRewardBalance()`, as that function considers the accumulated rewards in the Farm.

SPE-6 Missing Reward Calculation Update at `farmEndTime`

• Medium ⓘ Mitigated

Update

Marked as "Mitigated" by the client.
Addressed in: `46371b304de4d78b72c382ad326dc25926940947`.
The client provided the following explanation:

```
We made _updateFarmRewardData public and renamed it to updateFarmRewardData. This allows the function
to be triggered externally before farmEndTime to accrue user rewards, which cannot be accrued after the
farm expires.
  We considered modifying _updateFarmRewardData to accrue rewards until farmEndTime, even after farm
expiration. However, this approach proved too complex with multiple edge cases. To keep the code
simple, we decided to make this function public and resolve the issue using off-chain measures.
```

File(s) affected: `ExpirableFarm.sol`

Description: This extension to the Farm contract specifies a `farmEndTime` after which the Farm is considered closed, found in the overridden method `isFarmOpen()`, shown in the snippet below:

```
function isFarmOpen() public view virtual override returns (bool) {
    return super.isFarmOpen() && (block.timestamp <= farmEndTime);
}
```

After this time, no new deposits can be made and no new reward state updates calculated. That means that rewards that should accrue between the last update and the `farmEndTime` are mistakenly not accounted for.

Recommendation: Update the `_updateFarmRewardData()` function to calculate the rewards correctly for that time period if the farm is currently closed.

SPE-7 Loss of Precision in `Rewarder._calibrateReward()`

• Medium ⓘ Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `bfbbd5b6e7134244a360115032c9ee3fad3cef85`.
The client provided the following explanation:

```
Implemented the recommended fix. Updated the rewardRate calculation statement.
```

File(s) affected: `Rewarder.sol`

Description: The calculation `(((farmRewardConfig.apr * totalValue) / (APR_PRECISION * DENOMINATOR)) / ONE_YEAR) * priceData.precision` divides before multiplying with the defined precision. This truncates the values in the divide step.

Recommendation: We recommend restructuring this expression such that the multiplications occur before the divisions to mitigate precision loss, i.e.:

```
rewardRate = (farmRewardConfig.apr * totalValue * priceData.precision) / (APR_PRECISION * DENOMINATOR *
ONE_YEAR * priceData.price)
```

SPE-8 Rewarder Implementation Cannot Be Updated

• Low ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `7b881c496bbac9c75bb281673238ec80253e2f00`.
The client provided the following explanation:

```
Added a function in the RewarderFactory to update rewarderImplementation and added test cases.
```

File(s) affected: `Rewarder.sol`, `RewarderFactory.sol`

Description: There is no way to update the implementation address of the `Rewarder` contract used by `RewarderFactory`. This is unlike the `FarmDeployer`, that allows the owner to update the address used for the `Farm` implementation. If the `Rewarder` implementation address needs to be updated in the future, there is no way to do so.

Recommendation: Consider including functionality to update the implementation address of the `Rewarder` contract. Otherwise, it can be declared as `immutable`.

SPE-9 `FarmRegistry.updatePrivilege()` Does Not Validate Deployer Address

• Low ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `366a4ba4970f1635db1ad90b379e26243ec02c18`.
The client provided the following explanation:

```
Privileged deployers and registered farm deployers are separate. Privileged deployers receive a
discount on farm creation fees, whereas registered farm deployers are authorized addresses for farm
creation.
  To avoid this confusion in the future, we will rename isPrivilegedDeployer to isPrivilegedUser for
clarity.
```

The variable renaming has significantly clarified the intended functionality in the code.

File(s) affected: `FarmRegsitry.sol`

Description: Privileges can be updated for deployers that aren't already added to the list of registered deployers. Further, if a privileged deployer is being defined, they are not added to the deployers list.

Recommendation: Consider if a new deployer address can be added through this function. If so, add it to the list in this function. Otherwise, confirm that the deployer is present in the list. To avoid a linear search at this step, consider including the `id` of the deployer as a parameter in the `updatePrivilege()` function.

SPE-10 Implementation Contracts Maintain Meaningful State

• Low ⓘ Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `92b87cb7c29fbb020ffc1c87a8a7342ca4d4d9e7`.
The client provided the following explanation:

```
• To address this we have removed all initializers from the constructors of the implementation
  contracts and disabled initialization for the implementation contracts.
• We have switched dependencies to the _Upgradeable ones as recommended.
• For all the farms, we have moved the initializer modifier to Farm.sol's _setupFarm()
```

File(s) affected: `Rewarder.sol`, `Farm.sol`

Description: `Rewarder.constructor()` is missing a call to `_disableInitializers()`, meaning the implementation can be initialized many times. As it is an implementation contract, it should not hold any state, however, if anyone were to invoke `initialize()` on this implementation contract, state would be stored there which could lead to unexpected behaviors.

Further, as the non-upgradeable version of `Ownable` is inherited by `Rewarder`, `Rewarder.constructor()` stores the deployer address as the owner of the contract. This is also the case in the `Farm` contract.

Recommendation: Invoke `_disableInitializers()` in `Rewarder.constructor()`. Additionally, inherit `OwnableUpgradeable` and invoke `__Ownable_init()` in the `initialize()` function, in `Rewarder` and `Farm`.

SPE-11 Missing Input Validation

• Low ⓘ Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `3da26bea90a2c0bf43814ab8342b0c03ae440f77`.
The client provided the following explanation:

```
We have added most of the validations mentioned, with a few exceptions discussed in detail below:
1. For point 4, we do not need to check for non-zero as the reward rate can be zero. To ensure a
   reasonable bound, we have changed the reward rate type from uint256 to uint128, which will keep the
   reward rate within a safe range.
2. For point 5, we do not need to make any changes as we can handle any decimal places after the
   addition of PR #70.
3. For point 8.2, we need to enforce the farm ID to be always unique. We have added a @dev natspec
   informing users to provide an appropriate farm ID based on the new farm implementation.
4. For point 9, the APR can be 0. Checking for values smaller than APR_PRECISION * DENOMINATOR is not
   required after the addition of PR #66.
5. For point 10.1, the justification is the same as above.
6. For point 10.2, we can change the maxRewardRate type from uint256 to uint128, thus limiting its
   range.
```

File(s) affected: `Farm.sol`, `FarmRegistry.sol`, `FarmDeployer.sol`, `Rewarder.sol`

Description: It is important to validate inputs, even if they only come from trusted addresses, to avoid human error.

The following places can benefit from further input validation:

1. Fixed `OperableDeposit._decreaseDeposit()` allows decreases by the entire liquidity amount, and is missing a proper error message if the amount exceeds the liquidity.
2. Acknowledged `Farm.getNumSubscriptions()`: `_depositId` not checked to be valid.
3. Fixed `Farm.getRewardRates()`: `_rwdToken` not checked to be valid.
4. Acknowledged `Farm._addRewardData()`: `ERC20(_token).decimals()` not checked to be equal or less than `18`. While unlikely, if the decimals are greater than `18` it may lead to a denial of service within the `Rewarder` contract due to attempted underflows in `Rewarder.sol#L351`.
5. Fixed `FarmRegistry.registerFarm()`: `_farm` not checked to be non-zero or already registered, potentially leading to duplicate entires.
6. Fixed `FarmRegistry.updatePrivilege()`: `_deployer` not checked to be a registered deployer.
7. `FarmDeployer.updateFarmImplementation()`:
   1. Fixed `_newFarmImplementation` not checked to be non-zero or different from previous value.
   2. Acknowledged `_newFarmId` not checked to be non-empty or different from previous value.
8. Fixed `Rewarder.updateAPR()`: `_apr` not checked to be non-zero or smaller than `APR_PRECISION * DENOMINATOR`.
9. `Rewarder.updateRewardConfig()`:
   1. Acknowledged `_rewardConfig.apr` not checked to be non-zero or smaller than `APR_PRECISION * DENOMINATOR`.
   2. Acknowledged `_rewardConfig.maxRewardRate` not checked to be non-zero or otherwise reasonably bound.

Recommendation: Consider adding the recommended checks.

SPE-12 Adding Rewards Doesn't Always Update Reward Data

• Low ⓘ Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `187e7a03e9d2368005cc11f3d142ee1335c57e1a`.
The client provided the following explanation:

```
As per the audit review comment, the farm's reward data was not getting updated because Rewarder sends
the tokens to the farm via safeTransfer instead of addRewards. However when the Rewarder calls
farm.setRewardRate, it updates the farm reward data but the ideal scenario would be to checkpoint
farm's reward data first and then transfer the tokens.
  We have moved the safeTransfer call after the setRewardRate() is called ensuring the rewards are
check-pointed prior to the fund transfer.
  Note: We also acknowledge that the reward data is not updated when the reward is added to the farm
via a ERC20.transfer, but is updated when a user calls addRewards().
```

File(s) affected: `Rewarder.sol`, `Farm.sol`

Description: `Rewarder._calibrateReward()` uses a native ERC20 transfer, instead of invoking `Farm.addRewards()`. By using the native function the input checks and event emission in `Farm.addRewards()` are avoided. More generally, any tokens can be directly transferred into the Farm contract, avoiding the logging in `Farm.addRewards()`.

Recommendation: `Rewarder._calibrateReward()` should invoke `Farm.addRewards()` when depositing new rewards. Further, the team should explore the implications of reward data not always being updated when new funds are deposited.

SPE-13 `recoverRewardFunds()` Not Callable when Using `Rewarder`

• Low ⓘ Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `a98fa090ae9bc665814b7ab29b12fd7f6c9bcef5`.
The client provided the following explanation:

```
Added a wrapper function in Rewarder named recoverRewardFundsOfFarm to call recoverRewardFunds on the
farm from rewarder as token manager.
```

File(s) affected: `Rewarder.sol`

Description: The rewarder contract, when used, gets appointed the role of `RewardData.tknManager` within the `Farm` contract, allowing to call all correspondingly protected functions only through that rewarder contract:

1. `setRewardRate()`.
2. `updateRewardData()`.
3. `recoverRewardFunds()`.

While the `Rewarder` contract does expose corresponding access controlled wrapper functions for `setRewardRate()` and `updateRewardData()`, it does not have a corresponding function for `recoverRewardFunds()`, making it uncallable when using such a rewarder contract.

Recommendation: Consider adding a corresponding access controlled wrapper function within the `Rewarder` contract.

SPE-14 `E20Farm` Has Repetitive Inheritance

• Informational ⓘ Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `f66b64d789de21dc156ed2f44571bd02154fce23`.
The client provided the following explanation:

```
We have simplified the farm hierarchy to remove repetitive Inheritance.
    • OperableDeposit now inherits from Farm instead of ExpirableFarm.
    • Generic farms having increase/decrease liquidity options (ex: E20Farm) inherit Operable farms
    • Specific farm implementations (ex: BalancerV2/ UniV2 etc) inherit Expirable farms. So that it is
      modifiable at the topmost level.
    • Also removed onlyOwner redundant modifier from updateFarmStartTime and renamed _recoverERC20
      function for simplifying the code.
```

File(s) affected: `E20Farm.sol` , `CamelotV2Farm.sol` , `CamelotV3Farm.sol` , `UniV3Farm.sol`

Description: `E20Farm` inherits from both `ExpirableFarm` and `OperableDeposit` which is unnecessary as `OperableDeposit` inherits from `ExpirableFarm` . This is also the case in `CamelotV2Farm` , `CamelotV3Farm` and `UniV3Farm` .

Recommendation: In these contracts, it is only necessary to inherit from `OperableDeposit` , as that contract inherits from `ExpirableFarm` so all child contracts will have access to the functionality of both.

SPE-15 Farm Duration Can Be Extended Indefinitely Past `MAX_EXTENSION`

● Informational ⓘ Fixed

✓ **Update**

Marked as "Fixed" by the client.
Addressed in: `f0d4606cb9dc93f3ed8c9c4fb50c370211e52df4` .
The client provided the following explanation:

```
The code now checks if the new farm end time exceeds the maximum extension limit and reverts the
transaction with the "DurationExceeded" error if it does. This ensures that the farm extension duration
stays within the allowed range.
```

File(s) affected: `ExpirableFarm.sol`

Description: Farm durations can be extended using the `extendFarmDuration()` function, which checks that the extension is within the bounds defined by `MIN_EXTENSION` and `MAX_EXTENSION` . The latter can be exceeded by repeatedly calling the function, since extensions are appended to the current `farmEndTime` .

Recommendation: If this is undesired, check the difference between the current timestamp and the `farmEndTime` .

SPE-16 5000 Is Not Equal to 50% in 1e8 Precision

● Informational ⓘ Acknowledged

i **Update**

Marked as "Acknowledged" by the client.
Addressed in: `b06953c8ba54538ed70295155406ea79b30a276f` .
The client provided the following explanation:

```
Updated precision assignment and comments to make it consistent and avoid confusion.
```

File(s) affected: `Rewarder.sol`

Description: The comment suggests that a value of 5000 would indicate 50%, however the `APR_PRECISION` is set at 1e8 equals one percent. Additionally, `MAX_PERCENTAGE` is defined to be 10,000, which is not 1e8 precision.

Recommendation: Ensure that correct values are used.

SPE-17 `Rewarder._normalizeAmount()` Reverts for Tokens with More than 18 Decimals

● Informational ⓘ Fixed

✓ **Update**

Marked as "Fixed" by the client.
Addressed in: `8906a6ef82759336070666fc7c29a6efd6713024` .
The client provided the following explanation:

```
• We updated the _normalizeAmount function to handle tokens with decimals > 18.
• While resolving the issue we found another BUG: we didn't consider the precision for the
  rewardToken in the rewardRate calculation. To fix that we are normalizing the amounts based on
  the rewardToken's decimals.
```

File(s) affected: `Rewarder.sol`

Description: The calculation `_amount *= 10 ** (18 - _decimals[_token]);` only works for decimals equal or lower than 18. Otherwise the subtraction will revert.

Recommendation: Include a normalization which truncates decimals to 18.

Auditor Suggestions

SPE-S-1 Code Redundancy Reduction

Acknowledged

Update
Marked as "Fixed" by the client.
Addressed in: `e886ca4c0eb40887f8dc628c42a8b63c02159901`.
The client provided the following explanation:

```
1. We have implemented the recommended change.
2. We do agree that the implementation for these functions are quite similar, but further abstraction
   would result into gas cost increase for the transactions involving these functions.
3. Merging computeRewards() and _updateFarmRewardData() is very complicated and might add-up
   unnecessary gas cost post abstraction.
4. We have made deployerList and farms internal. We have kept the functions for making it easier to
   fetch data on frontend if required.
5. CamelotV3 and UniswapV3 farms involve different interfaces, we can try abstracting them, however it
   might result in unwanted complexities. Due to time constraints we will skip it for time being.
```

File(s) affected: `FarmRegistry.sol`, `Farm.sol`, `OperableDeposit.sol`

Description: We list some areas in the code that contain code redundancy below:

- Fixed Instead of including the constant `OperableDeposit.PRECISION`, just use `FarmStorage.PREC`, as it is inherited.
- Acknowledged `OperableDeposit._updateSubscriptionForIncrease()` and `_updateSubscriptionForDecrease()` contains a lot of duplicated code. To reduce contract size and improve code readability, consider combining these functions and adding or subtracting accordingly, based on a boolean specifier. Similarly, this can be done for `_increaseDeposit()` and `_decreaseDeposit()`.
- Acknowledged Both the `computeRewards()` and `_updateFarmRewardData()` functions aim to compute the accrued rewards for some period of time, although the former served only as a getter without applying the new values. Nonetheless, both calculations should be indentical and therefore should not be duplicated to avoid mistakes when updating the implementation.
- Acknowledged The `getFarmDeployerList()` and `getFarmList()` functions serve as simple getters for the `deployerList` and `farms` variables, which are already public and therefore automatically generate getters.
- Acknowledged `CamelotV3Farm.sol` and `UniV3Farm.sol` are almost identical and can be refactored in a way that shares common functions.

Recommendation: Consider removing this redundancy in the code.

SPE-S-2 Contracts Are Missing Interface Specification

Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `d2e53ab53ee535997a2e5ad0c8ab1d7ec8c8856c`.
The client provided the following explanation:

```
We have add the recommended interfaces for Farm.sol, Rewarder.sol, FarmRegistry.sol and
RewarderFactory.sol
```

File(s) affected: `IFarm.sol`, `Farm.sol`, `IRewarderFactory.sol`, `RewarderFactory.sol`, `IFarmRegistry.sol`, `FarmRegistry.sol`

Description: Interfaces are designed to allow other contracts to more easily interact with any contract implementing them. This requires the correct implementation of these interfaces, ensured by specifying their use in the contract implementation. This is missing for the respective implementation contracts.

Recommendation: Specify Interface implementation in `Farm.sol`, `FarmRegistry.sol` and `RewarderFactory.sol`.

SPE-S-3 Renouncable Ownership / Missing 2-step Ownership and Token Manager Transfers

Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Post deployment, the ownership of all Demeter protocol contracts will be held by  a Multisig (with 3
out of 5 signature confirmations), with the intention to transfer it to an on-chain governance once
established.
```

File(s) affected: `Farm.sol`

Description: The Farm implements the `Ownable` interface, which facilitates ownership and transfers thereof, including renouncement. Since transferring ownership can be error prone, it can be a good idea to use 2-step transfers where the role needs to be actively accepted by the recipient. This is also the case for reassigning a Token Manager.

Recommendation: Overwrite the `renounceOwnership()` function and consider using the `Ownable2Step` library instead.

SPE-S-4 Application Monitoring Can Be Improved by Emitting More Events

Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `ec606f47fb1f8d5aac5f0d83bc32ce148e4179f7`.
The client provided the following explanation:

```
• Tracking lastFundUpdateTime is not essential as it is this checkpoint is done in every
  transaction on farm, making it easily derivable.
• updateFarmRewardData variables are mathematically derivable and can be calculated when required.
• We have add the event for FarmEndTimeUpdate at the recommended places.
```

File(s) affected: `Farm.sol`, `ExpirableFarm.sol`

Description: In order to validate the proper deployment and initialization of the contracts, it is a good practice to emit events. Also, any important state transition can be logged, which is beneficial for monitoring the contract, and also tracking eventual bugs, or hacks. Below we present a non-exhaustive list of events that could be emitted to improve the application management:

1. Acknowledged `Farm._updateLastRewardAccrualTime()`: `lastFundUpdateTime`.
2. Acknowledged `Farm._updateFarmRewardData()`: `rewardData[].accRewardBal` and `rewardFunds[].accRewardPerShare[]`.
3. Fixed `ExpirableFarm.updateFarmStartTime()`: `farmEndTime`.

Recommendation: Consider emitting the events.

SPE-S-5 Internal Functions Can Benefit From Further Input Validation

Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
• We are keeping the code as is, this is because _increaseDeposit function is called after multiple
  operations like token transfer etc on the higher level function, hence we do quick validations
  prior to that logic.
```

File(s) affected: `OperableDeposit.sol`

Description: Some internal functions, such as `OperableDeposit._increaseDeposit()` do not include any input validation and rely on the calling function to have such checks. Although this is no issue in the current codebase, as all relevant checks are done in the calling functions, such as zero amount check or validation of deposit, in future versions of the code, contracts that implement `OperableDeposit` could omit these checks. Therefore, the team should consider including the relevant checks in the functions themselves.

Code Documentation

1. Acknowledged The following typographical errors have been noted:
    1. `Farm.sol#L87` : `a` → `an` .
    2. `Farm.sol#L128` : `Egs` → `E.g.` .
2. Fixed Missing or incorrect NatSpec comments:
    1. Fixed `Farm._getAccRewards()` : Missing NatSpec comment for return value.
    2. Fixed `Farm._subscribeRewardFund()` : Positions for NatSpec comments of parameters `_depositId` and `_fundId` are switched.
    3. Fixed `Farm._setupFarm()` : Missing NatSpec comment for parameter `_farmId` .
    4. Fixed `FarmRegistry.initialize()` : Positions for NatSpec comments of parameters `_feeToken` and `_feeReceiver` are switched.
    5. Fixed `FarmDeployer.updateFarmImplementation()` : Missing NatSpec comment for parameter `_newFarmId` .
    6. Acknowledged `E20Farm.initialize()` : Missing NatSpec comment for parameter `_farmId` .
    7. Fixed `CamelotV2Farm.initialize()` : Missing NatSpec comment for parameters `_farmId` , `_router` and `_nftPoolFactory` .
    8. Fixed `UniV3Farm._getLiquidity()` : Missing NatSpec comment for return value.

Adherence to Best Practices

1. Fixed To facilitate logging it is recommended to index address parameters within events. Therefore the `indexed` keyword should be added to the (other) address parameters in
    1. `Farm.RewardAdded()` .
    2. `Farm.RecoveredERC20()` .
    3. `Farm.RewardDataUpdated()` .
    4. `Farm.RewardTokenAdded()` .
    5. `FarmRegistry.FarmDeployerUpdated()` .
    6. `FarmRegistry.FeeParamsUpdated()` .
    7. `FarmRegistry.PrivilegeUpdated()` .
    8. `FarmDeployer.FarmCreated()` .
    9. `FarmDeployer.FarmImplementationUpdated()` .
    10. `ExpirableFarm.ExtensionFeeCollected()` .
    11. `RewarderFactory.OracleUpdated()` .
    12. `RewarderFactory.RewarderDeployed()` .
2. Fixed Since solidity version `0.8.22` for-loop variables are implicitly unchecked, such that it is no longer necessary to wrapped them explicitly within `unchecked {...}` -blocks, leading to improved readability. In this regard, consider the following instances:
    1. `Farm.sol#L162` .
    2. `Farm.sol#L241` .
    3. `Farm.sol#L245` .
    4. `Farm.sol#L293` .
    5. `Farm.sol#L389` .
    6. `Farm.sol#L557` .
    7. `Farm.sol#L563` .
    8. `Farm.sol#L580` .
    9. `Farm.sol#L615` .
    10. `Farm.sol#L641` .
    11. `Farm.sol#L646` .
    12. `Farm.sol#L696` .
    13. `Farm.sol#L704` .
    14. `Farm.sol#L841` .
    15. `Farm.sol#L871` .
    16. `Farm.sol#L887` .
    17. `OperableDeposit.sol#L56` .
    18. `OperableDeposit.sol#L61` .
    19. `OperableDeposit.sol#L80` .
    20. `OperableDeposit.sol#L85` .
3. Acknowledged Consider removing the following functions, which do not add any new functionality to the overriden functions:
    1. `CamelotV2Farm.updateFarmStartTime()` .
    2. `CamelotV2Farm.isFarmOpen()` .
    3. `CamelotV3Farm.updateFarmStartTime()` .
    4. `CamelotV3Farm.isFarmOpen()` .
    5. `UniV3Farm.updateFarmStartTime()` .
    6. `UniV3Farm.isFarmOpen()` .
4. Fixed To improve readability and lower the risk of introducing errors when making code changes, it is advised to not use magic constants throughout code, but instead declare them once (as constant and commented) and use these constant variables instead. Following instances should therefore be changed accordingly:
    1. `CamelotV3Farm.sol#L292` : `-887272` and `887272` .
    2. `UniV3Farm.sol#L292` : `-887272` and `887272` .
    3. `Rewarder.sol#L351` : `18` .
5. Fixed `CamelotV2Farm.sol` : Consider caching `lpToken` , `token0` and `token1` for further gas-optimization.

Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
- **Informational** – The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
- **Undetermined** – The impact of the issue is uncertain.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **Acknowledged** – The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Files

- `255...dde ./contracts/Farm.sol`
- `df2...5d0 ./contracts/FarmDeployer.sol`
- `c33...56d ./contracts/FarmStorage.sol`
- `1af...6c1 ./contracts/FarmRegistry.sol`
- `3ca...4a0 ./contracts/interfaces/IRewarderFactory.sol`
- `32b...28f ./contracts/interfaces/IOracle.sol`
- `816...d8e ./contracts/interfaces/DataTypes.sol`
- `d87...025 ./contracts/interfaces/IFarmRegistry.sol`
- `9ca...913 ./contracts/interfaces/IFarm.sol`
- `dce...447 ./contracts/e20-farms/E20Farm.sol`
- `767...43f ./contracts/e20-farms/balancerV2/BalancerV2Farm.sol`
- `3b2...96f ./contracts/e20-farms/balancerV2/BalancerV2FarmDeployer.sol`
- `00c...51a ./contracts/e20-farms/balancerV2/interfaces/IBalancerV2Vault.sol`
- `799...1a5 ./contracts/e20-farms/uniswapV2/UniV2Farm.sol`
- `519...4c2 ./contracts/e20-farms/uniswapV2/UniV2FarmDeployer.sol`
- `7ca...1c4 ./contracts/e20-farms/uniswapV2/interfaces/IUniswapV2Factory.sol`
- `b83...70e ./contracts/utils/TokenUtils.sol`
- `07f...12f ./contracts/rewarder/LegacyFarmRewarder.sol`
- `d85...c85 ./contracts/rewarder/RewarderFactory.sol`
- `6a6...e71 ./contracts/rewarder/Rewarder.sol`
- `921...f05 ./contracts/rewarder/interfaces/ILegacyRewarderHelpers.sol`
- `21a...cb0 ./contracts/features/ExpirableFarm.sol`
- `2aa...e0e ./contracts/features/OperableDeposit.sol`
- `f5b...2fb ./contracts/e721-farms/E721Farm.sol`
- `5bf...302 ./contracts/e721-farms/camelotV3/CamelotV3Farm.sol`
- `137...bc7 ./contracts/e721-farms/camelotV3/CamelotV3FarmDeployer.sol`
- `597...bd7 ./contracts/e721-farms/camelotV3/interfaces/ICamelotV3.sol`
- `5b6...bd2 ./contracts/e721-farms/camelotV3/interfaces/ICamelotV3Utils.sol`
- `f3a...400 ./contracts/e721-farms/camelotV3/interfaces/ICamelotV3NonfungiblePositionManagerUtils.sol`
- `c25...5d8 ./contracts/e721-farms/camelotV2/CamelotV2FarmDeployer.sol`
- `a56...131 ./contracts/e721-farms/camelotV2/CamelotV2Farm.sol`

- `66f...49f ./contracts/e721-farms/camelotV2/interfaces/ICamelotV2.sol`
- `7d0...41b ./contracts/e721-farms/uniswapV3/UniV3FarmDeployer.sol`
- `eeb...891 ./contracts/e721-farms/uniswapV3/UniV3Farm.sol`
- `f98...67f ./contracts/e721-farms/uniswapV3/UniV3ActiveLiquidityFarm.sol`
- `a56...c02 ./contracts/e721-farms/uniswapV3/UniV3ActiveLiquidityDeployer.sol`
- `cc5...19e ./contracts/e721-farms/uniswapV3/interfaces/INonfungiblePositionManagerUtils.sol`
- `9df...1bd ./contracts/e721-farms/uniswapV3/interfaces/IUniswapV3.sol`
- `4c4...6e2 ./contracts/e721-farms/uniswapV3/interfaces/IUniswapV3Utils.sol`
- `389...ece ./contracts/e721-farms/uniswapV3/tests/UniswapV3Test.sol`
- `f2f...0df ./contracts/e721-farms/uniswapV3/tests/ISwapRouter.sol`

Tests

- `285...7b7 ./test/FarmRegistry.t.sol`
- `923...00b ./test/Farm.t.sol`
- `9d5...093 ./test/e20-farms/E20Farm.t.sol`
- `b95...166 ./test/e20-farms/balancerV2/BalancerV2Farm.t.sol`
- `6c1...e85 ./test/utils/UpgradeUtil.t.sol`
- `f8a...1d4 ./test/utils/TestNetworkConfig.t.sol`
- `96d...888 ./test/utils/BaseSetup.t.sol`
- `25d...46a ./test/utils/networkConfig/Arbitrum.t.sol`
- `c04...db5 ./test/utils/networkConfig/INetworkConfig.sol`
- `c70...c73 ./test/rewarder/LegacyFarmRewarder.t.sol`
- `b01...4fa ./test/rewarder/Rewarder.t.sol`
- `dbe...cb4 ./test/features/ExpirableFarm.t.sol`
- `957...f85 ./test/e721-farms/E721Farm.t.sol`
- `5b4...105 ./test/e721-farms/camelotV3/CamelotV3Farm.t.sol`
- `5bc...e25 ./test/e721-farms/camelotV2/CamelotV2Farm.t.sol`
- `cc0...cf2 ./test/e721-farms/uniswapv3/UniV3ActiveLiquidityFarm.t.sol`
- `4b2...8d9 ./test/e721-farms/uniswapv3/UniV3Farm.t.sol`
- `705...8b5 ./test/e721-farms/uniswapv3/sushiswap/SushiswapV3Farm.t.sol`
- `8f9...41f ./test/e721-farms/uniswapv3/uniswap/UniswapV3ActiveLiquidityFarm.t.sol`
- `7aa...470 ./test/e721-farms/uniswapv3/uniswap/UniswapV3Farm.t.sol`

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:
- Slither v0.10.0

Steps taken to run the tools:
1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither .`

Automated Analysis

Slither

We filtered the issues highlighted by Slither and consolidated the relevant issues in this report.

Test Suite Results

All tests included are passing.

Fix Review Update

All tests are still passing. The tests include unhappy and happy paths, as well as tests for the fixes implemented.

```
[⠊] Compiling...
[⠒] Compiling 115 files with Solc 0.8.24
[⠢] Solc 0.8.24 finished in 36.56s
Compiler run successful!

Ran 1 test for test/rewarder/RewarderFactory.t.sol:DeployRewarderTest
[PASS] test_deployRewarder() (gas: 192380)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 8.15s (326.81ms CPU time)

Ran 1 test for test/rewarder/RewarderFactory.t.sol:TestInitialization
[PASS] test_Init() (gas: 10716)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 1.47s (43.17µs CPU time)

Ran 3 tests for test/FarmRegistry.t.sol:RemoveFarmDeployerTest
[PASS] test_RemoveFarmDeployer_RevertWhen_invalidDeployerId() (gas: 234376)
[PASS] test_removeLastDeployer() (gas: 315894)
[PASS] test_removeMiddleDeployer() (gas: 315913)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 9.80s (372.29µs CPU time)

Ran 3 tests for test/FarmRegistry.t.sol:UpdateFeeParamsTest
[PASS] test_UpdateFeeParams_RevertWhen_InvalidAddress() (gas: 234737)
[PASS] test_UpdateFeeParams_RevertWhen_callerIsNotOwner() (gas: 232838)
[PASS] test_updateFeeParams() (gas: 257051)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 9.80s (374.71µs CPU time)

Ran 3 tests for test/FarmRegistry.t.sol:InitializeTest
[PASS] test_Initialize_RevertWhen_receiverIsZeroAddress() (gas: 66335)
[PASS] test_Initialize_RevertWhen_tokenIsZeroAddress() (gas: 66507)
[PASS] test_init(uint256,uint256) (runs: 256, μ: 180723, ~: 180812)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 9.93s (133.09ms CPU time)

Ran 3 tests for test/rewarder/RewarderFactory.t.sol:UpdateOracleTest
[PASS] test_revertWhen_CallerIsNotOwner() (gas: 18107)
[PASS] test_revertWhen_InvalidAddress() (gas: 13476)
[PASS] test_updateOracle() (gas: 26414)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 1.49s (152.08µs CPU time)

Ran 3 tests for test/FarmRegistry.t.sol:UpdatePrivilegeTest
[PASS] test_UpdatePrivilege_RevertWhen_PrivilegeSameAsDesired() (gas: 234113)
[PASS] test_UpdatePrivilege_RevertWhen_callerIsNotOwner() (gas: 232604)
[PASS] test_updatePrivilege() (gas: 261617)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 1.31s (225.13µs CPU time)

Ran 3 tests for test/FarmRegistry.t.sol:RegisterFarmDeployerTest
[PASS] test_RegisterFarmDeployer_RevertWhen_DeployerAddressIsZero() (gas: 160021)
[PASS] test_RegisterFarmDeployer_RevertWhen_DeployerIsAlreadyRegistered() (gas: 232103)
[PASS] test_registerFarmDeployer() (gas: 237867)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 1.31s (218.37µs CPU time)

Ran 4 tests for test/FarmRegistry.t.sol:RegisterFarmTest
[PASS] test_RegisterFarm_RevertWhen_DeployerNotRegistered() (gas: 168730)
[PASS] test_RevertWhen_FarmAlreadyRegistered() (gas: 309487)
[PASS] test_RevertWhen_InvalidAddress() (gas: 236515)
[PASS] test_registerFarm() (gas: 312064)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 1.18s (294.21µs CPU time)

Ran 3 tests for test/rewarder/RewarderFactory.t.sol:UpdateRewarderImplementationTest
[PASS] test_revertWhen_CallerIsNotOwner() (gas: 18129)
[PASS] test_revertWhen_InvalidAddress() (gas: 13498)
[PASS] test_updateRewarderImplementation() (gas: 26503)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 635.10ms (123.92µs CPU time)

Ran 2 tests for test/rewarder/Rewarder.t.sol:TestInitialization
[PASS] test_Init() (gas: 26337)
[PASS] test_Initialize_RevertWhen_camelotPairIsZero() (gas: 6822467)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 15.16s (1.47s CPU time)

Ran 3 tests for test/rewarder/Rewarder.t.sol:TestRecoverRewardFundsOfFarm
```

```
[PASS] test_Initialize_RevertWhen_camelotPairIsZero() (gas: 6822378)
[PASS] test_RevertWhen_CallerIsNotTheOwner() (gas: 28684)
[PASS] test_recoverRewardFundsOfFarm() (gas: 275449)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 4.20s (1.63s CPU time)

Ran 4 tests for test/rewarder/Rewarder.t.sol:TestRecoverERC20
[PASS] test_Initialize_RevertWhen_camelotPairIsZero() (gas: 6822378)
[PASS] test_RecoverERC20() (gas: 203580)
[PASS] test_RevertWhen_CallerIsNotOwner() (gas: 28647)
[PASS] test_RevertWhen_ZeroAmount() (gas: 34621)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 16.68s (2.99s CPU time)

Ran 10 tests for test/rewarder/Rewarder.t.sol:TestUpdateRewardConfig
[PASS] test_Initialize_RevertWhen_camelotPairIsZero() (gas: 6822390)
[PASS] test_RevertWhen_BaseAssetPriceFeedDoesntExist() (gas: 234393)
[PASS] test_RevertWhen_BaseTokenDoesNotExistInPoolToken() (gas: 156295)
[PASS] test_RevertWhen_BaseTokenIsRepeated() (gas: 258558)
[PASS] test_RevertWhen_BaseTokensAreMoreThanAssets() (gas: 362096)
[PASS] test_RevertWhen_FarmDoesntHaveRewardToken() (gas: 222849)
[PASS] test_RevertWhen_NonLockupRewardPer0() (gas: 236520)
[PASS] test_RevertWhen_NonLockupRewardPerMoreThanMax() (gas: 241265)
[PASS] test_UpdateRewardToken() (gas: 312522)
[PASS] test_UpdateRewardToken_BaseTokensHaveBothPoolAssets() (gas: 407998)
Suite result: ok. 10 passed; 0 failed; 0 skipped; finished in 1.83s (388.83ms CPU time)

Ran 3 tests for test/rewarder/Rewarder.t.sol:TestUpdateTokenManagerOfFarm
[PASS] test_Initialize_RevertWhen_camelotPairIsZero() (gas: 6822390)
[PASS] test_RevertWhen_CallerIsNotTheOwner() (gas: 29043)
[PASS] test_UpdateTokenManagerOfFarm() (gas: 57023)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 588.77ms (525.00µs CPU time)

Ran 5 tests for test/rewarder/Rewarder.t.sol:TestCalibrationRestriction
[PASS] test_Initialize_RevertWhen_camelotPairIsZero() (gas: 6822378)
[PASS] test_RevertWhen_CallerIsNotOwner() (gas: 30671)
[PASS] test_ToggleCalibrationRestriction() (gas: 50953)
[PASS] test_ToggleCalibrationRestriction_CalibrateWhenCalledByOwner() (gas: 294433)
[PASS] test_ToggleCalibrationRestriction_RemoveRestriction() (gas: 272148)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 541.84ms (1.86ms CPU time)

Ran 2 tests for test/rewarder/Rewarder.t.sol:GetTokenAmountsTest
[PASS] test_Initialize_RevertWhen_camelotPairIsZero() (gas: 6822390)
[PASS] test_getTokenAmounts() (gas: 1735938)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 22.50s (8.80s CPU time)

Ran 1 test for test/rewarder/Rewarder.t.sol:RewarderTest
[PASS] test_Initialize_RevertWhen_camelotPairIsZero() (gas: 6822378)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 321.02ms (422.58µs CPU time)

Ran 2 tests for test/rewarder/Rewarder.t.sol:TestFlow
[PASS] test_Initialize_RevertWhen_camelotPairIsZero() (gas: 6822390)
[PASS] test_temp() (gas: 1600817)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 2.83s (1.44s CPU time)

Ran 2 tests for test/rewarder/Rewarder.t.sol:TestRewardsEndTime
[PASS] test_Initialize_RevertWhen_camelotPairIsZero() (gas: 6822390)
[PASS] test_rewardsEndTime() (gas: 2783688)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 25.64s (11.95s CPU time)

Ran 8 tests for test/rewarder/Rewarder.t.sol:TestUpdateAPR
[PASS] test_Initialize_RevertWhen_camelotPairIsZero() (gas: 6822412)
[PASS] test_RevertWhen_updateAPR_CallerIsNotTheOwner() (gas: 48793)
[PASS] test_RevertWhen_updateAPR_NotConfigured() (gas: 70945)
[PASS] test_UpdateAPR() (gas: 1604358)
[PASS] test_UpdateAPR_CapRewardsWithBalance() (gas: 1545048)
[PASS] test_UpdateAPR_CapRewardsWithMaxRwdRate() (gas: 1515110)
[PASS] test_UpdateAPR_ForBaseTokenDecimalsMoreThanRwdTokenDecimals() (gas: 1605641)
[PASS] test_UpdateAPR_ForRwdTokenDecimalsMoreThanBaseTokenDecimals() (gas: 1603760)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 11.88s (7.77s CPU time)

Ran 100 tests for test/e20-farms/balancerV2/BalancerV2Farm.t.sol:BalancerV2FarmTest
[PASS] testFuzz_Deposit_Before_Farm_StartTime() (gas: 1545280)
[PASS] testFuzz_IncreaseDepositTest(bool,uint256) (runs: 256, µ: 1047718, ~: 1121040)
```

```
[PASS] testFuzz_Multicall(uint256) (runs: 256, μ: 172289, ~: 172399)
[PASS] testFuzz_Multicall_RevertWhen_AnyIndividualTestFail(uint256) (runs: 256, μ: 83010, ~: 83126)
[PASS] testFuzz_addRewards(bool,uint256) (runs: 256, μ: 417003, ~: 417031)
[PASS] testFuzz_extendFarmDuration(bool,uint256,uint256) (runs: 256, μ: 716468, ~: 679518)
[PASS] testFuzz_recoverERC20(bool,uint256) (runs: 256, μ: 230776, ~: 230817)
[PASS] testFuzz_setRewardRate(bool,uint256,uint256) (runs: 256, μ: 185336, ~: 216983)
[PASS] testFuzz_updateCoolDown_lockupFarm(uint256) (runs: 256, μ: 30837, ~: 30837)
[PASS] testFuzz_updateFarmStartTime(bool,uint256,uint256) (runs: 256, μ: 635386, ~: 598749)
[PASS] testFuzz_updateFarmStartTimeWithExpiry(bool,uint256,uint256) (runs: 256, μ: 639718, ~: 601240)
[PASS] testFuzz_updateFarmStartTime_end_time_noDelta(bool,uint256) (runs: 256, μ: 636603, ~: 598704)
[PASS] testFuzz_updateFarmStartTime_end_time_withDelta(bool,uint256) (runs: 256, μ: 643517, ~: 679971)
[PASS] testFuzz_updateFarmStartTime_end_time_withDelta_multiUpdate(bool) (runs: 256, μ: 666228, ~:
627929)
[PASS] testFuzz_withdraw_closedAndExpired() (gas: 2465367)
[PASS] testFuzz_withdraw_notClosedButExpired() (gas: 1813663)
[PASS] testMaths_updateSubscriptionForDecrease() (gas: 1698863)
[PASS] testMaths_updateSubscriptionForIncrease() (gas: 1701419)
[PASS] test_AddRewards_RevertWhen_InvalidRewardToken() (gas: 221146)
[PASS] test_AddRewards_RevertWhen_ZeroAmount() (gas: 39145)
[PASS] test_CannotWithdrawZeroAmount() (gas: 987521)
[PASS] test_ClaimRewards_RevertWhen_DepositDoesNotExist() (gas: 1415264)
[PASS] test_ClaimRewards_RevertWhen_FarmIsClosed() (gas: 1472808)
[PASS] test_CloseFarm_RevertWhen_FarmIsClosed() (gas: 144600)
[PASS] test_DecreaseDeposit_RevertWhen_LockupFarm_DecreaseDepositNotPermitted() (gas: 987794)
[PASS] test_DecreaseDeposit_RevertWhen_farmIsClosed() (gas: 992596)
[PASS] test_Deposit_RevertWhen_FarmIsClosed() (gas: 288548)
[PASS] test_Deposit_RevertWhen_FarmIsInactive() (gas: 247175)
[PASS] test_Deposit_RevertWhen_LockupFunctionalityIsDisabled() (gas: 208466)
[PASS] test_Deposit_RevertWhen_NoLiquidityInPosition() (gas: 192616)
[PASS] test_E20FarmDeposit() (gas: 930124)
[PASS] test_ExtendFarmDuration_RevertWhen_DurationExceeded() (gas: 98812)
[PASS] test_ExtendFarmDuration_RevertWhen_FarmNotYetStarted() (gas: 606529)
[PASS] test_ExtendFarmDuration_RevertWhen_InvalidExtension() (gas: 82826)
[PASS] test_ExtendFarmDuration_RevertWhen_farmClosed() (gas: 144671)
[PASS] test_ExtendFarmDuration_RevertWhen_farmExpired() (gas: 607793)
[PASS] test_FarmPauseSwitch_RevertWhen_FarmAlreadyInRequiredState() (gas: 24919)
[PASS] test_FarmPauseSwitch_RevertWhen_FarmIsClosed() (gas: 145881)
[PASS] test_GetDeposit_RevertWhen_DepositDoesNotExist() (gas: 24971)
[PASS] test_GetRewardBalance_RevertWhen_InvalidRewardToken() (gas: 995998)
[PASS] test_GetRewardFundInfo_RevertWhen_RewardFundDoesNotExist() (gas: 24593)
[PASS] test_GetRewardTokensTest() (gas: 33825)
[PASS] test_IncreaseDeposit_RevertWhen_FarmIsInactive() (gas: 1107919)
[PASS] test_IncreaseDeposit_RevertWhen_InvalidAmount() (gas: 1069276)
[PASS] test_IncreaseDeposit_RevertWhen_depositInCoolDown() (gas: 1131940)
[PASS] test_InsufficientLiquidity() (gas: 991196)
[PASS] test_Multicall_RevertWhen_CallInternalFunction() (gas: 21118)
[PASS] test_RecoverERC20_RevertWhen_CannotWithdrawFarmToken() (gas: 44557)
[PASS] test_RecoverERC20_RevertWhen_CannotWithdrawRewardToken() (gas: 49893)
[PASS] test_RecoverERC20_RevertWhen_CannotWithdrawZeroAmount() (gas: 58364)
[PASS] test_SetRewardRate_RevertWhen_FarmIsClosed() (gas: 151754)
[PASS] test_SetRewardRate_RevertWhen_InvalidRewardRatesLength() (gas: 118842)
[PASS] test_SetupFarm_MAX_NUM_REWARDS() (gas: 741966)
[PASS] test_SetupFarm_RevertWhen_InvalidFarmStartTime() (gas: 207016)
[PASS] test_SetupFarm_RevertWhen_InvalidRewardData() (gas: 351307)
[PASS] test_SetupFarm_RevertWhen_RewardAlreadyAdded() (gas: 507096)
[PASS] test_SubscriptionInfo_RevertWhen_SubscriptionDoesNotExist() (gas: 844153)
[PASS] test_UpdateCoolDownPeriod_RevertWhen_FarmIsClosed() (gas: 144369)
[PASS] test_UpdateCoolDownPeriod_RevertWhen_InvalidCooldownPeriod() (gas: 25412)
[PASS] test_UpdateCoolDownPeriod_RevertWhen_InvalidCooldownPeriod0() (gas: 25506)
[PASS] test_UpdateCoolDownPeriod_noLockupFarm() (gas: 25459)
[PASS] test_UpdateFarmStartTime_RevertWhen_FarmAlreadyStarted() (gas: 25460)
[PASS] test_UpdateFarmStartTime_RevertWhen_FarmIsClosed() (gas: 144481)
[PASS] test_UpdateFarmStartTime_RevertWhen_InvalidTime() (gas: 584062)
[PASS] test_UpdateRewardTokenData_RevertWhen_FarmIsClosed() (gas: 149817)
[PASS] test_UpdateRewardTokenData_RevertWhen_InvalidAddress() (gas: 32441)
[PASS] test_UpdateRewardTokenData_RevertWhen_NotTheTokenManager() (gas: 37341)
[PASS] test_Withdraw_RevertWhen_DepositInSameTs() (gas: 987339)
[PASS] test_Withdraw_RevertWhen_DepositIsInCooldown() (gas: 1032512)
[PASS] test_Withdraw_RevertWhen_PleaseInitiateCooldown() (gas: 987901)
[PASS] test_claimRewards() (gas: 2198959)
[PASS] test_claimRewards_max_rewards() (gas: 1100416)
[PASS] test_claimRewards_rwd_rate_0() (gas: 995270)
```

```
[PASS] test_closeFarm() (gas: 310965)
[PASS] test_decreaseDeposit_decreaseInSameTransactionAsDeposit() (gas: 987068)
[PASS] test_deposit() (gas: 910977)
[PASS] test_farmPause() (gas: 131913)
[PASS] test_getDeposit() (gas: 849188)
[PASS] test_getRewardFundInfo_LockupFarm() (gas: 39052)
[PASS] test_initiateCooldown_LockupFarm() (gas: 1382096)
[PASS] test_initiateCooldown_RevertWhen_DepositDoesNotExist() (gas: 990684)
[PASS] test_initiateCooldown_nonLockupFarm() (gas: 843256)
[PASS] test_nonLockupFarm() (gas: 1107572)
[PASS] test_recoverRewardFund_AfterAddRewards() (gas: 1798740)
[PASS] test_recoverRewardFund_WithDirectlySentFunds() (gas: 786269)
[PASS] test_recoverRewardFund_partially() (gas: 790647)
[PASS] test_revertWhen_decreaseDeposit_decreaseInSameTransactionAsIncrease() (gas: 1118808)
[PASS] test_revertWhen_withdraw_withdrawInSameTransactionAsIncrease() (gas: 1119099)
[PASS] test_rewardBalance() (gas: 1696715)
[PASS] test_subInfo() (gas: 1680619)
[PASS] test_updateFarmStartTime_RevertWhen_FarmHasExpired() (gas: 26947)
[PASS] test_updateFarmStartTime_revertWhen_CallerIsNotOwner() (gas: 30615)
[PASS] test_updateTknManager() (gas: 89757)
[PASS] test_withdraw() (gas: 2279904)
[PASS] test_withdraw_closed() (gas: 2460852)
[PASS] test_withdraw_firstDeposit_multipleDeposits() (gas: 8619668)
[PASS] test_withdraw_inBetweenDeposit_multipleDeposits() (gas: 8619598)
[PASS] test_withdraw_lastDeposit_multipleDeposits() (gas: 8619637)
[PASS] test_withdraw_paused() (gas: 2394780)
[PASS] test_zeroAmount() (gas: 987476)
Suite result: ok. 100 passed; 0 failed; 0 skipped; finished in 63.57s (65.31s CPU time)

Ran 111 tests for test/e721-farms/uniswapv3/uniswap/UniswapV3Farm.t.sol:UniswapV3FarmTest
[PASS] testFuzz_DecreaseDeposit(bool,uint256) (runs: 256, μ: 1889183, ~: 1875664)
[PASS] testFuzz_Deposit_Before_Farm_StartTime() (gas: 2539239)
[PASS] testFuzz_IncreaseDeposit(bool,uint256) (runs: 256, μ: 1959008, ~: 2036922)
[PASS] testFuzz_Multicall(uint256) (runs: 256, μ: 172565, ~: 172685)
[PASS] testFuzz_Multicall_RevertWhen_AnyIndividualTestFail(uint256) (runs: 256, μ: 83157, ~: 83323)
[PASS] testFuzz_addRewards(bool,uint256) (runs: 256, μ: 416952, ~: 417000)
[PASS] testFuzz_extendFarmDuration(bool,uint256,uint256) (runs: 256, μ: 808224, ~: 771570)
[PASS] testFuzz_recoverERC20(bool,uint256) (runs: 256, μ: 229079, ~: 229127)
[PASS] testFuzz_setRewardRate(bool,uint256,uint256) (runs: 256, μ: 182458, ~: 151478)
[PASS] testFuzz_updateCoolDown_lockupFarm(uint256) (runs: 256, μ: 30813, ~: 30835)
[PASS] testFuzz_updateFarmStartTime(bool,uint256,uint256) (runs: 256, μ: 729164, ~: 690679)
[PASS] testFuzz_updateFarmStartTimeWithExpiry(bool,uint256,uint256) (runs: 256, μ: 731953, ~: 693163)
[PASS] testFuzz_updateFarmStartTime_end_time_noDelta(bool,uint256) (runs: 256, μ: 726041, ~: 690608)
[PASS] testFuzz_updateFarmStartTime_end_time_withDelta(bool,uint256) (runs: 256, μ: 735401, ~: 771871)
[PASS] testFuzz_updateFarmStartTime_end_time_withDelta_multiUpdate(bool) (runs: 256, μ: 760112, ~:
799030)
[PASS] testFuzz_withdraw_closedAndExpired() (gas: 3404189)
[PASS] testFuzz_withdraw_notClosedButExpired() (gas: 2789140)
[PASS] test_AddRewards_RevertWhen_InvalidRewardToken() (gas: 221109)
[PASS] test_AddRewards_RevertWhen_ZeroAmount() (gas: 39046)
[PASS] test_ClaimRewards_RevertWhen_DepositDoesNotExist() (gas: 1957381)
[PASS] test_ClaimRewards_RevertWhen_FarmIsClosed() (gas: 2014993)
[PASS] test_ClaimUniswapFee_RevertWhen_DepositDoesNotExist_during_claimUniswapFee() (gas: 48301)
[PASS] test_ClaimUniswapFee_RevertWhen_FarmIsClosed() (gas: 152632)
[PASS] test_ClaimUniswapFee_RevertWhen_NoFeeToClaim() (gas: 1618264)
[PASS] test_CloseFarm_RevertWhen_FarmIsClosed() (gas: 144490)
[PASS] test_DecreaseDeposit_RevertWhen_CannotWithdrawZeroAmount() (gas: 1598552)
[PASS] test_DecreaseDeposit_RevertWhen_DecreaseDepositNotPermitted() (gas: 1598899)
[PASS] test_DecreaseDeposit_RevertWhen_DepositDoesNotExist() (gas: 49003)
[PASS] test_DecreaseDeposit_RevertWhen_FarmIsClosed() (gas: 1677612)
[PASS] test_Deposit_RevertWhen_FarmIsClosed() (gas: 908489)
[PASS] test_Deposit_RevertWhen_FarmIsInactive() (gas: 846439)
[PASS] test_Deposit_RevertWhen_LockupFunctionalityIsDisabled() (gas: 798085)
[PASS] test_Deposit_RevertWhen_NoLiquidityInPosition() (gas: 819044)
[PASS] test_ExtendFarmDuration_RevertWhen_DurationExceeded() (gas: 98877)
[PASS] test_ExtendFarmDuration_RevertWhen_FarmNotYetStarted() (gas: 698585)
[PASS] test_ExtendFarmDuration_RevertWhen_InvalidExtension() (gas: 82848)
[PASS] test_ExtendFarmDuration_RevertWhen_farmClosed() (gas: 144539)
[PASS] test_ExtendFarmDuration_RevertWhen_farmExpired() (gas: 699846)
[PASS] test_FarmPauseSwitch_RevertWhen_FarmAlreadyInRequiredState() (gas: 24940)
[PASS] test_FarmPauseSwitch_RevertWhen_FarmIsClosed() (gas: 145792)
[PASS] test_GetDeposit_RevertWhen_DepositDoesNotExist() (gas: 24733)
```

```
[PASS] test_GetRewardBalance_RevertWhen_InvalidRewardToken() (gas: 1606612)
[PASS] test_GetRewardFundInfo_RevertWhen_RewardFundDoesNotExist() (gas: 24525)
[PASS] test_GetRewardTokensTest() (gas: 33892)
[PASS] test_IncreaseDeposit_RevertWhen_DepositDoesNotExist() (gas: 44838)
[PASS] test_IncreaseDeposit_RevertWhen_DepositIsInCooldown() (gas: 1951806)
[PASS] test_IncreaseDeposit_RevertWhen_FarmIsInactive() (gas: 1927353)
[PASS] test_IncreaseDeposit_RevertWhen_InvalidAmount() (gas: 1597936)
[PASS] test_Initialize() (gas: 604112)
[PASS] test_Initialize_RevertWhen_InvalidTickRange() (gas: 141814)
[PASS] test_Initialize_RevertWhen_InvalidUniswapPoolConfig() (gas: 82451)
[PASS] test_Multicall_RevertWhen_CallInternalFunction() (gas: 21185)
[PASS] test_NFTDeposit() (gas: 1877414)
[PASS] test_NFTDeposit_RevertWhen_NoData() (gas: 791945)
[PASS] test_NFTDeposit_RevertWhen_UnauthorisedNFTContract() (gas: 14252)
[PASS] test_OnERC721Received_RevertWhen_IncorrectPoolToken() (gas: 811250)
[PASS] test_OnERC721Received_RevertWhen_IncorrectTickRange() (gas: 1861254)
[PASS] test_RecoverERC20_RevertWhen_CannotWithdrawRewardToken() (gas: 47726)
[PASS] test_RecoverERC20_RevertWhen_CannotWithdrawZeroAmount() (gas: 56307)
[PASS] test_SetRewardRate_RevertWhen_FarmIsClosed() (gas: 151785)
[PASS] test_SetRewardRate_RevertWhen_InvalidRewardRatesLength() (gas: 119055)
[PASS] test_SetupFarm_MAX_NUM_REWARDS() (gas: 833993)
[PASS] test_SetupFarm_RevertWhen_InvalidFarmStartTime() (gas: 274049)
[PASS] test_SetupFarm_RevertWhen_InvalidRewardData() (gas: 418507)
[PASS] test_SetupFarm_RevertWhen_RewardAlreadyAdded() (gas: 574263)
[PASS] test_SubscriptionInfo_RevertWhen_SubscriptionDoesNotExist() (gas: 1454809)
[PASS] test_UpdateCoolDownPeriod_RevertWhen_FarmIsClosed() (gas: 144280)
[PASS] test_UpdateCoolDownPeriod_RevertWhen_InvalidCooldownPeriod() (gas: 25541)
[PASS] test_UpdateCoolDownPeriod_RevertWhen_InvalidCooldownPeriod0() (gas: 25484)
[PASS] test_UpdateCoolDownPeriod_noLockupFarm() (gas: 25458)
[PASS] test_UpdateFarmStartTime_RevertWhen_FarmAlreadyStarted() (gas: 25517)
[PASS] test_UpdateFarmStartTime_RevertWhen_FarmIsClosed() (gas: 144428)
[PASS] test_UpdateFarmStartTime_RevertWhen_InvalidTime() (gas: 676076)
[PASS] test_UpdateRewardTokenData_RevertWhen_FarmIsClosed() (gas: 149818)
[PASS] test_UpdateRewardTokenData_RevertWhen_InvalidAddress() (gas: 32441)
[PASS] test_UpdateRewardTokenData_RevertWhen_NotTheTokenManager() (gas: 37364)
[PASS] test_Withdraw() (gas: 1906546)
[PASS] test_Withdraw_RevertWhen_DepositDoesNotExist_during_withdraw() (gas: 44009)
[PASS] test_Withdraw_RevertWhen_DepositInSameTs() (gas: 1597804)
[PASS] test_Withdraw_RevertWhen_DepositIsInCooldown() (gas: 1642889)
[PASS] test_Withdraw_RevertWhen_PleaseInitiateCooldown() (gas: 1598296)
[PASS] test_Withdraw_closed() (gas: 1790648)
[PASS] test_Withdraw_closedAndExpired() (gas: 1792066)
[PASS] test_Withdraw_notClosedButExpired() (gas: 1712552)
[PASS] test_Withdraw_paused() (gas: 1730970)
[PASS] test_claimRewards() (gas: 3096722)
[PASS] test_claimRewards_max_rewards() (gas: 1674413)
[PASS] test_claimRewards_rwd_rate_0() (gas: 1586785)
[PASS] test_claimUniswapFee() (gas: 2016266)
[PASS] test_closeFarm() (gas: 311066)
[PASS] test_deposit() (gas: 1838191)
[PASS] test_farmPause() (gas: 131957)
[PASS] test_getDeposit() (gas: 1459674)
[PASS] test_getRewardFundInfo_LockupFarm() (gas: 39095)
[PASS] test_initiateCooldown_LockupFarm() (gas: 1956044)
[PASS] test_initiateCooldown_RevertWhen_DepositDoesNotExist() (gas: 1601156)
[PASS] test_initiateCooldown_nonLockupFarm() (gas: 1453813)
[PASS] test_recoverRewardFund_AfterAddRewards() (gas: 2727463)
[PASS] test_recoverRewardFund_WithDirectlySentFunds() (gas: 786371)
[PASS] test_recoverRewardFund_partially() (gas: 790783)
[PASS] test_rewardBalance() (gas: 2625362)
[PASS] test_subInfo() (gas: 2609386)
[PASS] test_updateFarmStartTime_RevertWhen_FarmHasExpired() (gas: 26971)
[PASS] test_updateFarmStartTime_revertWhen_CallerIsNotOwner() (gas: 30563)
[PASS] test_updateTknManager() (gas: 89869)
[PASS] test_withdraw() (gas: 3223628)
[PASS] test_withdraw_closed() (gas: 3404636)
[PASS] test_withdraw_firstDeposit_multipleDeposits() (gas: 16097693)
[PASS] test_withdraw_inBetweenDeposit_multipleDeposits() (gas: 16097640)
[PASS] test_withdraw_lastDeposit_multipleDeposits() (gas: 16097679)
[PASS] test_withdraw_paused() (gas: 3338775)
Suite result: ok. 111 passed; 0 failed; 0 skipped; finished in 43.83s (43.25s CPU time)
```

```
Ran 110 tests for test/e721-farms/camelotV2/CamelotV2Farm.t.sol:DemeterCamelotFarmInheritTest
[PASS] testFuzz_Deposit_Before_Farm_StartTime() (gas: 2881236)
[PASS] testFuzz_Multicall(uint256) (runs: 256, μ: 172319, ~: 172487)
[PASS] testFuzz_Multicall_RevertWhen_AnyIndividualTestFail(uint256) (runs: 256, μ: 83183, ~: 83278)
[PASS] testFuzz_addRewards(bool,uint256) (runs: 256, μ: 417143, ~: 417160)
[PASS] testFuzz_extendFarmDuration(bool,uint256,uint256) (runs: 256, μ: 859885, ~: 821398)
[PASS] testFuzz_recoverERC20(bool,uint256) (runs: 256, μ: 229053, ~: 229097)
[PASS] testFuzz_setRewardRate(bool,uint256,uint256) (runs: 256, μ: 186548, ~: 217149)
[PASS] testFuzz_updateCoolDown_lockupFarm(uint256) (runs: 256, μ: 30748, ~: 30770)
[PASS] testFuzz_updateFarmStartTime(bool,uint256,uint256) (runs: 256, μ: 783244, ~: 819106)
[PASS] testFuzz_updateFarmStartTimeWithExpiry(bool,uint256,uint256) (runs: 256, μ: 785014, ~: 821531)
[PASS] testFuzz_updateFarmStartTime_end_time_noDelta(bool,uint256) (runs: 256, μ: 781263, ~: 818974)
[PASS] testFuzz_updateFarmStartTime_end_time_withDelta(bool,uint256) (runs: 256, μ: 780067, ~: 743414)
[PASS] testFuzz_updateFarmStartTime_end_time_withDelta_multiUpdate(bool) (runs: 256, μ: 810212, ~:
849439)
[PASS] testFuzz_withdraw_closedAndExpired() (gas: 3751678)
[PASS] testFuzz_withdraw_notClosedButExpired() (gas: 3104734)
[PASS] test_AddRewards_RevertWhen_InvalidRewardToken() (gas: 221226)
[PASS] test_AddRewards_RevertWhen_ZeroAmount() (gas: 39090)
[PASS] test_AmountA_noLockupFarm() (gas: 2184443)
[PASS] test_ClaimRewards_RevertWhen_DepositDoesNotExist() (gas: 2078577)
[PASS] test_ClaimRewards_RevertWhen_FarmIsClosed() (gas: 2136154)
[PASS] test_CloseFarm_RevertWhen_FarmIsClosed() (gas: 144556)
[PASS] test_DecreaseDeposit_RevertWhen_FarmIsClosed() (gas: 1926196)
[PASS] test_DecreaseDeposit_RevertWhen_InvalidDeposit() (gas: 1777584)
[PASS] test_DecreaseDeposit_RevertWhen_ZeroAmount() (gas: 1771652)
[PASS] test_DecreaseDeposit_lockupFarm() (gas: 1719108)
[PASS] test_DecreaseDeposit_nonLockupFarm() (gas: 1926376)
[PASS] test_Deposit_RevertWhen_FarmIsClosed() (gas: 1013750)
[PASS] test_Deposit_RevertWhen_FarmIsInactive() (gas: 964093)
[PASS] test_Deposit_RevertWhen_LockupFunctionalityIsDisabled() (gas: 925348)
[PASS] test_Deposit_RevertWhen_NoLiquidityInPosition() (gas: 926070)
[PASS] test_ExtendFarmDuration_RevertWhen_DurationExceeded() (gas: 98724)
[PASS] test_ExtendFarmDuration_RevertWhen_FarmNotYetStarted() (gas: 748549)
[PASS] test_ExtendFarmDuration_RevertWhen_InvalidExtension() (gas: 82760)
[PASS] test_ExtendFarmDuration_RevertWhen_farmClosed() (gas: 144650)
[PASS] test_ExtendFarmDuration_RevertWhen_farmExpired() (gas: 749749)
[PASS] test_FarmPauseSwitch_RevertWhen_FarmAlreadyInRequiredState() (gas: 24898)
[PASS] test_FarmPauseSwitch_RevertWhen_FarmIsClosed() (gas: 145904)
[PASS] test_GetDeposit_RevertWhen_DepositDoesNotExist() (gas: 24908)
[PASS] test_GetRewardBalance_RevertWhen_InvalidRewardToken() (gas: 1711904)
[PASS] test_GetRewardFundInfo_RevertWhen_RewardFundDoesNotExist() (gas: 24637)
[PASS] test_GetRewardTokensTest() (gas: 33913)
[PASS] test_IncreaseDeposit_RevertWhen_FarmIsInactive() (gas: 2020336)
[PASS] test_IncreaseDeposit_RevertWhen_InvalidAmount() (gas: 1559028)
[PASS] test_IncreaseDeposit_RevertWhen_InvalidDeposit() (gas: 1773876)
[PASS] test_IncreaseDeposit_RevertWhen_depositInCoolDown() (gas: 2396436)
[PASS] test_IncreaseDeposit_lockupFarm() (gas: 2448312)
[PASS] test_IncreaseDeposit_noLockupFarm() (gas: 2165088)
[PASS] test_Initialize_RevertWhen_camelotPairIsZero() (gas: 6822456)
[PASS] test_Multicall_RevertWhen_CallInternalFunction() (gas: 21075)
[PASS] test_NFTDeposit() (gas: 2152448)
[PASS] test_NFTDeposit_RevertWhen_NoData() (gas: 916044)
[PASS] test_NFTDeposit_RevertWhen_UnauthorisedNFTContract() (gas: 14298)
[PASS] test_RecoverERC20_RevertWhen_CannotWithdrawRewardToken() (gas: 47749)
[PASS] test_RecoverERC20_RevertWhen_CannotWithdrawZeroAmount() (gas: 56197)
[PASS] test_SetRewardRate_RevertWhen_FarmIsClosed() (gas: 149061)
[PASS] test_SetRewardRate_RevertWhen_InvalidRewardRatesLength() (gas: 119106)
[PASS] test_SetupFarm_MAX_NUM_REWARDS() (gas: 889681)
[PASS] test_SetupFarm_RevertWhen_InvalidFarmStartTime() (gas: 313514)
[PASS] test_SetupFarm_RevertWhen_InvalidRewardData() (gas: 458105)
[PASS] test_SetupFarm_RevertWhen_RewardAlreadyAdded() (gas: 613641)
[PASS] test_SubscriptionInfo_RevertWhen_SubscriptionDoesNotExist() (gas: 1560085)
[PASS] test_UpdateCoolDownPeriod_RevertWhen_FarmIsClosed() (gas: 144412)
[PASS] test_UpdateCoolDownPeriod_RevertWhen_InvalidCooldownPeriod() (gas: 25453)
[PASS] test_UpdateCoolDownPeriod_RevertWhen_InvalidCooldownPeriod0() (gas: 25527)
[PASS] test_UpdateCoolDownPeriod_noLockupFarm() (gas: 25370)
[PASS] test_UpdateFarmStartTime_RevertWhen_FarmAlreadyStarted() (gas: 25526)
[PASS] test_UpdateFarmStartTime_RevertWhen_FarmIsClosed() (gas: 144503)
[PASS] test_UpdateFarmStartTime_RevertWhen_InvalidTime() (gas: 726149)
[PASS] test_UpdateRewardTokenData_RevertWhen_FarmIsClosed() (gas: 149993)
[PASS] test_UpdateRewardTokenData_RevertWhen_InvalidAddress() (gas: 32595)
```

```
[PASS] test_UpdateRewardTokenData_RevertWhen_NotTheTokenManager() (gas: 37385)
[PASS] test_Withdraw() (gas: 2029000)
[PASS] test_Withdraw_RevertWhen_DepositDoesNotExist_during_withdraw() (gas: 43878)
[PASS] test_Withdraw_RevertWhen_DepositInSameTs() (gas: 1702992)
[PASS] test_Withdraw_RevertWhen_DepositIsInCooldown() (gas: 1748112)
[PASS] test_Withdraw_RevertWhen_PleaseInitiateCooldown() (gas: 1703520)
[PASS] test_Withdraw_closed() (gas: 1897197)
[PASS] test_Withdraw_closedAndExpired() (gas: 1898531)
[PASS] test_Withdraw_notClosedButExpired() (gas: 1819024)
[PASS] test_Withdraw_paused() (gas: 1837466)
[PASS] test_claimPoolRewards_RevertWhen_FarmIsClosed() (gas: 1721724)
[PASS] test_claimPoolRewards_RevertWhen_InvalidDeposit() (gas: 1560120)
[PASS] test_claimPoolRewards_nonLockupFarm() (gas: 1573056)
[PASS] test_claimRewards() (gas: 3439112)
[PASS] test_claimRewards_max_rewards() (gas: 1794943)
[PASS] test_claimRewards_rwd_rate_0() (gas: 1694868)
[PASS] test_closeFarm() (gas: 311111)
[PASS] test_deposit() (gas: 2130276)
[PASS] test_farmPause() (gas: 131959)
[PASS] test_getDeposit() (gas: 1565003)
[PASS] test_getRewardFundInfo_LockupFarm() (gas: 39074)
[PASS] test_initiateCooldown_LockupFarm() (gas: 2077233)
[PASS] test_initiateCooldown_RevertWhen_DepositDoesNotExist() (gas: 1706486)
[PASS] test_initiateCooldown_nonLockupFarm() (gas: 1559110)
[PASS] test_onNFTHarvest() (gas: 19250)
[PASS] test_onNFTHarvest_RevertWhen_NotAllowed() (gas: 14169)
[PASS] test_recoverRewardFund_AfterAddRewards() (gas: 3041635)
[PASS] test_recoverRewardFund_WithDirectlySentFunds() (gas: 786464)
[PASS] test_recoverRewardFund_partially() (gas: 790928)
[PASS] test_rewardBalance() (gas: 2939364)
[PASS] test_subInfo() (gas: 2923369)
[PASS] test_updateFarmStartTime_RevertWhen_FarmHasExpired() (gas: 26970)
[PASS] test_updateFarmStartTime_revertWhen_CallerIsNotOwner() (gas: 30681)
[PASS] test_updateTknManager() (gas: 90109)
[PASS] test_withdraw() (gas: 3566449)
[PASS] test_withdraw_closed() (gas: 3748862)
[PASS] test_withdraw_firstDeposit_multipleDeposits() (gas: 20187528)
[PASS] test_withdraw_inBetweenDeposit_multipleDeposits() (gas: 20187546)
[PASS] test_withdraw_lastDeposit_multipleDeposits() (gas: 20187585)
[PASS] test_withdraw_paused() (gas: 3682845)
Suite result: ok. 110 passed; 0 failed; 0 skipped; finished in 73.52s (163.02s CPU time)

Ran 115 tests for test/e721-
farms/uniswapv3/uniswap/UniswapV3ActiveLiquidityFarm.t.sol:UniswapV3ActiveLiquidityFarmTest
[PASS] testFuzz_DecreaseDeposit(bool,uint256) (runs: 256, μ: 1629146, ~: 1631627)
[PASS] testFuzz_Deposit_Before_Farm_StartTime() (gas: 2458496)
[PASS] testFuzz_IncreaseDeposit(bool,uint256) (runs: 256, μ: 1863253, ~: 1783396)
[PASS] testFuzz_Multicall(uint256) (runs: 256, μ: 197711, ~: 197790)
[PASS] testFuzz_Multicall_RevertWhen_AnyIndividualTestFail(uint256) (runs: 256, μ: 86002, ~: 86141)
[PASS] testFuzz_addRewards(bool,uint256) (runs: 256, μ: 440015, ~: 440060)
[PASS] testFuzz_extendFarmDuration(bool,uint256,uint256) (runs: 256, μ: 814936, ~: 852640)
[PASS] testFuzz_recoverERC20(bool,uint256) (runs: 256, μ: 231310, ~: 231348)
[PASS] testFuzz_setRewardRate(bool,uint256,uint256) (runs: 256, μ: 210444, ~: 177668)
[PASS] testFuzz_updateCoolDown_lockupFarm(uint256) (runs: 256, μ: 33613, ~: 33613)
[PASS] testFuzz_updateFarmStartTime(bool,uint256,uint256) (runs: 256, μ: 732584, ~: 693499)
[PASS] testFuzz_updateFarmStartTimeWithExpiry(bool,uint256,uint256) (runs: 256, μ: 733815, ~: 695939)
[PASS] testFuzz_updateFarmStartTime_end_time_noDelta(bool,uint256) (runs: 256, μ: 730992, ~: 693384)
[PASS] testFuzz_updateFarmStartTime_end_time_withDelta(bool,uint256) (runs: 256, μ: 738489, ~: 774625)
[PASS] testFuzz_updateFarmStartTime_end_time_withDelta_multiUpdate(bool) (runs: 256, μ: 763380, ~:
802298)
[PASS] testFuzz_withdraw_closedAndExpired() (gas: 2912482)
[PASS] testFuzz_withdraw_notClosedButExpired() (gas: 2727231)
[PASS] test_AddRewards_RevertWhen_InvalidRewardToken() (gas: 223420)
[PASS] test_AddRewards_RevertWhen_ZeroAmount() (gas: 41824)
[PASS] test_ClaimRewards_RevertWhen_DepositDoesNotExist() (gas: 1662154)
[PASS] test_ClaimRewards_RevertWhen_FarmIsClosed() (gas: 1733445)
[PASS] test_ClaimUniswapFee_RevertWhen_DepositDoesNotExist_during_claimUniswapFee() (gas: 48324)
[PASS] test_ClaimUniswapFee_RevertWhen_FarmIsClosed() (gas: 177893)
[PASS] test_ClaimUniswapFee_RevertWhen_NoFeeToClaim() (gas: 1528059)
[PASS] test_CloseFarm_RevertWhen_FarmIsClosed() (gas: 169726)
[PASS] test_ComputeRewards_For_ActiveLiquidity() (gas: 1647848)
[PASS] test_DecreaseDeposit_RevertWhen_CannotWithdrawZeroAmount() (gas: 1503830)
```

```
[PASS] test_DecreaseDeposit_RevertWhen_DecreaseDepositNotPermitted() (gas: 1504177)
[PASS] test_DecreaseDeposit_RevertWhen_DepositDoesNotExist() (gas: 49003)
[PASS] test_DecreaseDeposit_RevertWhen_FarmIsClosed() (gas: 1585857)
[PASS] test_Deposit_RevertWhen_FarmIsClosed() (gas: 760206)
[PASS] test_Deposit_RevertWhen_FarmIsInactive() (gas: 698289)
[PASS] test_Deposit_RevertWhen_LockupFunctionalityIsDisabled() (gas: 644231)
[PASS] test_Deposit_RevertWhen_NoLiquidityInPosition() (gas: 702539)
[PASS] test_ExtendFarmDuration_RevertWhen_DurationExceeded() (gas: 98855)
[PASS] test_ExtendFarmDuration_RevertWhen_FarmNotYetStarted() (gas: 701363)
[PASS] test_ExtendFarmDuration_RevertWhen_InvalidExtension() (gas: 85648)
[PASS] test_ExtendFarmDuration_RevertWhen_farmClosed() (gas: 169709)
[PASS] test_ExtendFarmDuration_RevertWhen_farmExpired() (gas: 702602)
[PASS] test_FarmPauseSwitch_RevertWhen_FarmAlreadyInRequiredState() (gas: 38468)
[PASS] test_FarmPauseSwitch_RevertWhen_FarmIsClosed() (gas: 173235)
[PASS] test_GetDeposit_RevertWhen_DepositDoesNotExist() (gas: 24756)
[PASS] test_GetRewardBalance_RevertWhen_InvalidRewardToken() (gas: 1511892)
[PASS] test_GetRewardFundInfo_RevertWhen_RewardFundDoesNotExist() (gas: 24599)
[PASS] test_GetRewardTokensTest() (gas: 33937)
[PASS] test_IncreaseDeposit_RevertWhen_DepositDoesNotExist() (gas: 44860)
[PASS] test_IncreaseDeposit_RevertWhen_DepositIsInCooldown() (gas: 1865934)
[PASS] test_IncreaseDeposit_RevertWhen_FarmIsInactive() (gas: 1837910)
[PASS] test_IncreaseDeposit_RevertWhen_InvalidAmount() (gas: 1503215)
[PASS] test_Initialize() (gas: 604112)
[PASS] test_Initialize_RevertWhen_InvalidTickRange() (gas: 141836)
[PASS] test_Initialize_RevertWhen_InvalidUniswapPoolConfig() (gas: 82496)
[PASS] test_IsFarmActive_When_InactiveLiquidity() (gas: 62435)
[PASS] test_Multicall_RevertWhen_CallInternalFunction() (gas: 23963)
[PASS] test_NFTDeposit() (gas: 1741092)
[PASS] test_NFTDeposit_RevertWhen_NoData() (gas: 669298)
[PASS] test_NFTDeposit_RevertWhen_UnauthorisedNFTContract() (gas: 14252)
[PASS] test_OnERC721Received_RevertWhen_IncorrectPoolToken() (gas: 811268)
[PASS] test_OnERC721Received_RevertWhen_IncorrectTickRange() (gas: 1764447)
[PASS] test_RecoverERC20_RevertWhen_CannotWithdrawRewardToken() (gas: 50504)
[PASS] test_RecoverERC20_RevertWhen_CannotWithdrawZeroAmount() (gas: 58997)
[PASS] test_SetRewardRate_RevertWhen_FarmIsClosed() (gas: 176955)
[PASS] test_SetRewardRate_RevertWhen_InvalidRewardRatesLength() (gas: 166795)
[PASS] test_SetupFarm_MAX_NUM_REWARDS() (gas: 836771)
[PASS] test_SetupFarm_RevertWhen_InvalidFarmStartTime() (gas: 276872)
[PASS] test_SetupFarm_RevertWhen_InvalidRewardData() (gas: 421285)
[PASS] test_SetupFarm_RevertWhen_RewardAlreadyAdded() (gas: 577041)
[PASS] test_SubscriptionInfo_RevertWhen_SubscriptionDoesNotExist() (gas: 1357800)
[PASS] test_UpdateCoolDownPeriod_RevertWhen_FarmIsClosed() (gas: 169450)
[PASS] test_UpdateCoolDownPeriod_RevertWhen_InvalidCooldownPeriod() (gas: 28297)
[PASS] test_UpdateCoolDownPeriod_RevertWhen_InvalidCooldownPeriod0() (gas: 28262)
[PASS] test_UpdateCoolDownPeriod_noLockupFarm() (gas: 28214)
[PASS] test_UpdateFarmStartTime_RevertWhen_FarmAlreadyStarted() (gas: 28403)
[PASS] test_UpdateFarmStartTime_RevertWhen_FarmIsClosed() (gas: 169662)
[PASS] test_UpdateFarmStartTime_RevertWhen_InvalidTime() (gas: 678918)
[PASS] test_UpdateRewardTokenData_RevertWhen_FarmIsClosed() (gas: 175054)
[PASS] test_UpdateRewardTokenData_RevertWhen_InvalidAddress() (gas: 35307)
[PASS] test_UpdateRewardTokenData_RevertWhen_NotTheTokenManager() (gas: 37430)
[PASS] test_Withdraw() (gas: 1642854)
[PASS] test_Withdraw_For_ActiveLiquidity(uint256) (runs: 256, μ: 1490405, ~: 1491238)
[PASS] test_Withdraw_RevertWhen_DepositDoesNotExist_during_withdraw() (gas: 43987)
[PASS] test_Withdraw_RevertWhen_DepositInSameTs() (gas: 1504883)
[PASS] test_Withdraw_RevertWhen_DepositIsInCooldown() (gas: 1554780)
[PASS] test_Withdraw_RevertWhen_PleaseInitiateCooldown() (gas: 1505374)
[PASS] test_Withdraw_When_InactiveLiquidity() (gas: 1623853)
[PASS] test_Withdraw_closed() (gas: 1701862)
[PASS] test_Withdraw_closedAndExpired() (gas: 1705686)
[PASS] test_Withdraw_notClosedButExpired() (gas: 1623083)
[PASS] test_Withdraw_paused() (gas: 1642247)
[PASS] test_claimRewards() (gas: 2903412)
[PASS] test_claimRewards_max_rewards() (gas: 1408873)
[PASS] test_claimRewards_rwd_rate_0() (gas: 1410330)
[PASS] test_claimUniswapFee() (gas: 1923804)
[PASS] test_closeFarm() (gas: 346041)
[PASS] test_deposit() (gas: 1722516)
[PASS] test_farmPause() (gas: 171653)
[PASS] test_getDeposit() (gas: 1362768)
[PASS] test_getRewardFundInfo_LockupFarm() (gas: 39169)
[PASS] test_initiateCooldown_LockupFarm() (gas: 1595664)
```

```
[PASS] test_initiateCooldown_RevertWhen_DepositDoesNotExist() (gas: 1508235)
[PASS] test_initiateCooldown_nonLockupFarm() (gas: 1358585)
[PASS] test_recoverRewardFund_AfterAddRewards() (gas: 2667174)
[PASS] test_recoverRewardFund_WithDirectlySentFunds() (gas: 820534)
[PASS] test_recoverRewardFund_partially() (gas: 824964)
[PASS] test_rewardBalance() (gas: 2552813)
[PASS] test_subInfo() (gas: 2537085)
[PASS] test_updateFarmStartTime_RevertWhen_FarmHasExpired() (gas: 29791)
[PASS] test_updateFarmStartTime_revertWhen_CallerIsNotOwner() (gas: 30627)
[PASS] test_updateTknManager() (gas: 92933)
[PASS] test_withdraw() (gas: 3032350)
[PASS] test_withdraw_closed() (gas: 3113786)
[PASS] test_withdraw_firstDeposit_multipleDeposits() (gas: 16103764)
[PASS] test_withdraw_inBetweenDeposit_multipleDeposits() (gas: 16103711)
[PASS] test_withdraw_lastDeposit_multipleDeposits() (gas: 16103732)
[PASS] test_withdraw_paused() (gas: 3048404)
Suite result: ok. 115 passed; 0 failed; 0 skipped; finished in 56.58s (58.42s CPU time)

Ran 114 tests for test/e721-farms/camelotV3/CamelotV3Farm.t.sol:CamelotV3FarmInheritTest
[PASS] testFuzz_DecreaseDeposit(bool,uint256) (runs: 256, μ: 2079620, ~: 2066416)
[PASS] testFuzz_DecreaseDeposit_tickSpacingChanged(bool,uint256,int24) (runs: 256, μ: 2097669, ~:
2079808)
[PASS] testFuzz_Deposit_Before_Farm_StartTime() (gas: 2720096)
[PASS] testFuzz_IncreaseDeposit(bool,uint256) (runs: 256, μ: 2049628, ~: 2125644)
[PASS] testFuzz_Multicall(uint256) (runs: 256, μ: 172533, ~: 172665)
[PASS] testFuzz_Multicall_RevertWhen_AnyIndividualTestFail(uint256) (runs: 256, μ: 83188, ~: 83326)
[PASS] testFuzz_addRewards(bool,uint256) (runs: 256, μ: 416910, ~: 416945)
[PASS] testFuzz_claimCamelotFee_tickSpacingChanged(int24) (runs: 256, μ: 2912935, ~: 2912936)
[PASS] testFuzz_extendFarmDuration(bool,uint256,uint256) (runs: 256, μ: 814793, ~: 852193)
[PASS] testFuzz_recoverERC20(bool,uint256) (runs: 256, μ: 229062, ~: 229096)
[PASS] testFuzz_setRewardRate(bool,uint256,uint256) (runs: 256, μ: 183937, ~: 151394)
[PASS] testFuzz_updateCoolDown_lockupFarm(uint256) (runs: 256, μ: 30770, ~: 30770)
[PASS] testFuzz_updateFarmStartTime(bool,uint256,uint256) (runs: 256, μ: 735035, ~: 771542)
[PASS] testFuzz_updateFarmStartTimeWithExpiry(bool,uint256,uint256) (runs: 256, μ: 735264, ~: 773250)
[PASS] testFuzz_updateFarmStartTime_end_time_noDelta(bool,uint256) (runs: 256, μ: 732132, ~: 693003)
[PASS] testFuzz_updateFarmStartTime_end_time_withDelta(bool,uint256) (runs: 256, μ: 735341, ~: 735098)
[PASS] testFuzz_updateFarmStartTime_end_time_withDelta_multiUpdate(bool) (runs: 256, μ: 761006, ~:
722398)
[PASS] testFuzz_withdraw_closedAndExpired() (gas: 3561866)
[PASS] testFuzz_withdraw_notClosedButExpired() (gas: 2915099)
[PASS] test_AddRewards_RevertWhen_InvalidRewardToken() (gas: 221163)
[PASS] test_AddRewards_RevertWhen_ZeroAmount() (gas: 39133)
[PASS] test_ClaimCamelotFee_RevertWhen_DepositDoesNotExist_during_claimCamelotFee() (gas: 48399)
[PASS] test_ClaimCamelotFee_RevertWhen_FarmIsClosed() (gas: 152732)
[PASS] test_ClaimCamelotFee_RevertWhen_NoFeeToClaim() (gas: 1723857)
[PASS] test_ClaimRewards_RevertWhen_DepositDoesNotExist() (gas: 2080354)
[PASS] test_ClaimRewards_RevertWhen_FarmIsClosed() (gas: 2137966)
[PASS] test_CloseFarm_RevertWhen_FarmIsClosed() (gas: 144490)
[PASS] test_DecreaseDeposit_RevertWhen_CannotWithdrawZeroAmount() (gas: 1705551)
[PASS] test_DecreaseDeposit_RevertWhen_DecreaseDepositNotPermitted() (gas: 1705933)
[PASS] test_DecreaseDeposit_RevertWhen_DepositDoesNotExist() (gas: 49025)
[PASS] test_DecreaseDeposit_RevertWhen_FarmIsClosed() (gas: 1784736)
[PASS] test_Deposit_RevertWhen_FarmIsClosed() (gas: 992960)
[PASS] test_Deposit_RevertWhen_FarmIsInactive() (gas: 930888)
[PASS] test_Deposit_RevertWhen_LockupFunctionalityIsDisabled() (gas: 882512)
[PASS] test_Deposit_RevertWhen_NoLiquidityInPosition() (gas: 927228)
[PASS] test_ExtendFarmDuration_RevertWhen_DurationExceeded() (gas: 98768)
[PASS] test_ExtendFarmDuration_RevertWhen_FarmNotYetStarted() (gas: 700934)
[PASS] test_ExtendFarmDuration_RevertWhen_InvalidExtension() (gas: 82804)
[PASS] test_ExtendFarmDuration_RevertWhen_farmClosed() (gas: 144583)
[PASS] test_ExtendFarmDuration_RevertWhen_farmExpired() (gas: 702174)
[PASS] test_FarmPauseSwitch_RevertWhen_FarmAlreadyInRequiredState() (gas: 25052)
[PASS] test_FarmPauseSwitch_RevertWhen_FarmIsClosed() (gas: 145948)
[PASS] test_GetDeposit_RevertWhen_DepositDoesNotExist() (gas: 24862)
[PASS] test_GetRewardBalance_RevertWhen_InvalidRewardToken() (gas: 1713735)
[PASS] test_GetRewardFundInfo_RevertWhen_RewardFundDoesNotExist() (gas: 24481)
[PASS] test_GetRewardTokensTest() (gas: 33939)
[PASS] test_IncreaseDeposit_RevertWhen_DepositDoesNotExist() (gas: 44838)
[PASS] test_IncreaseDeposit_RevertWhen_DepositIsInCooldown() (gas: 2062254)
[PASS] test_IncreaseDeposit_RevertWhen_FarmIsInactive() (gas: 2037872)
[PASS] test_IncreaseDeposit_RevertWhen_InvalidAmount() (gas: 1705024)
[PASS] test_Initialize_RevertWhen_InvalidCamelotPoolConfig() (gas: 42538)
```

```
[PASS] test_Initialize_RevertWhen_InvalidTickRange() (gas: 212298)
[PASS] test_Multicall_RevertWhen_CallInternalFunction() (gas: 21185)
[PASS] test_NFTDeposit() (gas: 1973580)
[PASS] test_NFTDeposit_RevertWhen_NoData() (gas: 899398)
[PASS] test_NFTDeposit_RevertWhen_UnauthorisedNFTContract() (gas: 14186)
[PASS] test_OnERC721Received_RevertWhen_IncorrectPoolToken() (gas: 979637)
[PASS] test_OnERC721Received_RevertWhen_IncorrectTickRange() (gas: 1957772)
[PASS] test_RecoverERC20_RevertWhen_CannotWithdrawRewardToken() (gas: 47793)
[PASS] test_RecoverERC20_RevertWhen_CannotWithdrawZeroAmount() (gas: 56285)
[PASS] test_SetRewardRate_RevertWhen_FarmIsClosed() (gas: 151874)
[PASS] test_SetRewardRate_RevertWhen_InvalidRewardRatesLength() (gas: 119013)
[PASS] test_SetupFarm_MAX_NUM_REWARDS() (gas: 836409)
[PASS] test_SetupFarm_RevertWhen_InvalidFarmStartTime() (gas: 276573)
[PASS] test_SetupFarm_RevertWhen_InvalidRewardData() (gas: 421029)
[PASS] test_SetupFarm_RevertWhen_RewardAlreadyAdded() (gas: 576633)
[PASS] test_SubscriptionInfo_RevertWhen_SubscriptionDoesNotExist() (gas: 1561862)
[PASS] test_UpdateCoolDownPeriod_RevertWhen_FarmIsClosed() (gas: 144258)
[PASS] test_UpdateCoolDownPeriod_RevertWhen_InvalidCooldownPeriod() (gas: 25497)
[PASS] test_UpdateCoolDownPeriod_RevertWhen_InvalidCooldownPeriod0() (gas: 25462)
[PASS] test_UpdateCoolDownPeriod_noLockupFarm() (gas: 25414)
[PASS] test_UpdateFarmStartTime_RevertWhen_FarmAlreadyStarted() (gas: 25495)
[PASS] test_UpdateFarmStartTime_RevertWhen_FarmIsClosed() (gas: 144362)
[PASS] test_UpdateFarmStartTime_RevertWhen_InvalidTime() (gas: 678580)
[PASS] test_UpdateRewardTokenData_RevertWhen_FarmIsClosed() (gas: 149898)
[PASS] test_UpdateRewardTokenData_RevertWhen_InvalidAddress() (gas: 32455)
[PASS] test_UpdateRewardTokenData_RevertWhen_NotTheTokenManager() (gas: 37312)
[PASS] test_Withdraw() (gas: 2029427)
[PASS] test_Withdraw_RevertWhen_DepositDoesNotExist_during_withdraw() (gas: 43899)
[PASS] test_Withdraw_RevertWhen_DepositInSameTs() (gas: 1704769)
[PASS] test_Withdraw_RevertWhen_DepositIsInCooldown() (gas: 1749872)
[PASS] test_Withdraw_RevertWhen_PleaseInitiateCooldown() (gas: 1705278)
[PASS] test_Withdraw_closed() (gas: 1897608)
[PASS] test_Withdraw_closedAndExpired() (gas: 1898992)
[PASS] test_Withdraw_notClosedButExpired() (gas: 1819478)
[PASS] test_Withdraw_paused() (gas: 1837913)
[PASS] test_claimCamelotFee() (gas: 2915500)
[PASS] test_claimRewards() (gas: 3310009)
[PASS] test_claimRewards_max_rewards() (gas: 1797388)
[PASS] test_claimRewards_rwd_rate_0() (gas: 1693858)
[PASS] test_closeFarm() (gas: 311271)
[PASS] test_deposit() (gas: 1942841)
[PASS] test_farmPause() (gas: 132135)
[PASS] test_getDeposit() (gas: 1566745)
[PASS] test_getRewardFundInfo_LockupFarm() (gas: 39073)
[PASS] test_getTokenAmounts() (gas: 1735556)
[PASS] test_initialize() (gas: 602786)
[PASS] test_initiateCooldown_LockupFarm() (gas: 2079034)
[PASS] test_initiateCooldown_RevertWhen_DepositDoesNotExist() (gas: 1708227)
[PASS] test_initiateCooldown_nonLockupFarm() (gas: 1560832)
[PASS] test_recoverRewardFund_AfterAddRewards() (gas: 2853879)
[PASS] test_recoverRewardFund_WithDirectlySentFunds() (gas: 786430)
[PASS] test_recoverRewardFund_partially() (gas: 790860)
[PASS] test_rewardBalance() (gas: 2751584)
[PASS] test_subInfo() (gas: 2735606)
[PASS] test_updateFarmStartTime_RevertWhen_FarmHasExpired() (gas: 26884)
[PASS] test_updateFarmStartTime_revertWhen_CallerIsNotOwner() (gas: 30542)
[PASS] test_updateTknManager() (gas: 90089)
[PASS] test_withdraw() (gas: 3436855)
[PASS] test_withdraw_closed() (gas: 3617811)
[PASS] test_withdraw_firstDeposit_multipleDeposits() (gas: 16513725)
[PASS] test_withdraw_inBetweenDeposit_multipleDeposits() (gas: 16513813)
[PASS] test_withdraw_lastDeposit_multipleDeposits() (gas: 16513728)
[PASS] test_withdraw_paused() (gas: 3551984)
Suite result: ok. 114 passed; 0 failed; 0 skipped; finished in 63.40s (96.69s CPU time)

Ran 111 tests for test/e721-farms/uniswapv3/sushiswap/SushiswapV3Farm.t.sol:SushiV3FarmTest
[PASS] testFuzz_DecreaseDeposit(bool,uint256) (runs: 256, μ: 1887087, ~: 1875662)
[PASS] testFuzz_Deposit_Before_Farm_StartTime() (gas: 2539239)
[PASS] testFuzz_IncreaseDeposit(bool,uint256) (runs: 256, μ: 1955797, ~: 2036920)
[PASS] testFuzz_Multicall(uint256) (runs: 256, μ: 172527, ~: 172685)
[PASS] testFuzz_Multicall_RevertWhen_AnyIndividualTestFail(uint256) (runs: 256, μ: 83196, ~: 83323)
[PASS] testFuzz_addRewards(bool,uint256) (runs: 256, μ: 416953, ~: 416997)
```

```
[PASS] testFuzz_extendFarmDuration(bool,uint256,uint256) (runs: 256, μ: 811929, ~: 849949)
[PASS] testFuzz_recoverERC20(bool,uint256) (runs: 256, μ: 229090, ~: 229126)
[PASS] testFuzz_setRewardRate(bool,uint256,uint256) (runs: 256, μ: 186578, ~: 217237)
[PASS] testFuzz_updateCoolDown_lockupFarm(uint256) (runs: 256, μ: 30802, ~: 30835)
[PASS] testFuzz_updateFarmStartTime(bool,uint256,uint256) (runs: 256, μ: 727941, ~: 690679)
[PASS] testFuzz_updateFarmStartTimeWithExpiry(bool,uint256,uint256) (runs: 256, μ: 735350, ~: 771543)
[PASS] testFuzz_updateFarmStartTime_end_time_noDelta(bool,uint256) (runs: 256, μ: 728200, ~: 690608)
[PASS] testFuzz_updateFarmStartTime_end_time_withDelta(bool,uint256) (runs: 256, μ: 733888, ~: 771871)
[PASS] testFuzz_updateFarmStartTime_end_time_withDelta_multiUpdate(bool) (runs: 256, μ: 760730, ~:
799030)
[PASS] testFuzz_withdraw_closedAndExpired() (gas: 3404189)
[PASS] testFuzz_withdraw_notClosedButExpired() (gas: 2789140)
[PASS] test_AddRewards_RevertWhen_InvalidRewardToken() (gas: 221109)
[PASS] test_AddRewards_RevertWhen_ZeroAmount() (gas: 39046)
[PASS] test_ClaimRewards_RevertWhen_DepositDoesNotExist() (gas: 1957381)
[PASS] test_ClaimRewards_RevertWhen_FarmIsClosed() (gas: 2014993)
[PASS] test_ClaimUniswapFee_RevertWhen_DepositDoesNotExist_during_claimUniswapFee() (gas: 48301)
[PASS] test_ClaimUniswapFee_RevertWhen_FarmIsClosed() (gas: 152632)
[PASS] test_ClaimUniswapFee_RevertWhen_NoFeeToClaim() (gas: 1618264)
[PASS] test_CloseFarm_RevertWhen_FarmIsClosed() (gas: 144490)
[PASS] test_DecreaseDeposit_RevertWhen_CannotWithdrawZeroAmount() (gas: 1598552)
[PASS] test_DecreaseDeposit_RevertWhen_DecreaseDepositNotPermitted() (gas: 1598899)
[PASS] test_DecreaseDeposit_RevertWhen_DepositDoesNotExist() (gas: 49003)
[PASS] test_DecreaseDeposit_RevertWhen_FarmIsClosed() (gas: 1677612)
[PASS] test_Deposit_RevertWhen_FarmIsClosed() (gas: 908489)
[PASS] test_Deposit_RevertWhen_FarmIsInactive() (gas: 846439)
[PASS] test_Deposit_RevertWhen_LockupFunctionalityIsDisabled() (gas: 798085)
[PASS] test_Deposit_RevertWhen_NoLiquidityInPosition() (gas: 819044)
[PASS] test_ExtendFarmDuration_RevertWhen_DurationExceeded() (gas: 98877)
[PASS] test_ExtendFarmDuration_RevertWhen_FarmNotYetStarted() (gas: 698585)
[PASS] test_ExtendFarmDuration_RevertWhen_InvalidExtension() (gas: 82848)
[PASS] test_ExtendFarmDuration_RevertWhen_farmClosed() (gas: 144539)
[PASS] test_ExtendFarmDuration_RevertWhen_farmExpired() (gas: 699846)
[PASS] test_FarmPauseSwitch_RevertWhen_FarmAlreadyInRequiredState() (gas: 24940)
[PASS] test_FarmPauseSwitch_RevertWhen_FarmIsClosed() (gas: 145792)
[PASS] test_GetDeposit_RevertWhen_DepositDoesNotExist() (gas: 24733)
[PASS] test_GetRewardBalance_RevertWhen_InvalidRewardToken() (gas: 1606612)
[PASS] test_GetRewardFundInfo_RevertWhen_RewardFundDoesNotExist() (gas: 24525)
[PASS] test_GetRewardTokensTest() (gas: 33892)
[PASS] test_IncreaseDeposit_RevertWhen_DepositDoesNotExist() (gas: 44838)
[PASS] test_IncreaseDeposit_RevertWhen_DepositIsInCooldown() (gas: 1951806)
[PASS] test_IncreaseDeposit_RevertWhen_FarmIsInactive() (gas: 1927353)
[PASS] test_IncreaseDeposit_RevertWhen_InvalidAmount() (gas: 1597936)
[PASS] test_Initialize() (gas: 604116)
[PASS] test_Initialize_RevertWhen_InvalidTickRange() (gas: 141814)
[PASS] test_Initialize_RevertWhen_InvalidUniswapPoolConfig() (gas: 82451)
[PASS] test_Multicall_RevertWhen_CallInternalFunction() (gas: 21185)
[PASS] test_NFTDeposit() (gas: 1877414)
[PASS] test_NFTDeposit_RevertWhen_NoData() (gas: 791945)
[PASS] test_NFTDeposit_RevertWhen_UnauthorisedNFTContract() (gas: 14252)
[PASS] test_OnERC721Received_RevertWhen_IncorrectPoolToken() (gas: 809658)
[PASS] test_OnERC721Received_RevertWhen_IncorrectTickRange() (gas: 1861254)
[PASS] test_RecoverERC20_RevertWhen_CannotWithdrawRewardToken() (gas: 47726)
[PASS] test_RecoverERC20_RevertWhen_CannotWithdrawZeroAmount() (gas: 56307)
[PASS] test_SetRewardRate_RevertWhen_FarmIsClosed() (gas: 151785)
[PASS] test_SetRewardRate_RevertWhen_InvalidRewardRatesLength() (gas: 119055)
[PASS] test_SetupFarm_MAX_NUM_REWARDS() (gas: 833993)
[PASS] test_SetupFarm_RevertWhen_InvalidFarmStartTime() (gas: 274049)
[PASS] test_SetupFarm_RevertWhen_InvalidRewardData() (gas: 418507)
[PASS] test_SetupFarm_RevertWhen_RewardAlreadyAdded() (gas: 574263)
[PASS] test_SubscriptionInfo_RevertWhen_SubscriptionDoesNotExist() (gas: 1454809)
[PASS] test_UpdateCoolDownPeriod_RevertWhen_FarmIsClosed() (gas: 144280)
[PASS] test_UpdateCoolDownPeriod_RevertWhen_InvalidCooldownPeriod() (gas: 25541)
[PASS] test_UpdateCoolDownPeriod_RevertWhen_InvalidCooldownPeriod0() (gas: 25484)
[PASS] test_UpdateCoolDownPeriod_noLockupFarm() (gas: 25458)
[PASS] test_UpdateFarmStartTime_RevertWhen_FarmAlreadyStarted() (gas: 25517)
[PASS] test_UpdateFarmStartTime_RevertWhen_FarmIsClosed() (gas: 144428)
[PASS] test_UpdateFarmStartTime_RevertWhen_InvalidTime() (gas: 676076)
[PASS] test_UpdateRewardTokenData_RevertWhen_FarmIsClosed() (gas: 149818)
[PASS] test_UpdateRewardTokenData_RevertWhen_InvalidAddress() (gas: 32441)
[PASS] test_UpdateRewardTokenData_RevertWhen_NotTheTokenManager() (gas: 37364)
[PASS] test_Withdraw() (gas: 1906546)
```

```
[PASS] test_Withdraw_RevertWhen_DepositDoesNotExist_during_withdraw() (gas: 44009)
[PASS] test_Withdraw_RevertWhen_DepositInSameTs() (gas: 1597804)
[PASS] test_Withdraw_RevertWhen_DepositIsInCooldown() (gas: 1642889)
[PASS] test_Withdraw_RevertWhen_PleaseInitiateCooldown() (gas: 1598296)
[PASS] test_Withdraw_closed() (gas: 1790648)
[PASS] test_Withdraw_closedAndExpired() (gas: 1792066)
[PASS] test_Withdraw_notClosedButExpired() (gas: 1712552)
[PASS] test_Withdraw_paused() (gas: 1730970)
[PASS] test_claimRewards() (gas: 3096722)
[PASS] test_claimRewards_max_rewards() (gas: 1674413)
[PASS] test_claimRewards_rwd_rate_0() (gas: 1586785)
[PASS] test_claimUniswapFee() (gas: 3165736)
[PASS] test_closeFarm() (gas: 311066)
[PASS] test_deposit() (gas: 1838191)
[PASS] test_farmPause() (gas: 131957)
[PASS] test_getDeposit() (gas: 1459674)
[PASS] test_getRewardFundInfo_LockupFarm() (gas: 39095)
[PASS] test_initiateCooldown_LockupFarm() (gas: 1956044)
[PASS] test_initiateCooldown_RevertWhen_DepositDoesNotExist() (gas: 1601156)
[PASS] test_initiateCooldown_nonLockupFarm() (gas: 1453813)
[PASS] test_recoverRewardFund_AfterAddRewards() (gas: 2727463)
[PASS] test_recoverRewardFund_WithDirectlySentFunds() (gas: 786371)
[PASS] test_recoverRewardFund_partially() (gas: 790783)
[PASS] test_rewardBalance() (gas: 2625362)
[PASS] test_subInfo() (gas: 2609386)
[PASS] test_updateFarmStartTime_RevertWhen_FarmHasExpired() (gas: 26971)
[PASS] test_updateFarmStartTime_revertWhen_CallerIsNotOwner() (gas: 30563)
[PASS] test_updateTknManager() (gas: 89869)
[PASS] test_withdraw() (gas: 3223628)
[PASS] test_withdraw_closed() (gas: 3404636)
[PASS] test_withdraw_firstDeposit_multipleDeposits() (gas: 16097693)
[PASS] test_withdraw_inBetweenDeposit_multipleDeposits() (gas: 16097640)
[PASS] test_withdraw_lastDeposit_multipleDeposits() (gas: 16097679)
[PASS] test_withdraw_paused() (gas: 3338775)
Suite result: ok. 111 passed; 0 failed; 0 skipped; finished in 62.76s (124.40s CPU time)

Ran 27 test suites in 74.53s (510.94s CPU time): 730 tests passed, 0 failed, 0 skipped (730 total tests)
```

Code Coverage

Coverage is very thorough.

Fix Review Update:

The Code Coverage still remains high with the new tests considered.

File	% Lines	% Statements	% Branches	% Funcs
contracts/Farm.sol	99.66% (**292**/293)	99.73% (**364**/365)	91.67% (**88**/96)	93.75% (**45**/48)
contracts/FarmDeployer.sol	37.50% (**6**/16)	41.18% (**7**/17)	50.00% (**3**/6)	50.00% (**2**/4)
contracts/FarmRegistry.sol	97.67% (**42**/43)	97.56% (**40**/41)	100.00% (**14**/14)	90.91% (**10**/11)
contracts/e20-**farms/**E20Farm.sol	100.00% (**16**/16)	100.00% (**17**/17)	100.00% (**4**/4)	100.00% (**6**/6)
contracts/e20-**farms/balancerV2/**BalancerV2Farm.sol	83.33% (**5**/6)	85.71% (**6**/7)	100.00% (**0**/0)	80.00% (**4**/5)
contracts/e20-**farms/balancerV2/**BalancerV2FarmDeployer.sol	80.00% (**12**/15)	83.33% (**15**/18)	100.00% (**0**/0)	33.33% (**1**/3)

File	% Lines	% Statements	% Branches	% Funcs
contracts/e20-**farms/uniswapV2/**UniV2Farm.sol	0.00% (**0/**6)	0.00% (**0/**8)	100.00% (**0/**0)	0.00% (**0/**5)
contracts/e20-**farms/uniswapV2/**UniV2FarmDeployer.sol	0.00% (**0/**16)	0.00% (**0/**19)	100.00% (**0/**0)	0.00% (**0/**3)
contracts/e721-**farms/**E721Farm.sol	100.00% (**11/**11)	100.00% (**13/**13)	100.00% (**4/**4)	100.00% (**2/**2)
contracts/e721-**farms/camelotV2/**CamelotV2Farm.sol	96.72% (**59/**61)	93.59% (**73/**78)	70.00% (**14/**20)	100.00% (**13/**13)
contracts/e721-**farms/camelotV2/**CamelotV2FarmDeployer.sol	63.16% (**12/**19)	68.18% (**15/**22)	100.00% (**0/**0)	33.33% (**1/**3)
contracts/e721-**farms/camelotV3/**CamelotV3Farm.sol	98.25% (**56/**57)	98.78% (**81/**82)	81.25% (**13/**16)	100.00% (**9/**9)
contracts/e721-**farms/camelotV3/**CamelotV3FarmDeployer.sol	52.63% (**10/**19)	59.09% (**13/**22)	100.00% (**0/**0)	50.00% (**1/**2)
contracts/e721-**farms/uniswapV3/**UniV3ActiveLiquidityDeployer.sol	50.00% (**9/**18)	55.00% (**11/**20)	100.00% (**0/**0)	50.00% (**1/**2)
contracts/e721-**farms/uniswapV3/**UniV3ActiveLiquidityFarm.sol	100.00% (**11/**11)	100.00% (**19/**19)	100.00% (**2/**2)	100.00% (**4/**4)
contracts/e721-**farms/uniswapV3/**UniV3Farm.sol	98.25% (**56/**57)	97.56% (**80/**82)	87.50% (**14/**16)	88.89% (**8/**9)
contracts/e721-**farms/uniswapV3/**UniV3FarmDeployer.sol	50.00% (**9/**18)	55.00% (**11/**20)	100.00% (**0/**0)	50.00% (**1/**2)
contracts/e721-**farms/uniswapV3/tests/**UniswapV3Test.sol	0.00% (**0/**6)	0.00% (**0/**7)	100.00% (**0/**0)	0.00% (**0/**2)
contracts/features/ExpirableFarm.sol	100.00% (**27/**27)	100.00% (**36/**36)	100.00% (**8/**8)	100.00% (**5/**5)
contracts/features/OperableDeposit.sol	100.00% (**43/**43)	100.00% (**53/**53)	100.00% (**8/**8)	100.00% (**4/**4)
contracts/rewarder/Rewarder.sol	95.93% (**118/**123)	96.89% (**156/**161)	88.89% (**32/**36)	96.15% (**25/**26)
contracts/rewarder/RewarderFactory.sol	100.00% (**13/**13)	100.00% (**14/**14)	100.00% (**2/**2)	100.00% (**5/**5)
contracts/utils/TokenUtils.sol	62.50% (**15/**24)	61.29% (**19/**31)	100.00% (**0/**0)	66.67% (**2/**3)
test/Farm.t.sol	100.00% (**26/**26)	100.00% (**38/**38)	50.00% (**1/**2)	50.00% (**4/**8)

File	% Lines	% Statements	% Branches	% Funcs
test/FarmRegistry.t.sol	100.00% (**11/**11)	100.00% (**11/**11)	100.00% (**0/**0)	75.00% (**3/**4)
test/e20-**farms/balancerV2/**BalancerV2Farm.t.sol	100.00% (**35/**35)	100.00% (**49/**49)	100.00% (**0/**0)	40.00% (**2/**5)
test/e721-**farms/camelotV2/**CamelotV2Farm.t.sol	100.00% (**1/**1)	100.00% (**1/**1)	100.00% (**0/**0)	100.00% (**1/**1)
test/e721-**farms/camelotV3/**CamelotV3Farm.t.sol	100.00% (**71/**71)	100.00% (**103/**103)	75.00% (**3/**4)	58.33% (**7/**12)
test/e721-**farms/uniswapv3/**UniV3ActiveLiquidityFarm.t.sol	92.00% (**23/**25)	87.50% (**28/**32)	100.00% (**0/**0)	60.00% (**3/**5)
test/e721-**farms/uniswapv3/**UniV3Farm.t.sol	100.00% (**65/**65)	100.00% (**99/**99)	75.00% (**3/**4)	54.55% (**6/**11)
test/e721-**farms/uniswapv3/sushiswap/**SushiswapV3Farm.t.sol	100.00% (**5/**5)	100.00% (**5/**5)	100.00% (**0/**0)	100.00% (**1/**1)
test/e721-**farms/uniswapv3/uniswap/**UniswapV3ActiveLiquidityFarm.t.sol	100.00% (**27/**27)	100.00% (**32/**32)	50.00% (**1/**2)	66.67% (**2/**3)
test/e721-**farms/uniswapv3/uniswap/**UniswapV3Farm.t.sol	100.00% (**5/**5)	100.00% (**5/**5)	100.00% (**0/**0)	100.00% (**1/**1)
test/rewarder/Rewarder.t.sol	100.00% (**8/**8)	100.00% (**8/**8)	100.00% (**0/**0)	100.00% (**1/**1)
test/rewarder/RewarderFactory.t.sol	100.00% (**4/**4)	100.00% (**4/**4)	100.00% (**0/**0)	100.00% (**1/**1)
test/utils/BaseSetup.t.sol	100.00% (**12/**12)	100.00% (**16/**16)	100.00% (**0/**0)	100.00% (**3/**3)
test/utils/TestNetworkConfig.t.sol	100.00% (**11/**11)	100.00% (**13/**13)	100.00% (**0/**0)	100.00% (**1/**1)
test/utils/UpgradeUtil.t.sol	100.00% (**3/**3)	100.00% (**5/**5)	100.00% (**0/**0)	100.00% (**2/**2)
test/utils/networkConfig/Arbitrum.t.sol	94.12% (**16/**17)	95.00% (**19/**20)	50.00% (**1/**2)	20.00% (**1/**5)
Total	92.04% (**1145/**1244)	92.84% (**1479/**1593)	87.40% (**215/**246)	78.33% (**188/**240)

Changelog

- 2024-06-13 - Initial report
- 2024-07-01 - Final Report

About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp's team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 500 audits and secured over $200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT