Quantstamp

# Swaap Earn Protocol Vaults

## Executive Summary

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

| | |
|---|---|
| Type | Asset Management |
| Timeline | 2024-03-04 through 2024-03-28 |
| Language | Solidity |
| Methods | Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review |
| Specification | None |
| Source Code | • swaap-labs/swaap-earn-protocol #151426e |
| Auditors | • Guillermo Escobero Auditing Engineer<br>• Roman Rohleder Senior Auditing Engineer<br>• Adrian Koegl Auditing Engineer |

| | |
|---|---|
| Documentation quality | Medium |
| Test quality | Medium |
| Total Findings | 19<br>Fixed: 12 Acknowledged: 6<br>Mitigated: 1 |
| High severity findings | 0 |
| Medium severity findings | 0 |
| Low severity findings | 5 Fixed: 3 Acknowledged: 2 |
| Undetermined severity findings | 0 |
| Informational findings | 14<br>Fixed: 9 Acknowledged: 4<br>Mitigated: 1 |

## Summary of Findings

Swaap is a decentralized asset management protocol that allows fund managers to make use of other pre-approved DeFi protocols. Users can deposit the underlying asset of a fund to acquire shares, thereby investing in the strategy of the fund manager.

Swaap leverages the ERC4626 standard to build asset vaults and has developed adaptors to connect to protocols such as Aave, Balancer, Paraswap, and more. The value of assets in a vault is evaluated using their configured oracles.

Overall the code is well-written and the team has demonstrated great security awareness. We did not find any significant security concerns, having analyzed numerous economic exploits in addition to the code analysis. The test suite could be improved by the team, particularly branch coverage metrics over adaptors.

It was a pleasure to work with the Swaap team who have shown great willingness to improve and optimize their protocol.

While we have not found any specific attack vectors, our main concerns for these contracts are:

1. That a user can deposit shares and temporarily inflate the vault assets' value through a contract attached to an adaptor.
2. That a fund manager can slowly drain the value of the fund.

We recommend reviewing and performing extensive testing before including new positions or adaptors in the future.

**Update:** The Swaap team addressed all the issues found, and updated the test suite to cover the fixes.

| ID | DESCRIPTION | SEVERITY | STATUS |
|---|---|---|---|
| SWA-1 | **Strategist Can Get All Operation Fees Cut** | • Low | Fixed |
| SWA-2 | **Asset Can Be Editted in Less than** `EDIT_ASSET_DELAY` | • Low | Fixed |
| SWA-3 | **Chainlink Decimals Assumption** | • Low | Fixed |

| ID | DESCRIPTION | SEVERITY | STATUS |
|---|---|---|---|
| SWA-4 | Fund Data May Be Out of Sync with Registry | Low | Acknowledged |
| SWA-5 | Privileged Roles and Ownership | Low | Acknowledged |
| SWA-6 | Emergency Pause May Revert and Cause a Delay | Informational | Fixed |
| SWA-7 | `maxDeposit()` Value Can Be Greater than `shareSupplyCap` | Informational | Fixed |
| SWA-8 | Incorrect Chainlink Interface | Informational | Fixed |
| SWA-9 | Ownership of `PriceRouter` Can Be Transferred | Informational | Acknowledged |
| SWA-10 | `removePosition()` Misses Correct ID Check | Informational | Fixed |
| SWA-11 | Solidity Style Guidelines Are Not Adhered To | Informational | Acknowledged |
| SWA-12 | Redundant Contract Size | Informational | Acknowledged |
| SWA-13 | `totalAssets` May Be Externally Manipulated Leading to Share Price Manipulation | Informational | Acknowledged |
| SWA-14 | `Registry.checkAndUpdateFundTradeVolume()` May Overflow `endPeriod` Leading to Incorrect Fund Volume Updates | Informational | Fixed |
| SWA-15 | Missing Input Validation | Informational | Mitigated |
| SWA-16 | Undocumented Magic Constants | Informational | Fixed |
| SWA-17 | Application Monitoring Can Be Improved by Emitting More Events | Informational | Fixed |
| SWA-18 | Unused Code | Informational | Fixed |
| SWA-19 | Unlocked Pragma | 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.

During our security review, we considered the DAO to be honest. If the majority shares of the DAO can be acquired and votes manipulated, additional attack vectors emerge that would allow the attacker to drain value from the fund.

**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**

- `src/base/ERC4626.sol`
- `src/base/Fund.sol`
- `src/base/permutations/FundWithShareLockFlashLoansWhitelisting.sol`
- `src/base/permutations/FundWithShareLockPeriod.sol`
- `src/Registry.sol`
- `src/Deployer.sol`
- `src/modules/adaptors/Aave/V3/AaveV3ATokenManagerAdaptor.sol`
- `src/modules/adaptors/Aave/V3/AaveV3AccountExtension.sol`
- `src/modules/adaptors/Aave/V3/AaveV3AccountHelper.sol`
- `src/modules/adaptors/Aave/V3/AaveV3DebtManagerAdaptor.sol`
- `src/modules/adaptors/AggregatorBaseAdaptor.sol`
- `src/modules/adaptors/BaseAdaptor.sol`
- `src/modules/adaptors/ERC20Adaptor.sol`
- `src/modules/adaptors/PositionlessAdaptor.sol`
- `src/modules/adaptors/Paraswap/ParaswapAdaptor.sol`
- `src/modules/adaptors/Swaap/SwaapV2Adaptor.sol`
- `src/modules/adaptors/Swaap/SwaapFundAdaptor.sol`
- `src/modules/adaptors/Balancer/BalancerFlashLoanAdaptor.sol`
- `src/modules/adaptors/Balancer/BalancerFlashLoanHelper.sol`
- `src/utils/Math.sol`
- `src/utils/SigUtils.sol`
- `src/utils/Uint32Array.sol`
- `src/modules/price-router/PriceRouter.sol`
- `src/modules/price-router/Extensions/Extension.sol`
- `src/modules/price-router/Extensions/Swaap/SwaapSafeguardPoolExtension.sol`
- `src/modules/price-router/Extensions/Balancer/BalancerPoolExtension.sol`
- `src/modules/price-router/Extensions/ERC4626Extension.sol`
- `src/modules/fees/FeesManager.sol`
- `src/modules/fees/ManagementFeesLib.sol`
- `src/modules/fees/PerformanceFeesLib.sol`

# Findings

## SWA-1 Strategist Can Get All Operation Fees Cut

Low Fixed

**Update**

Fixed in: `5da53c9dfa7134bc741618433eacbd58b9f94293` .

The client provided the following explanation:

> *The strategist platform cut is set by the registry owner instead of the Fund's owner which makes more sense in terms of negotiation power. The strategist can stop providing its services if the cut is unfair which is a good leverage. No cap is provided.*

**File(s) affected:** `FeesManager.sol`

**Description:** The strategist (fund owner) can set any value for fees cut (up to 100%) by calling `FeesManager.setStrategistPlatformCut()`. If the fee cut is set to 100%, the protocol will not get any fees from that fund.

**Recommendation:** Consider reviewing the logic if this is not expected. Strategist cut can be limited so the protocol always gets a minimum percentage of the fees, or the logic can be modified so the cut can be only set by the governance.

## SWA-2 Asset Can Be Editted in Less than `EDIT_ASSET_DELAY`

Low Fixed

**Update**

Fixed in: `df11ab0ed67d60587e4916e2ae82d7ca15f91136`.

The client provided the following explanation:

> *Only one edit configuration can be present at a time for an asset.*

**File(s) affected:** `PriceRouter.sol`

**Description:** In the `PriceRouter` contract, the `owner` should only be allowed to edit an asset after `EDIT_ASSET_DELAY` has passed after submitting the intent. However, since several intents can exist simultaneously, the `owner` can edit an asset in less time.

The `owner` can prepare several edit hashes and submit them through `startEditAsset()`. All of them will exist at the same time. Then, at any time after `EDIT_ASSET_DELAY` has passed, the `owner` can switch between those configurations instantaneously.

The design of allowing several intents at the same time may add uncertainty for the user over which configuration will be activated.

**Recommendation:** Consider whether several edit hashes should exist simultaneously. If they can, consider whether all edit hashes should continue to exist once one has been activated through `completeEditAsset()`.

## SWA-3 Chainlink Decimals Assumption

Low Fixed

**Update**

Fixed in: `87b3ef1c4125f0313f68ef9173573c2375a2090f`.

The client provided the following explanation:

> *A check was added.*

**File(s) affected:** `PriceRouter.sol`

**Description:** Although most of the Chainlink price feeds return prices with 18 decimals for ETH pairs and 8 decimals for non-ETH pairs, a small subset of pairs do not follow this.

The protocol assumes 8 decimals for all price feeds, leading to wrong calculations if price decimals are different.

**Recommendation:** While setting a new Chainlink source in `PriceRouter._setupPriceForChainlinkDerivative()`, consider calling `decimals()` in the Chainlink contract and verifying the return value is eight.

## SWA-4 Fund Data May Be Out of Sync with Registry

Low Acknowledged

**Update**

Marked as "Acknowledged" by the client.

The client provided the following explanation:

> *This is intended as a new PriceRouter can be incompatible with a previous version of PriceRouter which can make an old Fund unusable. In addition distrusting an adaptor is intended to be used to (soft) deprecate an old version of an adaptor without having to make it unusable for old Funds.*

**File(s) affected:** `Fund.sol`, `Registry.sol`

**Description:** The funds may be out of sync with the registry in their trusted adaptors and price router. If a previously trusted adaptor is now distrusted in the registry through `distrustAdaptor()`, the funds may continue to use them until `removeAdaptorFromCatalogue()` is called on the respective funds. If an adaptor is distrusted due to an identified security concern, this may result in a delay, as every fund needs to perform this action.

The same holds true for the `priceRouter`: if it requires re-deployment, the new address would have to be updated in all funds first using `cachePriceRouter()`.

**Recommendation:** For every adaptor called in `_makeAdaptorCalls()`, consider calling `revertIfAdaptorIsNotTrusted()` on the registry. Regarding the `priceRouter`, consider using the address stored in the registry such that updating them in the fund is never required.

## SWA-5 Privileged Roles and Ownership

Low Acknowledged

**Update**

Marked as "Acknowledged" by the client.

The client provided the following explanation:

> *We intend to add an extensive explanation of the roles and impacts on the user in the docs.*

**File(s) affected:** `PriceRouter.sol`, `Registry.sol`, `Fund.sol`, `FeesManager.sol`, `Deployer.sol`, `FundWithShareLockPeriod.sol`, `FundWithShareLockFlashLoansWhitelisting.sol`

**Description:** Certain contracts have state variables, e.g. `owner`, which provide certain addresses with privileged roles. Such roles may pose a risk to end-users.

Note that the following listed roles are planned to be controlled by several multi-sigs of the protocol DAO, while only the `automationActions` address will be controlled by the fund's strategist.

The `PriceRouter.sol` contract contains the following privileged roles (as inherited from `Ownable`):

- `_owner`, as initialized during the constructor call to parameter `newOwner`:
  - `renounceOwnership()`: Renounce ownership and thereby be **irreversibly unable to call any of the below-mentioned functions** (while no owner transition through the registry is pending).
  - `transferOwnership()`: Transfer the role to an arbitrary other address (while no owner transition through the registry is pending).
  - `addAsset()`: Add a new asset and its pricing configuration to the contract.
  - `startEditAsset()`: Intent to edit an existing asset, whose changes need to be completed via `completeEditAsset()` after a minimum waiting time of `EDIT_ASSET_DELAY = 7 days`.
  - `completeEditAsset()`: Finalize a pending asset modification after a minimum waiting time of `EDIT_ASSET_DELAY = 7 days`.
  - `cancelEditAsset()`: Cancel a pending asset modification intent.
- `registry.getAddress(0)`, as defined by the given `Registry` contract:
  - Initiate (and cancel) a 2-step time-locked (7 days) role transfer of `_owner` to a given pending address.

**A compromised or malicious `_owner` could**:

- Manipulate the pricing of assets by changing their corresponding configurations, however, such changes would only take effect after a waiting period of 7 days.

**A compromised or malicious `_owner` could however NOT**:

- Immediately manipulate asset pricing or have an immediate influence on a fund's assets.

The `Registry.sol` contract contains the following privileged roles (as inherited from `Ownable`):

- `_owner`, as initialized during the constructor call to parameter `newOwner`:
  - `renounceOwnership()`: Renounce ownership and thereby be **irreversibly unable to call any of the below-mentioned functions** (while no owner transition through the registry is pending).
  - `transferOwnership()`: Transfer the role to an arbitrary other address (while no owner transition through the registry is pending).
  - `setApprovedForDepositOnBehalf()`: Enable/Disable depositing on others' behalf for depositors.
  - `setAddress()`: Change the address of registered contracts (i.e. pricing router).
  - `batchPause()` / `batchUnpause()`: Pause/Unpause multiple fund contract addresses (only up to 9 months after contract creation, after that pausing is no longer possible).
  - `trustAdaptor()` / `distrustAdaptor()`: Allow/Disallow a given address to be used as an adaptor by funds.
  - `trustPosition()` / `distrustPosition`: Allow/Disallow a given position (adaptor with a given configuration) to be used by funds.
  - `setMaxAllowedAdaptorVolumeParams()`: Change the limit a fund may swap in a given time interval.

**A compromised or malicious `_owner` could**:

- Change the price router contract address and thereby influence pricing on all funds, once the new address has been cached by the funds owner.
- Allow arbitrary or malicious positions and execute them on funds leading to loss of funds.
- Disallow positions and thereby impact a strategist's rebalancing options.

The `Fund.sol` contract contains the following privileged roles (as inherited from `Ownable`):

- `_owner`, as initialized during the constructor call to parameter `_owner`:
  - `renounceOwnership()`: Renounce ownership and thereby be **irreversibly unable to call any of the below-mentioned functions** (while no owner transition through the registry is pending).
  - `transferOwnership()`: Transfer the role to an arbitrary other address (while no owner transition through the registry is pending).
  - `cachePriceRouter()`: Update the price router contract address, if allowed by the registry contract (May only be changed to the current price router contract address as defined by the registry).
  - `addPositionToCatalogue()` / `removePositionFromCatalogue()`: Add/Remove position IDs to/from the funds allowed positions catalogue.
  - `addAdaptorToCatalogue()` / `removeAdaptorFromCatalogue()`: Add/Remove adaptors to/from the funds allowed adaptors catalogue.
  - `addPosition()` / `removePosition()`: Add/Remove catalogued positions to/from either the credit or debt position arrays. Position removal is only possible as long as it doesn't hold any balance.
  - `forcePositionOut()`: Remove position even if it still holds a balance, as long as it is no longer trusted by the registry. **This may potentially lead to locked funds.**
  - `swapPositions()`: Swap two positions at the given indices in either the credit or debt position array.
  - `initiateShutdown()` / `liftShutdown()`: Prevent/re-enable the callability of functions `addPosition()`, `beforeDeposit()` and `callOnAdaptor()` (deposits and rebalancing).
  - `setAutomationActions()`: Change the address that is allowed to rebalance the fund (/strategist), if permitted by the registry contract.
  - `setRebalanceDeviation()`: Change the maximum allowed rebalance deviation for total assets (up to `0.1e18 = 10%`, with a default of `0.0003e18`).
  - `callOnAdaptor()`: Rebalance positions of the fund by making arbitrary adaptor calls, while maintaining certain invariants (number of shares or total assets may not change significantly).
  - `setShareSupplyCap()`: Arbitrarily change the funds share supply limit.
- `automationActions` (strategist), as initialized during the constructor call to parameter `_owner`:
  - `callOnAdaptor()`: Rebalance positions of the fund by making arbitrary adaptor calls, while maintaining certain invariants (number of shares or total assets may not change significantly).

**A compromised or malicious `_owner` could**:

- Denial the service by taking on debt position(s) (introducing a health factor that may block withdraws up to a certain amount) and initiate shutdown mode, blocking further deposits. Once the debt position(s) would be liquidated further withdraws would be blocked, ultimately leading to loss of funds.

**A compromised or malicious `_owner` could however NOT**:

- Rebalance to positions unauthorized by the registry.

The `FeesManager.sol` contract contains the following privileged roles:

- Fund owner/strategist, as given by `Fund(fund).owner()`:
  - `setStrategistPayoutAddress()`: Change the fee receiving address for the strategists' fees.
  - `setStrategistPlatformCut()`: Change the strategists' fee cut (**up to `1e18 = 100%`!**), where only the remaining part would go to the protocol.
  - `setManagementFeesPerYear()`: Change the **management** fee rate which is applied on deposits/withdraws.
  - `setPerformanceFees()`: Change the performance fee rate which is applied on deposits/withdraws.
  - `setEnterFees()` / `setExitFees()`: Change the enter/exit fees (up to 10%).
- Registry owner, as given by `registry.owner()`:
  - `setProtocolPayoutAddress()`: Change the fee receiving address for protocol fees.
  - `resetHighWaterMark()`: Reset a fund's high watermark share price to its current share price.

**A compromised or malicious fund owner could**:

- Change the fee cut to entirely go to the strategist or the protocol.

The `Deployer.sol` contract contains the following privileged roles (as inherited from `Owned`):

- `owner`, as initialized during the constructor call to parameter `_owner`:
  - `transferOwnership()`: Transfer the role to an arbitrary other address, including the zero address which would lock out all following listed privileged function calls.
  - `adjustDeployer()`: Add/Remove addresses from the deployer whitelist.

The `FundWithShareLockPeriod.sol` contract contains the following privileged actions for role `_owner` in addition to `Fund.sol` (see above):

- `setShareLockPeriod()`: Change the share lock period (between `5 minutes` and `2 days`, with `5 minutes` being the default).

The `FundWithShareLockFlashLoansWhitelisting.sol` contract contains the following privileged actions for role `_owner` in addition to `FundWithShareLockPeriod.sol` (see above):

- `enableWhitelist()` / `disableWhitelist()`: Enable/Disable the deposit and mint whitelisting which when on only allows callers that have valid signatures from either the strategist or automation actions address.

Summarized, while a compromised or malicious protocol DAO would have extensive control over all aspects of the protocol and its funds and

could lead to loss of funds in multiple ways, a compromised or malicious strategist may only lead to temporarily locked funds or lost funds over time, which in turn could be mitigated by the protocol DAO.

**Recommendation:** Clarify the impact of these privileged actions to the end-users via publicly facing documentation.

## SWA-6 Emergency Pause May Revert and Cause a Delay

Informational | Fixed

**Update**

Fixed in: `71daccef8ad490fdb6d5d3bb2a4ef95a4fef4d6f`.

The client provided the following explanation:

> *Removed revert.*

**File(s) affected:** `Registry.sol`

**Description:** In emergency situations, pausing involved contracts needs to happen quickly. The `Registry.sol` contract offers the functionality to `batchPause()` several contracts at the same time. However, if one of these contracts is already paused, the entire transaction will revert. In urgent situations, this may cause a delay to pausing the intended contracts as one may forget or oversee that a certain contract was already paused.

**Recommendation:** We recommend to not revert in `_pauseTarget()` if a contract is already paused. Accidentally pausing a contract again will not cause any issues. Reverting when a contract is already paused, however, can be critical in unlikely situations.

## SWA-7 `maxDeposit()` Value Can Be Greater than `shareSupplyCap`

Informational | Fixed

**Update**

Fixed in: `be63c44d1f0de59c90f745d3f63bb8433baff51d`.

The client provided the following explanation:

> *Removed special case when supplyCap == type(uint192)max for maxDeposit() and maxMint().*

**File(s) affected:** `Fund.sol`

**Description:** There is a inconsistency between `maxDeposit()` and `deposit()` functions in `Fund` contract. `maxDeposit()` will return `type(uint256).max` if `shareSupplyCap` is `type(uint192).max`, but `deposit()` can revert (`Fund__ShareSupplyCapExceeded()`) if `_totalSupply + shares` is larger than `type(uint192).max`.

**Recommendation:** Consider reviewing `maxDeposit()` logic and modify it to make it consistent with `deposit()` and supply cap limits.

## SWA-8 Incorrect Chainlink Interface

Informational | Fixed

**Update**

Fixed in: `341c883754b3e8127b2648b58865633cec1b31ec`.

The client provided the following explanation:

> *Used correct interfaces*

**Description:** The `IChainlinkAggregator` interface is not entirely correct in its implementation. The `minAnswer()` and `maxAnswer()` function never exist in the same contract as the `aggregator()` function. More specifically, the `aggregator()` function is implemented in Chainlink's price feed contract and the `minAnswer()` and `maxAnswer()` functions are implemented in the aggregator contract returned by the `aggregator()` function.

However, it doesn't cause any issues in this codebase because the different functions are only called on the respective contracts.

**Recommendation:** To prevent confusion when reusing this interface, we recommend splitting it into two interfaces, representing the respective contracts: the aggregator and the price feed contract.

## SWA-9 Ownership of `PriceRouter` Can Be Transferred

Informational | Acknowledged

**Update**

Marked as "Acknowledged" by the client.

The client provided the following explanation:

> *This is intended, the Registry.getAddress[0] should only act as a backup*

**File(s) affected:** `PriceRouter.sol`

**Description:** The `PriceRouter` contract contains functionality for governance to determine a new `owner`. However, the `transferOwnership()` function of `Ownable` is still visible, such that the `owner` could determine a new `owner` themselves.

**Recommendation:** Consider whether this design is intended.

## SWA-10 `removePosition()` Misses Correct ID Check

Informational | Fixed

**Update**

Fixed in: `8c305e28e29db5b681bf324603400bedf776314d`.

The client provided the following explanation:

> *Added input sanity check when removing a position from the Fund.*

**File(s) affected:** `Fund.sol`

**Description:** The `removePosition()` accepts an `index` as parameters that will determine the `positionID` from either the `debtPositions` or `creditPositions`. However, there is no safety check that the resulting `positionID` is the actually intended ID. Due to the usage of several IDs and the possibility of swapping positions, user error can be avoided by making sure the intended position ID is selected. It should be noted that `forcePositionOut()` performs such a check.

**Recommendation:** Consider adding a `positionId` parameter and a sanity check that the intended position ID is selected. This would be the same design as used in `forcePositionOut()`.

## SWA-11 Solidity Style Guidelines Are Not Adhered To

Informational | Acknowledged

**Update**

The client provided the following explanation:

> *We will keep the section-styled format for large contracts.*

**Description:** The solidity style guidelines are an important point of orientation to follow when developing code. Adhering to them allows developers or auditors to understand the code. In most of the contracts, the guidelines are violated in the following ways:

- constructor before variables, events, and errors
- functions in between variables, events, and errors
- convoluted declaration of events in different places
- convoluted declaration of variables in different places
- convoluted declaration of errors in different places

While we understand that section-styled files may be more convenient for the developers of a large project, it is not readable for other users without proper documentation.

1. Consider naming Internal and private variables and functions starting with an underscore: `beforeDeposit()`, `afterDeposit()`...
2. Some typographical errors were found. Consider using a spell checker to fix them.
3. Some comments are outdated and refer to the forked version of the code. Consider updating them to be consistent with the current logic (e.g. `SwaapSafeguardPoolExtension.setupSource()`, `Fund.redeem()`...). Also, Fund contract functions' comments make a distinction between "strategist" and "governance", but only one "owner" address is validated.

**Recommendation:** Consider following the layout of order guidelines outlined here. If you would like to keep the section-styled codebase, please consider documenting the content of a file at the top.

## SWA-12 Redundant Contract Size

Informational | Acknowledged

> **Update**
>
> The client provided the following explanation:
>
> > *We acknowledge the recommendation; however, transitioning to interfaces from our current implementation using libraries such as Solmate would require considerable effort and would only minimally impact the code size.*

**File(s) affected:** `PriceRouter.sol`, `ERC4626Adaptor.sol`, `AaveATokenAdaptor.sol`, `ERC4626.sol`, `Registry.sol`, `FundWithAaveFlashloan.sol`

**Description:** Some contracts include redundant imports that may lead to increased contract size and, thus, deployment cost. This is because, e.g., the `PriceRouter` contract imports the `ERC20` contract. However, the contract does not need to have knowledge of any implementation details; it only needs to know about the function names and parameters. This applies to all the following contracts and imports:

1. `PriceRouter`: `ERC20`
2. `ERC4626`: `ERC20`
3. `ERC4626Adaptor`:
    - `ERC20`
    - `ERC4626`
4. `AaveATokenAdaptor`: `ERC20`
5. `Registry`: `ERC20`
6. `FundWithAaveFlashloan`: `ERC20`
7. `Fund.sol`: `ERC20`

**Recommendation:** We recommend inheriting the respective interface, e.g., `IERC20` and `IERC4626`, and replacing all occurrences in the contracts with their interface equivalent to possibly reduce redundant contract size. As the ERC20 and ERC4626 interfaces vary from their standard, we recommend implementing the required interfaces.

## SWA-13

### `totalAssets` May Be Externally Manipulated Leading to Share Price Manipulation

Informational | Acknowledged

> **Update**
>
> Marked as "Acknowledged" by the client.
>
> The client provided the following explanation:
>
> > *This will increase gas cost and complexity greatly and we prefer to leave this as it is and get the actually owned assets by the Fund.*

**File(s) affected:** `AaveV3ATokenManagerAdaptor.sol`, `SwaapV2Adapter.sol`, `ERC20Adaptor.sol`

**Description:** The amount of total assets held by a fund is crucial for ensuring safe rebalancing or computing the share price. It is computed in `Fund._calculateTotalAssets()` by iterating over all current positions and querying their individual `balanceOf()` implementations. However, the following instances would allow external actors (or a malicious strategist) to manipulate the total assets held.

1. `AaveV3ATokenManagerAdaptor`: The amount of `aToken`s held by an existing and approved position may be increased through an external actor by supplying on behalf of the existing managed account.
2. `SwaapV2Adaptor`: The amount of `SPT`s held by an existing and approved position may be increased through an external actor by joining the same pool and providing the fund as the recipient (or directly transfering `SPT` tokens to the fund address).
3. `ERC20Adaptor`: The amount of a whitelisted ERC20 held may be increased by direct deposits to the fund address.

Such asset transfers would not be subject to the lock period as otherwise enforced in contracts `FundWithShareLock*` and could therefore be executed using flash loans.

**Recommendation:** Consider adding new variables for tracking deposited/withdrawn assets from such adaptors and checking against it when computing the number of assets held.

## SWA-14

### `Registry.checkAndUpdateFundTradeVolume()` May Overflow `endPeriod` Leading to Incorrect Fund Volume Updates

Informational | Fixed

> **Update**
>
> Fixed in: `c833e39a286eae817b0fd6ad542e5649bd7ecdda`.
>
> The client provided the following explanation:
>
> > *Fixed potential overflow*

**File(s) affected:** `Registry.sol`

**Description:** Function `checkAndUpdateFundTradeVolume()` updates variable `endPeriod` as follows:

```
uint256 endPeriod;
unchecked {
   endPeriod = fundVolumeData.lastUpdate + fundVolumeData.periodLength;
}
```

Where struct variables `fundVolumeData.lastUpdate` and `fundVolumeData.periodLength` are both of type `uint48`. While `lastUpdate` can only ever be `uint48(block.timestamp)`, variable `periodLength` is arbitrarily settable by the contract owner through function `setMaxAllowedAdaptorVolumeParams()`.

Given that the addition is executed within an `unchecked {...}` block, the result may overflow, leading to incorrect `endPeriod` values and in turn incorrect `fundVolumeData.volumeInUSD` values, leading to a wrong representation of a funds trade volume.

**Recommendation:** We recommend casting `fundVolumeData.lastUpdate` to `uint256(fundVolumeData.lastUpdate)` (or `fundVolumeData.periodLength` respectively) within the addition operation and removing the `unchecked` block, as to reinterpret the variables as `uint256` and not leading to overflows.

## SWA-15 Missing Input Validation

Informational | Mitigated

**Update**

Addressed in: `633ec65b812b02abc1c09bb94f5e4414af9cfccf`.

The client provided the following explanation:

> *Fixed recommendation #2 and did not fix the others.*
>
> *1: Unlikely to happen, and should revert when comparing asset price to expected asset price*
>
> *3: We intend to have swapRouter set to address(0) initially*
>
> *4: Since the Fund owner has the ability to ignore the check, we do not see the necessity to have a smaller range*
>
> *5: consumes more gas, it should be handled on the backend if needed*
>
> *6: consumes more gas, it should be handled on the backend if needed*

**File(s) affected:** `PriceRouter.sol`, `Registry.sol`, `Fund.sol`, `FeesManager.sol`

**Description:** It is important to validate inputs, even if they only come from trusted addresses, to avoid human error. The following functions do not have a proper validation of input parameters:

1. `PriceRouter._setupPriceForTwapDerivative()`: While unlikely, consider checking `parameters.baseDecimals` to be less than `39` to prevent potential truncations in `PriceRouter.sol#L843` (`uint128(10 ** parameters.baseDecimals)`).
2. `PriceRouter.startEditAsset()`: Consider adding a check for `assetEditableTimestamp[editHash]` being zero to prevent accidental renewal of the delay.
3. `Registry._register()`: Parameter `newContract` not checked to be different from `address(0x0)`.
4. `Fund.cachePriceRouter()`: Parameter `allowableRange` may assume arbitrary values between `0` and `10000` (0% and 100%). Consider adding reasonable bounds.
5. `Fund.addPositionToCatalogue()` (/and `removePositionFromCatalogue()`): Parameter `positionId` not checked to be a new position ID (/existing position ID, respectively), potentially leading to invalid event emissions.
6. `Fund.addAdaptorToCatalogue()` (/and `removeAdaptorFromCatalogue()`): Parameter `adaptor` not checked to be a new adaptor address (/existing adaptor address, respectively), potentially leading to invalid event emissions.

**Recommendation:** Consider adding according input checks.

## SWA-16 Undocumented Magic Constants

Informational | Fixed

**Update**

Fixed in: `808adb600dbb2b2e19860d7a368621071f92e4ff`.

The client provided the following explanation:

> *Documented magic constants*

**File(s) affected:** `PriceRouter.sol`, `Fund.sol`, `AaveV3ATokenManagerAdaptor.sol`, `AggregatorBaseAdaptor.sol`

**Description:** 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. The following instances should therefore be changed accordingly:

1. `PriceRouter.sol#L372`, `L373`, `L672` and `L673`: `1e18`.
2. `PriceRouter.sol#L672`: `1.1e18`.
3. `PriceRouter.sol#L673`: `0.9e18`.
4. `Fund.sol#L127` and `L128`: `1e4` (Consider using `_BPS_ONE_HUNDRED_PER_CENT` instead).
5. `AaveV3ATokenManagerAdaptor.sol#L212`: `1e14`.
6. `AaveV3ATokenManagerAdaptor.sol#L256`: `1e8`.
7. `AggregatorBaseAdaptor.sol#L92`: `1e4`.
8. `AggregatorBaseAdaptor.sol#L157`: `0.96e4`.

**Recommendation:** Ensure that all constants are defined as intended, and use named constants where appropriate. Add documentation explaining the rationale behind each constant.

## SWA-17
## Application Monitoring Can Be Improved by Emitting More Events

Informational · Fixed

**Update**

Fixed in: `c9980e3b1a3d19bd39e5844e764c45424f3bfef0`.

The client provided the following explanation:

> *Added missing events and indexed important data*

**File(s) affected:** `PriceRouter.sol`, `Registry.sol`, `FeesManager.sol`, `Deployer.sol`, `FundWithShareLockFlashLoansWhitelisting.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. `PriceRouter.transitionOwner()`: `pendingOwner` and `transitionStart`.
2. `PriceRouter.cancelTransition()`: `pendingOwner` and `transitionStart`.
3. `PriceRouter.completeTransition()`: `pendingOwner` and `transitionStart`.
4. `Registry.transitionOwner()`: `pendingOwner` and `transitionStart`.
5. `Registry.cancelTransition()`: `pendingOwner` and `transitionStart`.
6. `Registry.completeTransition()`: `pendingOwner` and `transitionStart`.
7. `Registry.trustAdaptor()`: `isAdaptorTrusted[]` and `isIdentifierUsed[]`.
8. `Registry.distrustAdaptor()`: `isAdaptorTrusted[]`.
9. `FeesManager.setEnterFees()`: `fundFeesData[].enterFeesRate`.
10. `FeesManager.setExitFees()`: `fundFeesData[].exitFeesRate`.
11. `Deployer.adjustDeployer()`: `isDeployer[]`.
12. `FundWithShareLockFlashLoansWhitelisting.enableWhitelist()`: `isWhitelistEnabled`.
13. `FundWithShareLockFlashLoansWhitelisting.disableWhitelist()`: `isWhitelistEnabled`.

**Recommendation:** Consider emitting the events.

## SWA-18 Unused Code

Informational · Fixed

**Update**

Fixed in: `129c7b82dfd4ec489abf686578903bd73e671457`.

The client provided the following explanation:

> *1. We intended to leave this as it is in case a new version of an adaptor is deployed that uses this function.*
>
> *2. Fixed (removed)*

**File(s) affected:** `AaveV3AccountExtension.sol`, `Fund.sol`

**Description:** "Dead" code refers to code that is never executed and hence makes no impact on the final result of running a program. Dead code raises a concern, since either the code is unnecessary or the necessary code's results were ignored.

Here are some snippets of dead code found in the codebase:

1. `AaveV3AccountExtension.approveATokenToFund()`
2. `Fund.maxDeposit()` assigns `shareSupplyCap` to `_cap` variable twice. Consider evaluating `_cap` without assigning the `shareSupplyCap` value to it again.

**Recommendation:** Consider removing the unused imports, functions, and variables that are not needed in the codebase.

## SWA-19 Unlocked Pragma

Informational | Fixed

**Update**

Fixed in: `447b6959b44731e9b58e5becafde210323c9d7c1` . The client provided the following explanation:

> *Fixed pragma*

**File(s) affected:** `FeesManager.sol`

**Related Issue(s):** SWC-103

**Description:** Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.8.*`. The caret (`^`) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version *and above*, hence the term "unlocked".

**Recommendation:** For consistency and to prevent unexpected behavior in the future, we recommend to remove the caret to lock the file onto a specific Solidity version.

# 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.

### Contracts

- `bab...8d2 ./src/Registry.sol`
- `184...a10 ./src/Deployer.sol`
- `517...0e6 ./src/base/Fund.sol`
- `f8d...158 ./src/base/ERC4626.sol`
- `872...40f ./src/base/permutations/FundWithShareLockFlashLoansWhitelisting.sol`
- `dd8...b41 ./src/base/permutations/FundWithShareLockPeriod.sol`
- `21a...f62 ./src/modules/adaptors/BaseAdaptor.sol`
- `46a...dac ./src/modules/adaptors/AggregatorBaseAdaptor.sol`
- `94d...ec9 ./src/modules/adaptors/PositionlessAdaptor.sol`

- ec9...7b5 ./src/modules/adaptors/ERC20Adaptor.sol
- b59...fb2 ./src/modules/adaptors/Paraswap/ParaswapAdaptor.sol
- c92...e1d ./src/modules/adaptors/Swaap/SwaapFundAdaptor.sol
- 354...675 ./src/modules/adaptors/Swaap/SwaapV2Adaptor.sol
- 9c3...dd4 ./src/modules/adaptors/Balancer/BalancerFlashLoanHelper.sol
- ee3...48d ./src/modules/adaptors/Balancer/BalancerFlashLoanAdaptor.sol
- f53...3b3 ./src/modules/adaptors/Aave/V3/AaveV3AccountExtension.sol
- 45a...f06 ./src/modules/adaptors/Aave/V3/AaveV3AccountHelper.sol
- aaf...fc6 ./src/modules/adaptors/Aave/V3/AaveV3DebtManagerAdaptor.sol
- e26...462 ./src/modules/adaptors/Aave/V3/AaveV3ATokenManagerAdaptor.sol
- 62e...1ef ./src/modules/price-router/PriceRouter.sol
- 839...f58 ./src/modules/price-router/Extensions/ERC4626Extension.sol
- 5b6...8a0 ./src/modules/price-router/Extensions/Extension.sol
- 83d...dd6 ./src/modules/price-router/Extensions/Swaap/SwaapSafeguardPoolExtension.sol
- 301...052 ./src/modules/price-router/Extensions/Balancer/BalancerPoolExtension.sol
- e7b...4f8 ./src/modules/fees/PerformanceFeesLib.sol
- f2f...ca8 ./src/modules/fees/ManagementFeesLib.sol
- bf2...2d9 ./src/modules/fees/FeesManager.sol
- c17...518 ./src/utils/Math.sol
- e98...5d8 ./src/utils/SigUtils.sol
- 36d...e51 ./src/utils/Uint32Array.sol

**Tests**

- cea...9c2 ./test/FundWithShareLockPeriod.t.sol
- d92...140 ./test/FundWithShareLockFlashLoansWhitelisting.t.sol
- c47...b6b ./test/Registry.t.sol
- 231...b84 ./test/FeesManager.t.sol
- 2dd...8c0 ./test/Fund.sol
- 1a1...54d ./test/testIntegrations/UpdatingPriceRouter.t.sol
- 9d3...cd2 ./test/testPriceRouter/BalancerStablePool.t.sol
- 2d9...c51 ./test/testPriceRouter/ERC4626Extension.t.sol
- 986...c31 ./test/testPriceRouter/SwaapSafeguardPool.t.sol
- d55...380 ./test/testPriceRouter/PriceRouter.t.sol
- 2ef...5c6 ./test/testAdaptors/AaveV3Manager.t.sol
- 2cb...e36 ./test/testAdaptors/ERC20Adaptor.t.sol
- fcb...fa3 ./test/testAdaptors/Paraswap.t.sol
- efe...7a4 ./test/testAdaptors/SwaapV2Adaptor.t.sol
- 72f...bcf ./test/testAdaptors/Aave.t.sol
- 05f...07b ./test/testAdaptors/AggregatorBaseAdaptor.t.sol
- 4d7...11d ./test/testAdaptors/CellarAdaptor.t.sol
- 27e...9fa ./test/testAdaptors/AaveV3.t.sol
- 9a9...4ad ./test/testAdaptors/BalancerPoolAdaptor.t.sol
- 54a...27b ./test/resources/AdaptorHelperFunctions.sol
- 9f3...c9d ./test/resources/MainnetAddresses.sol
- 9ff...67f ./test/resources/PositionIds.sol
- 19a...329 ./test/resources/ContractDeploymentNames.sol
- 01d...b87 ./test/resources/MainnetStarter.t.sol
- 2c2...3be ./test/resources/SwaapMainnetStarter.t.sol
- e8b...567 ./test/resources/Base/BaseAddresses.sol
- 01d...b87 ./test/resources/Arbitrum/ArbitrumStarter.t.sol
- 503...85d ./test/resources/Arbitrum/ArbitrumAddresses.sol

# Toolset

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

**Setup**

Tool Setup:

- Slither v1.0.1

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**

Slither was used to get a static analysis of the repository. All the issues and recommendations are discussed in this report or classified as false positives.

# Test Suite Results

We ran the tests using the following commands and adjust the `.env` file:

```
forge install
forge build
forge test
```

All 383 tests successfully passed.

```
Running 4 tests for test/FundWithShareLockPeriod.t.sol:FundWithShareLockPeriodTest
[PASS] testChainlinkPriceFeedUpdateSandwichAttack() (gas: 6221930)
[PASS] testDepositOnBehalf() (gas: 414544)
[PASS] testShareLockUpPeriod() (gas: 1374756)
[PASS] testTransfer() (gas: 433092)
Test result: ok. 4 passed; 0 failed; 0 skipped; finished in 932.56ms

Running 5 tests for test/testAdaptors/OneInch.t.sol:FundOneInchTest
[PASS] testOneInchSwap() (gas: 533632)
[PASS] testRevertAggregatorSwapIfTotalVolumeIsSurpassed() (gas: 768249)
[PASS] testRevertForUnsupportedAssets() (gas: 778859)
[PASS] testSlippageChecks() (gas: 3756709)
[PASS] testVolumeIsIncrementedCorrectly() (gas: 1719983)
Test result: ok. 5 passed; 0 failed; 0 skipped; finished in 397.24ms

Running 17 tests for test/testPriceRouter/PriceRouter.t.sol:PriceRouterTest
[PASS] testAddAssetWithInvalidMaxPrice() (gas: 169602)
[PASS] testAddAssetWithInvalidMinPrice() (gas: 169415)
[PASS] testAddChainlinkAsset() (gas: 257349)
[PASS] testAddInvalidAsset() (gas: 21551)
[PASS] testAddingATwapAsset() (gas: 16351715)
[PASS] testAssetAboveMaxPrice() (gas: 187718)
[PASS] testAssetBelowMinPrice() (gas: 187995)
[PASS] testAssetStalePrice() (gas: 46630)
[PASS] testETHtoUSDPriceFeedIsChecked() (gas: 379483)
[PASS] testEditAsset() (gas: 110944)
[PASS] testExchangeRate() (gas: 16711011)
[PASS] testGetValue(uint256,uint256,uint256) (runs: 256, μ: 309322, ~: 309266)
[PASS] testMinPriceGreaterThanMaxPrice() (gas: 171659)
[PASS] testNumericError() (gas: 223360)
[PASS] testTransitioningOwner() (gas: 215284)
[PASS] testUnsupportedAsset() (gas: 122133)
[PASS] testWrongChainlinkDecimals() (gas: 84956)
Test result: ok. 17 passed; 0 failed; 0 skipped; finished in 1.71s

Running 5 tests for test/testAdaptors/Paraswap.t.sol:FundParaswapTest
[PASS] testParaswapSwap() (gas: 540916)
[PASS] testRevertAggregatorSwapIfTotalVolumeIsSurpassed() (gas: 764763)
```

```
[PASS] testRevertForUnsupportedAssets() (gas: 778813)
[PASS] testSlippageChecks() (gas: 3748777)
[PASS] testVolumeIsIncrementedCorrectly() (gas: 1716041)
Test result: ok. 5 passed; 0 failed; 0 skipped; finished in 623.30ms

Running 5 tests for test/testPriceRouter/RedstonePriceFeedExtension.t.sol:RedstonePriceFeedExtensionTest
[PASS] testRedstonePriceFeedExtension() (gas: 319859)
[PASS] testRedstonePriceFeedExtensionSwEth() (gas: 219918)
[PASS] testUsingExtensionWithStalePrice() (gas: 177628)
[PASS] testUsingExtensionWithWrongDataFeedId() (gas: 168751)
[PASS] testUsingExtensionWithZeroPrice() (gas: 158051)
Test result: ok. 5 passed; 0 failed; 0 skipped; finished in 729.40ms

Running 6 tests for
test/testPriceRouter/RedstonePriceFeedExtensionEth.t.sol:RedstoneEthPriceFeedExtensionTest
[PASS] testRedstoneEthPriceFeedExtension() (gas: 424577)
[PASS] testUsingActualSwethEthFeed() (gas: 214997)
[PASS] testUsingExtensionWithStalePrice() (gas: 265524)
[PASS] testUsingExtensionWithWrongDataFeedId() (gas: 260619)
[PASS] testUsingExtensionWithZeroPrice() (gas: 245320)
[PASS] testUsingExtensionWithoutPriceRouterSupportingWETH() (gas: 266894)
Test result: ok. 6 passed; 0 failed; 0 skipped; finished in 606.79ms

Running 8 tests for test/Registry.t.sol:RegistryTest
[PASS] testInitialization() (gas: 28771)
[PASS] testRegister() (gas: 46044)
[PASS] testSetAddress() (gas: 58251)
[PASS] testSetAddressOfInvalidId() (gas: 19867)
[PASS] testSetApprovedForDepositOnBehalf() (gas: 33600)
[PASS] testTransitioningOwner() (gas: 204577)
[PASS] testTrustingAndDistrustingAdaptor() (gas: 519818)
[PASS] testTrustingAndDistrustingPosition() (gas: 705778)
Test result: ok. 8 passed; 0 failed; 0 skipped; finished in 6.57ms

Running 1 test for test/testPriceRouter/SequencerPriceRouter.t.sol:SequencerPriceRouterTest
[PASS] testSequencerUptimeFeedLogic() (gas: 108783)
Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 340.70ms

Running 3 tests for test/FundWithERC4626Adaptor.t.sol:FundWithERC4626AdaptorTest
[PASS] testFundWithFundPositions() (gas: 11089980)
[PASS] testTotalAssets(uint256,uint256,uint256,uint256,uint256) (runs: 256, μ: 2702593, ~: 2702587)
[PASS] testUsingIlliquidFundPosition() (gas: 6678519)
Test result: ok. 3 passed; 0 failed; 0 skipped; finished in 6.02s

Running 9 tests for
test/FundWithShareLockFlashLoansWhitelisting.t.sol:MockFundWithShareLockFlashLoansWhitelistingTest
[PASS] testRandomUserTurnWhitelistOff() (gas: 20229)
[PASS] testRandomUserTurnWhitelistOn() (gas: 16753)
[PASS] testSetUpState() (gas: 11694)
[PASS] testSignatureVerificationAsFundAutomationActions() (gas: 138893)
[PASS] testSignatureVerificationAsFundOwner() (gas: 78817)
[PASS] testUserDelayAboveValidity() (gas: 141357)
[PASS] testWhitelistOff() (gas: 708663)
[PASS] testWhitelistOnAndThenOff() (gas: 726515)
[PASS] testWrongSignatureWhitelistOff() (gas: 17334)
Test result: ok. 9 passed; 0 failed; 0 skipped; finished in 21.00ms

Running 9 tests for test/SimpleSlippageRouter.t.sol:SimpleSlippageRouterTest
[PASS] testBadDeadline(uint256) (runs: 256, μ: 193910, ~: 194000)
[PASS] testDeposit(uint256) (runs: 256, μ: 480566, ~: 480660)
[PASS] testDepositMinimumSharesUnmet(uint256) (runs: 256, μ: 503675, ~: 503767)
[PASS] testMint(uint256) (runs: 256, μ: 529280, ~: 529348)
[PASS] testMintMaxAssetsRqdSurpassed(uint256) (runs: 256, μ: 675555, ~: 675646)
[PASS] testRedeem(uint256) (runs: 256, μ: 601566, ~: 601658)
[PASS] testRedeemMinAssetsUnmet(uint256) (runs: 256, μ: 540894, ~: 540976)
[PASS] testWithdraw(uint256) (runs: 256, μ: 550067, ~: 550175)
[PASS] testWithdrawMaxSharesSurpassed(uint256) (runs: 256, μ: 530885, ~: 530974)
Test result: ok. 9 passed; 0 failed; 0 skipped; finished in 7.01s

Running 5 tests for test/testAdaptors/CellarAdaptorWithSDai.t.sol:SwaapFundAdaptorWithSDaiTest
[PASS] testDeposit(uint256) (runs: 256, μ: 488182, ~: 488265)
[PASS] testInterestAccrual(uint256) (runs: 256, μ: 538008, ~: 538084)
```

```
[PASS] testStrategistFunctions(uint256) (runs: 256, μ: 946564, ~: 946703)
[PASS] testUsersGetPendingInterest(uint256) (runs: 256, μ: 792792, ~: 793517)
[PASS] testWithdraw(uint256) (runs: 256, μ: 647718, ~: 648017)
Test result: ok. 5 passed; 0 failed; 0 skipped; finished in 7.00s

Running 21 tests for test/testAdaptors/AaveV3Manager.t.sol:FundAaveV3Test
[PASS] testBlockExternalReceiver() (gas: 195736)
[PASS] testCreateAaveAccountExtension() (gas: 1111280)
[PASS] testDefaultAaveAccountDeployedOnInitDeposit() (gas: 15720)
[PASS] testDeposit(uint256) (runs: 256, μ: 495134, ~: 495211)
[PASS] testIntegration() (gas: 5590646)
[PASS] testMultipleATokensAndDebtTokens() (gas: 2943575)
[PASS] testRepayingDebtThatIsNotOwed() (gas: 501428)
[PASS] testRepayingLoans() (gas: 903045)
[PASS] testRevertWhenDeployAaveAccountWithIncorrectData() (gas: 205608)
[PASS] testTakingOutAFlashLoan() (gas: 942813)
[PASS] testTakingOutLoanInUntrackedPosition() (gas: 475740)
[PASS] testTakingOutLoans() (gas: 800143)
[PASS] testTakingOutLoansInUntrackedPosition() (gas: 555612)
[PASS] testTotalAssets(uint256) (runs: 256, μ: 488341, ~: 488424)
[PASS] testWithdraw(uint256) (runs: 256, μ: 745156, ~: 745231)
[PASS] testWithdrawableFromaV3USDC() (gas: 1321962)
[PASS] testWithdrawableFromaV3WETH() (gas: 2524730)
[PASS] testWithdrawalLogicEModeWithDebt() (gas: 2973984)
[PASS] testWithdrawalLogicEmodeNoDebt() (gas: 3170074)
[PASS] testWithdrawalLogicNoEModeNoDebt() (gas: 1801743)
[PASS] testWithdrawalLogicNoEModeWithDebt() (gas: 2747471)
Test result: ok. 21 passed; 0 failed; 0 skipped; finished in 7.05s

Running 6 tests for test/testPriceRouter/StEthExtension.t.sol:StEthExtensionTest
[PASS] testAnswersDiverging() (gas: 646521)
[PASS] testChainlinkReverts() (gas: 324741)
[PASS] testStEthExtension() (gas: 294718)
[PASS] testUniswapOracleFailuresDefaultingToChainlinkIfMeanLiquidityLow() (gas: 11849613)
[PASS] testUniswapOracleFailuresDefaultingToChainlinkIfObservationsToNew() (gas: 11815639)
[PASS] testUsingExtensionWithWrongAsset() (gas: 47714)
Test result: ok. 6 passed; 0 failed; 0 skipped; finished in 721.89ms

Running 15 tests for test/testAdaptors/Aave.t.sol:FundAaveTest
[PASS] testAddingPositionWithUnsupportedAssetsReverts() (gas: 444670)
[PASS] testBlockExternalReceiver() (gas: 471240)
[PASS] testDeposit(uint256) (runs: 256, μ: 458718, ~: 458809)
[PASS] testIntegration() (gas: 5979365)
[PASS] testMultipleATokensAndDebtTokens() (gas: 3148837)
[PASS] testRepayingDebtThatIsNotOwed() (gas: 497283)
[PASS] testRepayingLoans(uint256) (runs: 256, μ: 925425, ~: 924314)
[PASS] testTakingOutAFlashLoan() (gas: 973540)
[PASS] testTakingOutLoanInUntrackedPosition() (gas: 475333)
[PASS] testTakingOutLoans(uint256) (runs: 256, μ: 794648, ~: 794774)
[PASS] testTakingOutLoansInUntrackedPosition() (gas: 550695)
[PASS] testTotalAssets(uint256) (runs: 256, μ: 487841, ~: 487921)
[PASS] testWithdraw(uint256) (runs: 256, μ: 809465, ~: 809545)
[PASS] testWithdrawableFromaV2USDC() (gas: 1481364)
[PASS] testWithdrawableFromaV2WETH() (gas: 2772977)
Test result: ok. 15 passed; 0 failed; 0 skipped; finished in 8.05s

Running 5 tests for test/testAdaptors/SushiswapV3.t.sol:SushiswapV3AdaptorTest
[PASS] testAddingToExistingPosition() (gas: 1583727)
[PASS] testOpenUSDC_DAIPosition() (gas: 1271264)
[PASS] testOpeningAndClosingUniV3Position() (gas: 1428168)
[PASS] testRangeOrders() (gas: 1032416)
[PASS] testTakingFromExistingPosition() (gas: 1526250)
Test result: ok. 5 passed; 0 failed; 0 skipped; finished in 393.15ms

Running 7 tests for test/testAdaptors/Compound.t.sol:FundCompoundTest
[PASS] testAddingPositionWithUnsupportedAssetsReverts() (gas: 438478)
[PASS] testClaimCompAndVest() (gas: 3107917)
[PASS] testDeposit(uint256) (runs: 256, μ: 449830, ~: 449903)
[PASS] testErrorCodeCheck() (gas: 1238090)
[PASS] testMaliciousStrategistMovingFundsIntoUntrackedCompoundPosition() (gas: 4017667)
[PASS] testTotalAssets() (gas: 1030155)
[PASS] testWithdraw(uint256) (runs: 256, μ: 731760, ~: 731820)
```

```
Test result: ok. 7 passed; 0 failed; 0 skipped; finished in 2.47s

Running 4 tests for test/testAdaptors/AggregatorBaseAdaptor.t.sol:FundAggregatorBaseAdaptorTest
[PASS] testRevertAggregatorSwapIfTotalVolumeIsSurpassed() (gas: 768047)
[PASS] testRevertForUnsupportedAssets() (gas: 778537)
[PASS] testSlippageChecks() (gas: 3754458)
[PASS] testVolumeIsIncrementedCorrectly() (gas: 1719287)
Test result: ok. 4 passed; 0 failed; 0 skipped; finished in 44.36ms

Running 23 tests for test/testAdaptors/UniswapV3.t.sol:UniswapV3AdaptorTest
[PASS] testAddingPositionWithUnsupportedToken0Reverts() (gas: 203122)
[PASS] testAddingPositionWithUnsupportedToken1Reverts() (gas: 285285)
[PASS] testAddingToExistingPosition() (gas: 1557262)
[PASS] testFundAddingAndRemovingPositionReverts() (gas: 2077435)
[PASS] testFundPurgingSinglePositionsAndAllUnusedPositions() (gas: 9568113)
[PASS] testFundWithSmorgasbordOfUniV3Positions() (gas: 3788461)
[PASS] testGriefingAttack() (gas: 5103646)
[PASS] testHandlingUnusedApprovals() (gas: 1998658)
[PASS] testIdsAreIgnoredIfNotOwnedByFund() (gas: 2181904)
[PASS] testIntegration() (gas: 30714820)
[PASS] testIsDebtReturnsFalse() (gas: 7133)
[PASS] testOpenUSDC_DAIPosition() (gas: 1126686)
[PASS] testOpenUSDC_WETHPosition() (gas: 1246355)
[PASS] testOpeningAndClosingUniV3Position() (gas: 1308924)
[PASS] testPositionBurning() (gas: 2757660)
[PASS] testRangeOrders() (gas: 1034144)
[PASS] testTakingFees() (gas: 2325039)
[PASS] testTakingFromExistingPosition() (gas: 1463115)
[PASS] testUserDepositAndWithdrawRevert() (gas: 18052)
[PASS] testUsingLPTokensNotOwnedByFundOrTokensThatDoNotExist() (gas: 645776)
[PASS] testUsingUntrackedLPPosition() (gas: 437080)
[PASS] testWithdrawableFromReturnsZero() (gas: 12773)
[PASS] testWorkingWithMaxNumberOfTrackedTokens() (gas: 31402955)
Test result: ok. 23 passed; 0 failed; 0 skipped; finished in 805.26ms

Running 3 tests for test/testPriceRouter/SwaapSafeguardPool.t.sol:SwaapSafeguarPoolTest
[PASS] testPoolPricingWithManagementFees(uint256) (runs: 256, μ: 676422, ~: 676511)
[PASS] testPricingSafeguardPoolWithUnsupportedPool() (gas: 26980)
[PASS] testPricingUSDC_WETH_SPT(uint256) (runs: 256, μ: 846146, ~: 846271)
Test result: ok. 3 passed; 0 failed; 0 skipped; finished in 4.03s

Running 7 tests for test/testPriceRouter/BalancerStablePool.t.sol:BalancerStablePoolTest
[PASS] testMisConfiguredStorageData() (gas: 653507)
[PASS] testPricingBb_a_Usd() (gas: 1534923)
[PASS] testPricingCBETH_WSTETH_Bpt(uint256) (runs: 256, μ: 1212273, ~: 1212371)
[PASS] testPricingRETH_WETH_Bpt(uint256) (runs: 256, μ: 1008675, ~: 1010027)
[PASS] testPricingStablePoolWithUnsupportedUnderlying() (gas: 216578)
[PASS] testPricingUSDC_DAI_USDT_Bpt(uint256) (runs: 256, μ: 1043645, ~: 1048582)
[PASS] testPricingWstETH_WETH_Bpt(uint256) (runs: 256, μ: 877510, ~: 877983)
Test result: ok. 7 passed; 0 failed; 0 skipped; finished in 8.68s

Running 1 test for test/testAdaptors/CellarAdaptor.t.sol:SwaapFundAdaptorTest
[PASS] testUsingIlliquidFundPosition() (gas: 5879170)
Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 12.99ms

Running 12 tests for test/VestingSimple.t.sol:VestingTest
[PASS] testDepositOnBehalf(uint256,uint256) (runs: 256, μ: 336261, ~: 336261)
[PASS] testFailDepositLessThanMinimum() (gas: 8831)
[PASS] testFailDepositZero() (gas: 11456)
[PASS] testFullVest(uint256) (runs: 256, μ: 267197, ~: 267201)
[PASS] testFullVestWithPartialClaim(uint256,uint256) (runs: 256, μ: 292722, ~: 292727)
[PASS] testInitialization() (gas: 12017)
[PASS] testMultipleClaims(uint256,uint256) (runs: 256, μ: 339154, ~: 339157)
[PASS] testMultipleDeposits(uint256,uint256) (runs: 256, μ: 516496, ~: 516496)
[PASS] testMultipleUsers(uint256,uint256) (runs: 256, μ: 705037, ~: 705039)
[PASS] testPartialVest(uint256,uint256) (runs: 256, μ: 314578, ~: 314578)
[PASS] testPartialVestWithPartialClaim(uint256,uint256,uint256) (runs: 256, μ: 356088, ~: 356088)
[PASS] testWithdrawAnyFor(uint256,uint256) (runs: 256, μ: 559561, ~: 559560)
Test result: ok. 12 passed; 0 failed; 0 skipped; finished in 2.42s

Running 3 tests for test/testPriceRouter/WstEthExtension.t.sol:WstEthExtensionTest
[PASS] testAddingWstethWithoutPricingSteth() (gas: 92519)
```

```
[PASS] testUsingExtensionWithWrongAsset() (gas: 88072)
[PASS] testWstEthExtension() (gas: 220080)
Test result: ok. 3 passed; 0 failed; 0 skipped; finished in 10.88ms

Running 2 tests for test/testAdaptors/ZeroX.t.sol:FundZeroXTest
[PASS] test0xSwap() (gas: 831499)
[PASS] testSlippageChecks() (gas: 3883999)
Test result: ok. 2 passed; 0 failed; 0 skipped; finished in 382.35ms

Running 6 tests for test/testAdaptors/DSRAdaptor.t.sol:FundDSRTest
[PASS] testDeposit(uint256) (runs: 256, μ: 398504, ~: 398592)
[PASS] testDrip(uint256) (runs: 256, μ: 589112, ~: 589229)
[PASS] testInterestAccrual(uint256) (runs: 256, μ: 544316, ~: 544381)
[PASS] testStrategistFunctions(uint256) (runs: 256, μ: 851314, ~: 851452)
[PASS] testUsersDoNotGetPendingInterest(uint256) (runs: 256, μ: 678366, ~: 678477)
[PASS] testWithdraw(uint256) (runs: 256, μ: 537626, ~: 537692)
Test result: ok. 6 passed; 0 failed; 0 skipped; finished in 6.92s

Running 2 tests for test/testPriceRouter/ERC4626Extension.t.sol:ERC4626ExtensionTest
[PASS] testERC4626ExtensionSDai() (gas: 231293)
[PASS] testUsingExtensionWithUnsupportedAsset() (gas: 196339)
Test result: ok. 2 passed; 0 failed; 0 skipped; finished in 9.38ms

Running 9 tests for test/testAdaptors/Vesting.t.sol:FundVestingTest
[PASS] testCannotTakeUserDeposits(uint256) (runs: 256, μ: 372741, ~: 372825)
[PASS] testDepositAndWithdrawReturnsZero(uint256) (runs: 256, μ: 455837, ~: 455977)
[PASS] testDepositToVesting(uint256) (runs: 256, μ: 465689, ~: 465814)
[PASS] testFailWithdrawMoreThanVested(uint256) (runs: 256, μ: 530001, ~: 530109)
[PASS] testStrategistPartialWithdrawFromVesting(uint256) (runs: 256, μ: 605865, ~: 605978)
[PASS] testStrategistWithdrawAllFromVesting(uint256) (runs: 256, μ: 631757, ~: 631901)
[PASS] testStrategistWithdrawAnyFromVesting(uint256) (runs: 256, μ: 505201, ~: 505284)
[PASS] testStrategistWithdrawFromVesting(uint256) (runs: 256, μ: 505725, ~: 505805)
[PASS] testUserWithdrawFromVesting(uint256) (runs: 256, μ: 646367, ~: 648452)
Test result: ok. 9 passed; 0 failed; 0 skipped; finished in 5.21s

Running 1 test for test/testAdaptors/ERC20Adaptor.t.sol:ERC20AdaptorTest
[PASS] testLogic(uint256,uint256) (runs: 256, μ: 797216, ~: 800676)
Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 910.62ms

Running 21 tests for test/FeesManager.t.sol:FeesManagerTest
[PASS] testCollectFeesFromFund() (gas: 702046)
[PASS] testCollectFeesWhenSettingManagementFees() (gas: 704160)
[PASS] testCollectFeesWhenSettingPerformanceFees() (gas: 1212337)
[PASS] testDepositAndMintSharePriceAreEqualWithEnterFeesOn() (gas: 579941)
[PASS] testEnterFeesDepositHook() (gas: 926333)
[PASS] testEnterFeesMintHook() (gas: 951835)
[PASS] testExitFeesRedeemHook(uint16) (runs: 256, μ: 975492, ~: 975709)
[PASS] testExitFeesWithdrawHook(uint16) (runs: 256, μ: 801848, ~: 802084)
[PASS] testFeesPayoutWithStrategistAddressAndCutUnset() (gas: 627818)
[PASS] testFeesPayoutWithStrategistAddressUnset() (gas: 808027)
[PASS] testFeesPayoutWithStrategistPayoutAndCutSet() (gas: 1000848)
[PASS] testHighWaterMarkReset() (gas: 1875443)
[PASS] testManagementFeesEnterHook() (gas: 1259784)
[PASS] testPerformanceFeesDepositHook() (gas: 1880272)
[PASS] testPerformanceFeesMintHook() (gas: 1722740)
[PASS] testPerformanceFeesRedeemHook() (gas: 1565626)
[PASS] testPerformanceFeesWithdrawHook() (gas: 1564958)
[PASS] testRevertOnWrongCaller() (gas: 60874)
[PASS] testRevertOnWrongFeesInputs() (gas: 44052)
[PASS] testUpdateFeesRatesCorrectly() (gas: 1210289)
[PASS] testWithdrawAndRedeemSharePriceAreEqualWithExitFeesOn(uint256,uint16) (runs: 256, μ: 588825, ~:
588722)
Test result: ok. 21 passed; 0 failed; 0 skipped; finished in 3.01s

Running 12 tests for test/testAdaptors/SwaapV2Adaptor.t.sol:SwaapV2AdaptorTest
[PASS] testAllowlistJoin(uint256) (runs: 256, μ: 848889, ~: 848960)
[PASS] testAssetsUsed() (gas: 12014)
[PASS] testBalancerFlashLoanChecks() (gas: 35038)
[PASS] testDepositToHoldingPosition() (gas: 6221727)
[PASS] testExitPoolReverts() (gas: 767968)
[PASS] testFailTransferEthToFund() (gas: 18089)
[PASS] testIsDebt() (gas: 10142)
```

```
[PASS] testJoinPoolReverts() (gas: 936629)
[PASS] testSwaapFlashLoans() (gas: 674591)
[PASS] testSwaapProportionalExitPool(uint256) (runs: 256, μ: 841112, ~: 841228)
[PASS] testTotalAssetsAfterExit(uint256) (runs: 256, μ: 709977, ~: 710091)
[PASS] testTotalAssetsAfterJoin(uint256) (runs: 256, μ: 932490, ~: 932644)
Test result: ok. 12 passed; 0 failed; 0 skipped; finished in 6.81s

Running 13 tests for test/testAdaptors/AaveV2Morpho.t.sol:FundAaveV2MorphoTest
[PASS] testBlockExternalReceiver(uint256) (runs: 256, μ: 1375570, ~: 1375657)
[PASS] testDeposit(uint256) (runs: 256, μ: 524666, ~: 552949)
[PASS] testHealthFactorChecks() (gas: 2411158)
[PASS] testIntegrationRealYieldEth(uint256) (runs: 256, μ: 3168397, ~: 3168397)
[PASS] testIntegrationRealYieldUsd(uint256) (runs: 256, μ: 7412389, ~: 7413232)
[PASS] testIsBorrowingAnyFullRepay(uint256) (runs: 256, μ: 1922351, ~: 1918868)
[PASS] testRepayingDebtThatIsNotOwed(uint256) (runs: 256, μ: 1064422, ~: 1064532)
[PASS] testRepayingLoans(uint256) (runs: 256, μ: 2822403, ~: 2822497)
[PASS] testTakingOutLoans(uint256) (runs: 256, μ: 2614007, ~: 2614109)
[PASS] testTakingOutLoansInUntrackedPosition(uint256) (runs: 256, μ: 1428861, ~: 1428956)
[PASS] testTotalAssets(uint256) (runs: 256, μ: 578423, ~: 607804)
[PASS] testWithdraw(uint256) (runs: 256, μ: 1180865, ~: 1208043)
[PASS] testWithdrawalLogic(uint256) (runs: 256, μ: 2054108, ~: 2050315)
Test result: ok. 13 passed; 0 failed; 0 skipped; finished in 23.82s

Running 15 tests for test/testAdaptors/FraxLendFToken.t.sol:FraxLendFTokenAdaptorTest
[PASS] testAddInterest() (gas: 417677)
[PASS] testDeposit(uint256) (runs: 256, μ: 312063, ~: 312140)
[PASS] testDepositV2(uint256) (runs: 256, μ: 481814, ~: 481888)
[PASS] testDifferencesWhenAccountingForInterestV1() (gas: 12386115)
[PASS] testDifferencesWhenAccountingForInterestV2() (gas: 11745043)
[PASS] testLendingFrax(uint256) (runs: 256, μ: 396572, ~: 396655)
[PASS] testMultiplePositionsTotalAssets(uint256) (runs: 256, μ: 976654, ~: 976781)
[PASS] testMultiplePositionsUserWithdraw(uint256) (runs: 256, μ: 1303159, ~: 1303232)
[PASS] testRebalancingBetweenPairs(uint256) (runs: 256, μ: 813760, ~: 813891)
[PASS] testTotalAssets(uint256) (runs: 256, μ: 342606, ~: 342692)
[PASS] testUsingPairNotSetupAsPosition(uint256) (runs: 256, μ: 415618, ~: 415730)
[PASS] testWithdraw(uint256) (runs: 256, μ: 420700, ~: 420761)
[PASS] testWithdrawV2(uint256) (runs: 256, μ: 610582, ~: 610756)
[PASS] testWithdrawableFrom() (gas: 1198215)
[PASS] testWithdrawingFrax(uint256) (runs: 256, μ: 427894, ~: 427978)
Test result: ok. 15 passed; 0 failed; 0 skipped; finished in 9.15s

Running 29 tests for test/Fund.sol:FundTest
[PASS] testCachePriceRouter() (gas: 1653053)
[PASS] testCallerOfCallOnAdaptor() (gas: 1226513)
[PASS] testDebtTokensInFunds() (gas: 6708577)
[PASS] testDepeggedAssetNotUsedByFund() (gas: 1004538)
[PASS] testDepeggedAssetUsedByTheFund() (gas: 1412292)
[PASS] testDepeggedFundAsset() (gas: 2445013)
[PASS] testDepeggedHoldingPosition() (gas: 1979117)
[PASS] testDepositAndWithdraw(uint256) (runs: 256, μ: 2169421, ~: 2169141)
[PASS] testDepositMintWithdrawRedeemWithZeroInputs() (gas: 1804729)
[PASS] testEndPauseDurationButFundIsShutDownThenLiftShutdown() (gas: 6417148)
[PASS] testFundDNOSPerformanceFeesWithZeroShares() (gas: 1113311)
[PASS] testFundWithFundPositions() (gas: 11090191)
[PASS] testInitialization() (gas: 209289)
[PASS] testInteractingWithDistrustedAdaptors() (gas: 1286391)
[PASS] testInteractingWithDistrustedPositions() (gas: 492605)
[PASS] testLimits() (gas: 1431146)
[PASS] testManagingPositions() (gas: 2177768)
[PASS] testMintAndRedeem(uint256) (runs: 256, μ: 1768807, ~: 1768981)
[PASS] testProhibitedActionsWhileShutdown() (gas: 684607)
[PASS] testReentrancyAttack() (gas: 3670768)
[PASS] testRegistryPauseButEndDurationReached() (gas: 9144214)
[PASS] testRegistryPauseStoppingAllFundActions() (gas: 3319148)
[PASS] testSettingBadRebalanceDeviation() (gas: 14259)
[PASS] testShutdown() (gas: 31916)
[PASS] testTotalAssets(uint256,uint256,uint256,uint256,uint256) (runs: 256, μ: 2702360, ~: 2702379)
[PASS] testTrustPositionForUnsupportedAssetLocksAllFunds() (gas: 805141)
[PASS] testWithdrawInOrder() (gas: 2756966)
[PASS] testWithdrawWithDuplicateReceivedAssets() (gas: 8371400)
[PASS] testWithdrawingWhileShutdown() (gas: 684527)
Test result: ok. 29 passed; 0 failed; 0 skipped; finished in 6.84s
```

```
Running 16 tests for
test/testAdaptors/FraxlendCollateralAndDebtV2.t.sol:FundFraxLendCollateralAndDebtTestV2
[PASS] testBlockExternalReceiver(uint256) (runs: 256, μ: 373722, ~: 373779)
[PASS] testDeposit(uint256) (runs: 256, μ: 594508, ~: 594607)
[PASS] testFailRemoveCollateralBecauseLTV(uint256) (runs: 256, μ: 1032466, ~: 982083)
[PASS] testLTV(uint256) (runs: 256, μ: 1556275, ~: 1556372)
[PASS] testLoanInUntrackedPosition(uint256) (runs: 256, μ: 862883, ~: 862966)
[PASS] testMultipleFraxlendPositions() (gas: 3081871)
[PASS] testRemoveAllCollateralWithTypeUINT256Max(uint256) (runs: 256, μ: 739527, ~: 739653)
[PASS] testRemoveCollateral(uint256) (runs: 256, μ: 739463, ~: 739548)
[PASS] testRemoveCollateralWithTypeUINT256MaxAfterRepay(uint256) (runs: 256, μ: 1453402, ~: 1453510)
[PASS] testRemoveSomeCollateral(uint256) (runs: 256, μ: 769819, ~: 769909)
[PASS] testRepayPartialDebt(uint256) (runs: 256, μ: 1237113, ~: 1237220)
[PASS] testRepayingDebtThatIsNotOwed(uint256) (runs: 256, μ: 385686, ~: 385739)
[PASS] testRepayingLoans(uint256) (runs: 256, μ: 1288129, ~: 1288218)
[PASS] testTakingOutLoanInUntrackedPositionV2(uint256) (runs: 256, μ: 457592, ~: 457689)
[PASS] testTakingOutLoansV2(uint256) (runs: 256, μ: 1009044, ~: 1009133)
[PASS] testTotalAssets(uint256) (runs: 256, μ: 649315, ~: 649439)
Test result: ok. 16 passed; 0 failed; 0 skipped; finished in 14.54s

Running 16 tests for
test/testAdaptors/FraxlendCollateralAndDebtV1.t.sol:FundFraxLendCollateralAndDebtTestV1
[PASS] testBlockExternalReceiver(uint256) (runs: 256, μ: 355112, ~: 355178)
[PASS] testDeposit(uint256) (runs: 256, μ: 568379, ~: 568485)
[PASS] testFailRemoveCollateralBecauseLTV(uint256) (runs: 256, μ: 1126416, ~: 1126522)
[PASS] testLTV(uint256) (runs: 256, μ: 1283695, ~: 1283807)
[PASS] testLoanInUntrackedPosition(uint256) (runs: 256, μ: 838962, ~: 839057)
[PASS] testMultipleFraxlendPositions() (gas: 2762599)
[PASS] testRemoveAllCollateralWithTypeUINT256Max(uint256) (runs: 256, μ: 691151, ~: 691253)
[PASS] testRemoveCollateral(uint256) (runs: 256, μ: 691794, ~: 691902)
[PASS] testRemoveCollateralWithTypeUINT256MaxAfterRepay(uint256) (runs: 256, μ: 1276124, ~: 1276223)
[PASS] testRemoveSomeCollateral(uint256) (runs: 256, μ: 721329, ~: 721424)
[PASS] testRepayPartialDebt(uint256) (runs: 256, μ: 1071367, ~: 1071470)
[PASS] testRepayingDebtThatIsNotOwed(uint256) (runs: 256, μ: 361336, ~: 361398)
[PASS] testRepayingLoans(uint256) (runs: 256, μ: 1125345, ~: 1125445)
[PASS] testTakingOutLoanInUntrackedPositionV1(uint256) (runs: 256, μ: 424933, ~: 425035)
[PASS] testTakingOutLoansV1(uint256) (runs: 256, μ: 860579, ~: 860676)
[PASS] testTotalAssets(uint256) (runs: 256, μ: 620246, ~: 620356)
Test result: ok. 16 passed; 0 failed; 0 skipped; finished in 14.88s

Running 28 tests for test/testAdaptors/BalancerPoolAdaptor.t.sol:BalancerPoolAdaptorTest
[PASS] testAssetsUsed() (gas: 14598)
[PASS] testBalancerFlashLoanChecks() (gas: 32440)
[PASS] testBalancerFlashLoans() (gas: 723942)
[PASS] testClaimRewards() (gas: 1830583)
[PASS] testConstructorReverts() (gas: 80238)
[PASS] testDepositToHoldingPosition() (gas: 5887879)
[PASS] testExitBoostedPool(uint256) (runs: 256, μ: 1485990, ~: 1486439)
[PASS] testExitBoostedPoolProportional(uint256) (runs: 256, μ: 1781239, ~: 1781316)
[PASS] testExitPoolReverts() (gas: 2380807)
[PASS] testExitPoolSlippageCheck(uint256) (runs: 256, μ: 1313390, ~: 1313512)
[PASS] testExitVanillaPool(uint256) (runs: 256, μ: 1272340, ~: 1268407)
[PASS] testExitVanillaPoolProportional(uint256) (runs: 256, μ: 1392024, ~: 1392165)
[PASS] testFailTransferEthToFund() (gas: 18419)
[PASS] testIsDebt() (gas: 10626)
[PASS] testJoinBoostedPool(uint256) (runs: 256, μ: 1174990, ~: 1175028)
[PASS] testJoinBoostedPoolWithMultipleTokens(uint256) (runs: 256, μ: 1920987, ~: 1920989)
[PASS] testJoinPoolNoSwapsReverts() (gas: 1201772)
[PASS] testJoinPoolSlippageCheck(uint256) (runs: 256, μ: 1129977, ~: 1130152)
[PASS] testJoinPoolWithSwapsReverts() (gas: 1097566)
[PASS] testJoinVanillaPool(uint256) (runs: 256, μ: 1041024, ~: 1050670)
[PASS] testJoinVanillaPoolWithMultiTokens(uint256) (runs: 256, μ: 1517241, ~: 1517310)
[PASS] testNonStableCoinJoinMultiTokens(uint256) (runs: 256, μ: 1348584, ~: 1342723)
[PASS] testStakeBpt(uint256) (runs: 256, μ: 1491861, ~: 1491998)
[PASS] testStakeUint256Max(uint256) (runs: 256, μ: 1492872, ~: 1492989)
[PASS] testTotalAssets(uint256) (runs: 256, μ: 1974973, ~: 1975050)
[PASS] testUnstakeBpt(uint256) (runs: 256, μ: 1461863, ~: 1461998)
[PASS] testUnstakeUint256Max(uint256) (runs: 256, μ: 1462438, ~: 1462556)
[PASS] testUserWithdrawPullFromGauge(uint256,uint256) (runs: 256, μ: 2110122, ~: 2090764)
Test result: ok. 28 passed; 0 failed; 0 skipped; finished in 29.17s
```

```
Running 17 tests for test/testAdaptors/AaveV3.t.sol:FundAaveV3Test
[PASS] testBlockExternalReceiver() (gas: 454286)
[PASS] testDeposit(uint256) (runs: 256, μ: 441577, ~: 441666)
[PASS] testIntegration() (gas: 5408988)
[PASS] testMultipleATokensAndDebtTokens() (gas: 2818786)
[PASS] testRepayingDebtThatIsNotOwed() (gas: 484717)
[PASS] testRepayingLoans() (gas: 835326)
[PASS] testTakingOutAFlashLoan() (gas: 873350)
[PASS] testTakingOutLoanInUntrackedPosition() (gas: 458375)
[PASS] testTakingOutLoans() (gas: 736371)
[PASS] testTakingOutLoansInUntrackedPosition() (gas: 532705)
[PASS] testTotalAssets(uint256) (runs: 256, μ: 470730, ~: 470811)
[PASS] testWithdraw(uint256) (runs: 256, μ: 745141, ~: 745238)
[PASS] testWithdrawableFromaV3USDC() (gas: 1257851)
[PASS] testWithdrawableFromaV3WETH() (gas: 2439972)
[PASS] testWithdrawalLogicEModeWithDebt() (gas: 2369576)
[PASS] testWithdrawalLogicNoDebt() (gas: 2446086)
[PASS] testWithdrawalLogicNoEModeWithDebt() (gas: 2647031)
Test result: ok. 17 passed; 0 failed; 0 skipped; finished in 31.51s

Running 12 tests for test/testAdaptors/AaveV3Morpho.t.sol:FundAaveV3MorphoTest
[PASS] testBlockExternalReceiver(uint256) (runs: 256, μ: 3435387, ~: 4681074)
[PASS] testDeposit(uint256) (runs: 256, μ: 459635, ~: 459684)
[PASS] testHealthFactor(uint256) (runs: 256, μ: 2820775, ~: 2820871)
[PASS] testHealthFactorChecks() (gas: 3683783)
[PASS] testIntegrationRealYieldEth(uint256) (runs: 256, μ: 5845039, ~: 5845039)
[PASS] testRepayingDebtThatIsNotOwed(uint256) (runs: 256, μ: 988851, ~: 988940)
[PASS] testRepayingLoans(uint256) (runs: 256, μ: 4943735, ~: 5909495)
[PASS] testTakingOutLoans(uint256) (runs: 256, μ: 4618337, ~: 5559055)
[PASS] testTakingOutLoansInUntrackedPosition(uint256) (runs: 256, μ: 3254415, ~: 4107350)
[PASS] testTotalAssets(uint256) (runs: 256, μ: 509840, ~: 509885)
[PASS] testWithdraw(uint256) (runs: 256, μ: 683862, ~: 688504)
[PASS] testWithdrawalLogic(uint256) (runs: 256, μ: 4120417, ~: 4231343)
Test result: ok. 12 passed; 0 failed; 0 skipped; finished in 33.01s

(base) guillermoescobero@MacBook-Pro-2468 swaap_labs-swaap-earn-protocol-main-github % forge test
[⠢] Compiling...
No files changed, compilation skipped

Running 4 tests for test/FundWithShareLockPeriod.t.sol:FundWithShareLockPeriodTest
[PASS] testChainlinkPriceFeedUpdateSandwichAttack() (gas: 6221930)
[PASS] testDepositOnBehalf() (gas: 414544)
[PASS] testShareLockUpPeriod() (gas: 1374756)
[PASS] testTransfer() (gas: 433092)
Test result: ok. 4 passed; 0 failed; 0 skipped; finished in 903.02ms

Running 5 tests for test/testAdaptors/OneInch.t.sol:FundOneInchTest
[PASS] testOneInchSwap() (gas: 533632)
[PASS] testRevertAggregatorSwapIfTotalVolumeIsSurpassed() (gas: 768249)
[PASS] testRevertForUnsupportedAssets() (gas: 778859)
[PASS] testSlippageChecks() (gas: 3756709)
[PASS] testVolumeIsIncrementedCorrectly() (gas: 1719983)
Test result: ok. 5 passed; 0 failed; 0 skipped; finished in 405.40ms

Running 17 tests for test/testPriceRouter/PriceRouter.t.sol:PriceRouterTest
[PASS] testAddAssetWithInvalidMaxPrice() (gas: 169602)
[PASS] testAddAssetWithInvalidMinPrice() (gas: 169415)
[PASS] testAddChainlinkAsset() (gas: 257349)
[PASS] testAddInvalidAsset() (gas: 21551)
[PASS] testAddingATwapAsset() (gas: 16351715)
[PASS] testAssetAboveMaxPrice() (gas: 187718)
[PASS] testAssetBelowMinPrice() (gas: 187995)
[PASS] testAssetStalePrice() (gas: 46630)
[PASS] testETHtoUSDPriceFeedIsChecked() (gas: 379483)
[PASS] testEditAsset() (gas: 110944)
[PASS] testExchangeRate() (gas: 16711011)
[PASS] testGetValue(uint256,uint256,uint256) (runs: 256, μ: 309361, ~: 309373)
[PASS] testMinPriceGreaterThanMaxPrice() (gas: 171659)
[PASS] testNumericError() (gas: 223360)
[PASS] testTransitioningOwner() (gas: 215284)
[PASS] testUnsupportedAsset() (gas: 122133)
[PASS] testWrongChainlinkDecimals() (gas: 84956)
```

```
Test result: ok. 17 passed; 0 failed; 0 skipped; finished in 1.80s

Running 5 tests for test/testAdaptors/Paraswap.t.sol:FundParaswapTest
[PASS] testParaswapSwap() (gas: 540916)
[PASS] testRevertAggregatorSwapIfTotalVolumeIsSurpassed() (gas: 764763)
[PASS] testRevertForUnsupportedAssets() (gas: 778813)
[PASS] testSlippageChecks() (gas: 3748777)
[PASS] testVolumeIsIncrementedCorrectly() (gas: 1716041)
Test result: ok. 5 passed; 0 failed; 0 skipped; finished in 622.66ms

Running 5 tests for test/testPriceRouter/RedstonePriceFeedExtension.t.sol:RedstonePriceFeedExtensionTest
[PASS] testRedstonePriceFeedExtension() (gas: 319859)
[PASS] testRedstonePriceFeedExtensionSwEth() (gas: 219918)
[PASS] testUsingExtensionWithStalePrice() (gas: 177628)
[PASS] testUsingExtensionWithWrongDataFeedId() (gas: 168751)
[PASS] testUsingExtensionWithZeroPrice() (gas: 158051)
Test result: ok. 5 passed; 0 failed; 0 skipped; finished in 414.87ms

Running 6 tests for
test/testPriceRouter/RedstonePriceFeedExtensionEth.t.sol:RedstoneEthPriceFeedExtensionTest
[PASS] testRedstoneEthPriceFeedExtension() (gas: 424577)
[PASS] testUsingActualSwethEthFeed() (gas: 214997)
[PASS] testUsingExtensionWithStalePrice() (gas: 265524)
[PASS] testUsingExtensionWithWrongDataFeedId() (gas: 260619)
[PASS] testUsingExtensionWithZeroPrice() (gas: 245320)
[PASS] testUsingExtensionWithoutPriceRouterSupportingWETH() (gas: 266894)
Test result: ok. 6 passed; 0 failed; 0 skipped; finished in 588.99ms

Running 8 tests for test/Registry.t.sol:RegistryTest
[PASS] testInitialization() (gas: 28771)
[PASS] testRegister() (gas: 46044)
[PASS] testSetAddress() (gas: 58251)
[PASS] testSetAddressOfInvalidId() (gas: 19867)
[PASS] testSetApprovedForDepositOnBehalf() (gas: 33600)
[PASS] testTransitioningOwner() (gas: 204577)
[PASS] testTrustingAndDistrustingAdaptor() (gas: 519818)
[PASS] testTrustingAndDistrustingPosition() (gas: 705778)
Test result: ok. 8 passed; 0 failed; 0 skipped; finished in 7.15ms

Running 1 test for test/testPriceRouter/SequencerPriceRouter.t.sol:SequencerPriceRouterTest
[PASS] testSequencerUptimeFeedLogic() (gas: 108783)
Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 601.50ms

Running 17 tests for test/testAdaptors/AaveV3.t.sol:FundAaveV3Test
[PASS] testBlockExternalReceiver() (gas: 454286)
[PASS] testDeposit(uint256) (runs: 256, μ: 441586, ~: 441672)
[PASS] testIntegration() (gas: 5408988)
[PASS] testMultipleATokensAndDebtTokens() (gas: 2818786)
[PASS] testRepayingDebtThatIsNotOwed() (gas: 484717)
[PASS] testRepayingLoans() (gas: 835326)
[PASS] testTakingOutAFlashLoan() (gas: 873350)
[PASS] testTakingOutLoanInUntrackedPosition() (gas: 458375)
[PASS] testTakingOutLoans() (gas: 736371)
[PASS] testTakingOutLoansInUntrackedPosition() (gas: 532705)
[PASS] testTotalAssets(uint256) (runs: 256, μ: 470711, ~: 470804)
[PASS] testWithdraw(uint256) (runs: 256, μ: 745158, ~: 745249)
[PASS] testWithdrawableFromaV3USDC() (gas: 1257851)
[PASS] testWithdrawableFromaV3WETH() (gas: 2439972)
[PASS] testWithdrawalLogicEModeWithDebt() (gas: 2369576)
[PASS] testWithdrawalLogicNoDebt() (gas: 2446086)
[PASS] testWithdrawalLogicNoEModeWithDebt() (gas: 2647031)
Test result: ok. 17 passed; 0 failed; 0 skipped; finished in 4.36s

Running 21 tests for test/FeesManager.t.sol:FeesManagerTest
[PASS] testCollectFeesFromFund() (gas: 702046)
[PASS] testCollectFeesWhenSettingManagementFees() (gas: 704160)
[PASS] testCollectFeesWhenSettingPerformanceFees() (gas: 1212337)
[PASS] testDepositAndMintSharePriceAreEqualWithEnterFeesOn() (gas: 579941)
[PASS] testEnterFeesDepositHook() (gas: 926333)
[PASS] testEnterFeesMintHook() (gas: 951835)
[PASS] testExitFeesRedeemHook(uint16) (runs: 256, μ: 975469, ~: 975709)
[PASS] testExitFeesWithdrawHook(uint16) (runs: 256, μ: 801801, ~: 802084)
```

```
[PASS] testFeesPayoutWithStrategistAddressAndCutUnset() (gas: 627818)
[PASS] testFeesPayoutWithStrategistAddressUnset() (gas: 808027)
[PASS] testFeesPayoutWithStrategistPayoutAndCutSet() (gas: 1000848)
[PASS] testHighWaterMarkReset() (gas: 1875443)
[PASS] testManagementFeesEnterHook() (gas: 1259784)
[PASS] testPerformanceFeesDepositHook() (gas: 1880272)
[PASS] testPerformanceFeesMintHook() (gas: 1722740)
[PASS] testPerformanceFeesRedeemHook() (gas: 1565626)
[PASS] testPerformanceFeesWithdrawHook() (gas: 1564958)
[PASS] testRevertOnWrongCaller() (gas: 60874)
[PASS] testRevertOnWrongFeesInputs() (gas: 44052)
[PASS] testUpdateFeesRatesCorrectly() (gas: 1210289)
[PASS] testWithdrawAndRedeemSharePriceAreEqualWithExitFeesOn(uint256,uint16) (runs: 256, μ: 588849, ~:
588722)
Test result: ok. 21 passed; 0 failed; 0 skipped; finished in 4.77s

Running 3 tests for test/FundWithERC4626Adaptor.t.sol:FundWithERC4626AdaptorTest
[PASS] testFundWithFundPositions() (gas: 11089980)
[PASS] testTotalAssets(uint256,uint256,uint256,uint256,uint256) (runs: 256, μ: 2702559, ~: 2702557)
[PASS] testUsingIlliquidFundPosition() (gas: 6678519)
Test result: ok. 3 passed; 0 failed; 0 skipped; finished in 6.20s

Running 9 tests for
test/FundWithShareLockFlashLoansWhitelisting.t.sol:MockFundWithShareLockFlashLoansWhitelistingTest
[PASS] testRandomUserTurnWhitelistOff() (gas: 20229)
[PASS] testRandomUserTurnWhitelistOn() (gas: 16753)
[PASS] testSetUpState() (gas: 11694)
[PASS] testSignatureVerificationAsFundAutomationActions() (gas: 138893)
[PASS] testSignatureVerificationAsFundOwner() (gas: 78817)
[PASS] testUserDelayAboveValidity() (gas: 141357)
[PASS] testWhitelistOff() (gas: 708663)
[PASS] testWhitelistOnAndThenOff() (gas: 726515)
[PASS] testWrongSignatureWhitelistOff() (gas: 17334)
Test result: ok. 9 passed; 0 failed; 0 skipped; finished in 32.33ms

Running 21 tests for test/testAdaptors/AaveV3Manager.t.sol:FundAaveV3Test
[PASS] testBlockExternalReceiver() (gas: 195736)
[PASS] testCreateAaveAccountExtension() (gas: 1111280)
[PASS] testDefaultAaveAccountDeployedOnInitDeposit() (gas: 15720)
[PASS] testDeposit(uint256) (runs: 256, μ: 495111, ~: 495198)
[PASS] testIntegration() (gas: 5590646)
[PASS] testMultipleATokensAndDebtTokens() (gas: 2943575)
[PASS] testRepayingDebtThatIsNotOwed() (gas: 501428)
[PASS] testRepayingLoans() (gas: 903045)
[PASS] testRevertWhenDeployAaveAccountWithIncorrectData() (gas: 205608)
[PASS] testTakingOutAFlashLoan() (gas: 942813)
[PASS] testTakingOutLoanInUntrackedPosition() (gas: 475740)
[PASS] testTakingOutLoans() (gas: 800143)
[PASS] testTakingOutLoansInUntrackedPosition() (gas: 555612)
[PASS] testTotalAssets(uint256) (runs: 256, μ: 488349, ~: 488429)
[PASS] testWithdraw(uint256) (runs: 256, μ: 745153, ~: 745243)
[PASS] testWithdrawableFromaV3USDC() (gas: 1321962)
[PASS] testWithdrawableFromaV3WETH() (gas: 2524730)
[PASS] testWithdrawalLogicEModeWithDebt() (gas: 2973984)
[PASS] testWithdrawalLogicEmodeNoDebt() (gas: 3170074)
[PASS] testWithdrawalLogicNoEModeNoDebt() (gas: 1801743)
[PASS] testWithdrawalLogicNoEModeWithDebt() (gas: 2747471)
Test result: ok. 21 passed; 0 failed; 0 skipped; finished in 2.96s

Running 9 tests for test/SimpleSlippageRouter.t.sol:SimpleSlippageRouterTest
[PASS] testBadDeadline(uint256) (runs: 256, μ: 193916, ~: 194010)
[PASS] testDeposit(uint256) (runs: 256, μ: 480592, ~: 480670)
[PASS] testDepositMinimumSharesUnmet(uint256) (runs: 256, μ: 503691, ~: 503777)
[PASS] testMint(uint256) (runs: 256, μ: 529266, ~: 529352)
[PASS] testMintMaxAssetsRqdSurpassed(uint256) (runs: 256, μ: 675587, ~: 675661)
[PASS] testRedeem(uint256) (runs: 256, μ: 601569, ~: 601657)
[PASS] testRedeemMinAssetsUnmet(uint256) (runs: 256, μ: 540905, ~: 540985)
[PASS] testWithdraw(uint256) (runs: 256, μ: 550084, ~: 550175)
[PASS] testWithdrawMaxSharesSurpassed(uint256) (runs: 256, μ: 530875, ~: 530957)
Test result: ok. 9 passed; 0 failed; 0 skipped; finished in 7.46s

Running 5 tests for test/testAdaptors/CellarAdaptorWithSDai.t.sol:SwaapFundAdaptorWithSDaiTest
```

```
[PASS] testDeposit(uint256) (runs: 256, μ: 488190, ~: 488269)
[PASS] testInterestAccrual(uint256) (runs: 256, μ: 538008, ~: 538081)
[PASS] testStrategistFunctions(uint256) (runs: 256, μ: 946580, ~: 946703)
[PASS] testUsersGetPendingInterest(uint256) (runs: 256, μ: 792605, ~: 793517)
[PASS] testWithdraw(uint256) (runs: 256, μ: 647777, ~: 648017)
Test result: ok. 5 passed; 0 failed; 0 skipped; finished in 7.83s

Running 6 tests for test/testPriceRouter/StEthExtension.t.sol:StEthExtensionTest
[PASS] testAnswersDiverging() (gas: 646521)
[PASS] testChainlinkReverts() (gas: 324741)
[PASS] testStEthExtension() (gas: 294718)
[PASS] testUniswapOracleFailuresDefaultingToChainlinkIfMeanLiquidityLow() (gas: 11849613)
[PASS] testUniswapOracleFailuresDefaultingToChainlinkIfObservationsToNew() (gas: 11815639)
[PASS] testUsingExtensionWithWrongAsset() (gas: 47714)
Test result: ok. 6 passed; 0 failed; 0 skipped; finished in 724.75ms

Running 23 tests for test/testAdaptors/UniswapV3.t.sol:UniswapV3AdaptorTest
[PASS] testAddingPositionWithUnsupportedToken0Reverts() (gas: 203122)
[PASS] testAddingPositionWithUnsupportedToken1Reverts() (gas: 285285)
[PASS] testAddingToExistingPosition() (gas: 1557262)
[PASS] testFundAddingAndRemovingPositionReverts() (gas: 2077435)
[PASS] testFundPurgingSinglePositionsAndAllUnusedPositions() (gas: 9568113)
[PASS] testFundWithSmorgasbordOfUniV3Positions() (gas: 3788461)
[PASS] testGriefingAttack() (gas: 5103646)
[PASS] testHandlingUnusedApprovals() (gas: 1998658)
[PASS] testIdsAreIgnoredIfNotOwnedByFund() (gas: 2181904)
[PASS] testIntegration() (gas: 30714820)
[PASS] testIsDebtReturnsFalse() (gas: 7133)
[PASS] testOpenUSDC_DAIPosition() (gas: 1126686)
[PASS] testOpenUSDC_WETHPosition() (gas: 1246355)
[PASS] testOpeningAndClosingUniV3Position() (gas: 1308924)
[PASS] testPositionBurning() (gas: 2757660)
[PASS] testRangeOrders() (gas: 1034144)
[PASS] testTakingFees() (gas: 2325039)
[PASS] testTakingFromExistingPosition() (gas: 1463115)
[PASS] testUserDepositAndWithdrawRevert() (gas: 18052)
[PASS] testUsingLPTokensNotOwnedByFundOrTokensThatDoNotExist() (gas: 645776)
[PASS] testUsingUntrackedLPPosition() (gas: 437080)
[PASS] testWithdrawableFromReturnsZero() (gas: 12773)
[PASS] testWorkingWithMaxNumberOfTrackedTokens() (gas: 31402955)
Test result: ok. 23 passed; 0 failed; 0 skipped; finished in 1.01s

Running 15 tests for test/testAdaptors/Aave.t.sol:FundAaveTest
[PASS] testAddingPositionWithUnsupportedAssetsReverts() (gas: 444670)
[PASS] testBlockExternalReceiver() (gas: 471240)
[PASS] testDeposit(uint256) (runs: 256, μ: 458738, ~: 458821)
[PASS] testIntegration() (gas: 5979365)
[PASS] testMultipleATokensAndDebtTokens() (gas: 3148837)
[PASS] testRepayingDebtThatIsNotOwed() (gas: 497283)
[PASS] testRepayingLoans(uint256) (runs: 256, μ: 925619, ~: 924314)
[PASS] testTakingOutAFlashLoan() (gas: 973540)
[PASS] testTakingOutLoanInUntrackedPosition() (gas: 475333)
[PASS] testTakingOutLoans(uint256) (runs: 256, μ: 794649, ~: 794774)
[PASS] testTakingOutLoansInUntrackedPosition() (gas: 550695)
[PASS] testTotalAssets(uint256) (runs: 256, μ: 487821, ~: 487915)
[PASS] testWithdraw(uint256) (runs: 256, μ: 809454, ~: 809543)
[PASS] testWithdrawableFromaV2USDC() (gas: 1481364)
[PASS] testWithdrawableFromaV2WETH() (gas: 2772977)
Test result: ok. 15 passed; 0 failed; 0 skipped; finished in 8.54s

Running 4 tests for test/testAdaptors/AggregatorBaseAdaptor.t.sol:FundAggregatorBaseAdaptorTest
[PASS] testRevertAggregatorSwapIfTotalVolumeIsSurpassed() (gas: 768047)
[PASS] testRevertForUnsupportedAssets() (gas: 778537)
[PASS] testSlippageChecks() (gas: 3754458)
[PASS] testVolumeIsIncrementedCorrectly() (gas: 1719287)
Test result: ok. 4 passed; 0 failed; 0 skipped; finished in 37.66ms

Running 5 tests for test/testAdaptors/SushiswapV3.t.sol:SushiswapV3AdaptorTest
[PASS] testAddingToExistingPosition() (gas: 1583727)
[PASS] testOpenUSDC_DAIPosition() (gas: 1271264)
[PASS] testOpeningAndClosingUniV3Position() (gas: 1428168)
[PASS] testRangeOrders() (gas: 1032416)
```

```
[PASS] testTakingFromExistingPosition() (gas: 1526250)
Test result: ok. 5 passed; 0 failed; 0 skipped; finished in 407.19ms

Running 7 tests for test/testAdaptors/Compound.t.sol:FundCompoundTest
[PASS] testAddingPositionWithUnsupportedAssetsReverts() (gas: 438478)
[PASS] testClaimCompAndVest() (gas: 3107917)
[PASS] testDeposit(uint256) (runs: 256, μ: 449838, ~: 449910)
[PASS] testErrorCodeCheck() (gas: 1238090)
[PASS] testMaliciousStrategistMovingFundsIntoUntrackedCompoundPosition() (gas: 4017667)
[PASS] testTotalAssets() (gas: 1030155)
[PASS] testWithdraw(uint256) (runs: 256, μ: 731745, ~: 731813)
Test result: ok. 7 passed; 0 failed; 0 skipped; finished in 2.60s

Running 3 tests for test/testPriceRouter/SwaapSafeguardPool.t.sol:SwaapSafeguarPoolTest
[PASS] testPoolPricingWithManagementFees(uint256) (runs: 256, μ: 676418, ~: 676511)
[PASS] testPricingSafeguardPoolWithUnsupportedPool() (gas: 26980)
[PASS] testPricingUSDC_WETH_SPT(uint256) (runs: 256, μ: 846138, ~: 846271)
Test result: ok. 3 passed; 0 failed; 0 skipped; finished in 4.40s

Running 9 tests for test/testAdaptors/Vesting.t.sol:FundVestingTest
[PASS] testCannotTakeUserDeposits(uint256) (runs: 256, μ: 372713, ~: 372800)
[PASS] testDepositAndWithdrawReturnsZero(uint256) (runs: 256, μ: 455831, ~: 455977)
[PASS] testDepositToVesting(uint256) (runs: 256, μ: 465671, ~: 465814)
[PASS] testFailWithdrawMoreThanVested(uint256) (runs: 256, μ: 529968, ~: 530109)
[PASS] testStrategistPartialWithdrawFromVesting(uint256) (runs: 256, μ: 605846, ~: 605977)
[PASS] testStrategistWithdrawAllFromVesting(uint256) (runs: 256, μ: 631743, ~: 631901)
[PASS] testStrategistWithdrawAnyFromVesting(uint256) (runs: 256, μ: 505191, ~: 505284)
[PASS] testStrategistWithdrawFromVesting(uint256) (runs: 256, μ: 505720, ~: 505798)
[PASS] testUserWithdrawFromVesting(uint256) (runs: 256, μ: 644957, ~: 648452)
Test result: ok. 9 passed; 0 failed; 0 skipped; finished in 5.30s

Running 1 test for test/testAdaptors/CellarAdaptor.t.sol:SwaapFundAdaptorTest
[PASS] testUsingIlliquidFundPosition() (gas: 5879170)
Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 13.58ms

Running 6 tests for test/testAdaptors/DSRAdaptor.t.sol:FundDSRTest
[PASS] testDeposit(uint256) (runs: 256, μ: 398519, ~: 398592)
[PASS] testDrip(uint256) (runs: 256, μ: 589118, ~: 589229)
[PASS] testInterestAccrual(uint256) (runs: 256, μ: 544300, ~: 544381)
[PASS] testStrategistFunctions(uint256) (runs: 256, μ: 851309, ~: 851452)
[PASS] testUsersDoNotGetPendingInterest(uint256) (runs: 256, μ: 678364, ~: 678477)
[PASS] testWithdraw(uint256) (runs: 256, μ: 537623, ~: 537690)
Test result: ok. 6 passed; 0 failed; 0 skipped; finished in 7.78s

Running 12 tests for test/VestingSimple.t.sol:VestingTest
[PASS] testDepositOnBehalf(uint256,uint256) (runs: 256, μ: 336261, ~: 336261)
[PASS] testFailDepositLessThanMinimum() (gas: 8831)
[PASS] testFailDepositZero() (gas: 11456)
[PASS] testFullVest(uint256) (runs: 256, μ: 267196, ~: 267196)
[PASS] testFullVestWithPartialClaim(uint256,uint256) (runs: 256, μ: 292723, ~: 292727)
[PASS] testInitialization() (gas: 12017)
[PASS] testMultipleClaims(uint256,uint256) (runs: 256, μ: 339154, ~: 339157)
[PASS] testMultipleDeposits(uint256,uint256) (runs: 256, μ: 516496, ~: 516496)
[PASS] testMultipleUsers(uint256,uint256) (runs: 256, μ: 705037, ~: 705039)
[PASS] testPartialVest(uint256,uint256) (runs: 256, μ: 314578, ~: 314578)
[PASS] testPartialVestWithPartialClaim(uint256,uint256,uint256) (runs: 256, μ: 356088, ~: 356088)
[PASS] testWithdrawAnyFor(uint256,uint256) (runs: 256, μ: 559562, ~: 559560)
Test result: ok. 12 passed; 0 failed; 0 skipped; finished in 2.60s

Running 3 tests for test/testPriceRouter/WstEthExtension.t.sol:WstEthExtensionTest
[PASS] testAddingWstethWithoutPricingSteth() (gas: 92519)
[PASS] testUsingExtensionWithWrongAsset() (gas: 88072)
[PASS] testWstEthExtension() (gas: 220080)
Test result: ok. 3 passed; 0 failed; 0 skipped; finished in 11.73ms

Running 2 tests for test/testAdaptors/ZeroX.t.sol:FundZeroXTest
[PASS] test0xSwap() (gas: 831499)
[PASS] testSlippageChecks() (gas: 3883999)
Test result: ok. 2 passed; 0 failed; 0 skipped; finished in 632.15ms

Running 1 test for test/testAdaptors/ERC20Adaptor.t.sol:ERC20AdaptorTest
[PASS] testLogic(uint256,uint256) (runs: 256, μ: 797330, ~: 800696)
```

```
Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 931.11ms

Running 2 tests for test/testPriceRouter/ERC4626Extension.t.sol:ERC4626ExtensionTest
[PASS] testERC4626ExtensionSDai() (gas: 231293)
[PASS] testUsingExtensionWithUnsupportedAsset() (gas: 196339)
Test result: ok. 2 passed; 0 failed; 0 skipped; finished in 9.58ms

Running 7 tests for test/testPriceRouter/BalancerStablePool.t.sol:BalancerStablePoolTest
[PASS] testMisConfiguredStorageData() (gas: 653507)
[PASS] testPricingBb_a_Usd() (gas: 1534923)
[PASS] testPricingCBETH_WSTETH_Bpt(uint256) (runs: 256, μ: 1212266, ~: 1212371)
[PASS] testPricingRETH_WETH_Bpt(uint256) (runs: 256, μ: 1008231, ~: 1009999)
[PASS] testPricingStablePoolWithUnsupportedUnderlying() (gas: 216578)
[PASS] testPricingUSDC_DAI_USDT_Bpt(uint256) (runs: 256, μ: 1042817, ~: 1048582)
[PASS] testPricingWstETH_WETH_Bpt(uint256) (runs: 256, μ: 877256, ~: 877983)
Test result: ok. 7 passed; 0 failed; 0 skipped; finished in 9.11s

Running 16 tests for
test/testAdaptors/FraxlendCollateralAndDebtV1.t.sol:FundFraxLendCollateralAndDebtTestV1
[PASS] testBlockExternalReceiver(uint256) (runs: 256, μ: 355110, ~: 355181)
[PASS] testDeposit(uint256) (runs: 256, μ: 568377, ~: 568485)
[PASS] testFailRemoveCollateralBecauseLTV(uint256) (runs: 256, μ: 1126416, ~: 1126522)
[PASS] testLTV(uint256) (runs: 256, μ: 1283699, ~: 1283807)
[PASS] testLoanInUntrackedPosition(uint256) (runs: 256, μ: 838952, ~: 839057)
[PASS] testMultipleFraxlendPositions() (gas: 2762599)
[PASS] testRemoveAllCollateralWithTypeUINT256Max(uint256) (runs: 256, μ: 691150, ~: 691253)
[PASS] testRemoveCollateral(uint256) (runs: 256, μ: 691798, ~: 691902)
[PASS] testRemoveCollateralWithTypeUINT256MaxAfterRepay(uint256) (runs: 256, μ: 1276112, ~: 1276223)
[PASS] testRemoveSomeCollateral(uint256) (runs: 256, μ: 721312, ~: 721424)
[PASS] testRepayPartialDebt(uint256) (runs: 256, μ: 1071343, ~: 1071470)
[PASS] testRepayingDebtThatIsNotOwed(uint256) (runs: 256, μ: 361336, ~: 361396)
[PASS] testRepayingLoans(uint256) (runs: 256, μ: 1125325, ~: 1125445)
[PASS] testTakingOutLoanInUntrackedPositionV1(uint256) (runs: 256, μ: 424920, ~: 425035)
[PASS] testTakingOutLoansV1(uint256) (runs: 256, μ: 860556, ~: 860676)
[PASS] testTotalAssets(uint256) (runs: 256, μ: 620248, ~: 620356)
Test result: ok. 16 passed; 0 failed; 0 skipped; finished in 22.47s

Running 12 tests for test/testAdaptors/AaveV3Morpho.t.sol:FundAaveV3MorphoTest
[PASS] testBlockExternalReceiver(uint256) (runs: 256, μ: 3480015, ~: 4681074)
[PASS] testDeposit(uint256) (runs: 256, μ: 459626, ~: 459688)
[PASS] testHealthFactor(uint256) (runs: 256, μ: 2820770, ~: 2820871)
[PASS] testHealthFactorChecks() (gas: 3683783)
[PASS] testIntegrationRealYieldEth(uint256) (runs: 256, μ: 5845039, ~: 5845039)
[PASS] testRepayingDebtThatIsNotOwed(uint256) (runs: 256, μ: 988834, ~: 988940)
[PASS] testRepayingLoans(uint256) (runs: 256, μ: 4950645, ~: 5909598)
[PASS] testTakingOutLoans(uint256) (runs: 256, μ: 4591929, ~: 5559055)
[PASS] testTakingOutLoansInUntrackedPosition(uint256) (runs: 256, μ: 3224840, ~: 4107407)
[PASS] testTotalAssets(uint256) (runs: 256, μ: 509836, ~: 509901)
[PASS] testWithdraw(uint256) (runs: 256, μ: 682291, ~: 688490)
[PASS] testWithdrawalLogic(uint256) (runs: 256, μ: 4094382, ~: 4231343)
Test result: ok. 12 passed; 0 failed; 0 skipped; finished in 30.74s

Running 12 tests for test/testAdaptors/SwaapV2Adaptor.t.sol:SwaapV2AdaptorTest
[PASS] testAllowlistJoin(uint256) (runs: 256, μ: 848839, ~: 848938)
[PASS] testAssetsUsed() (gas: 12014)
[PASS] testBalancerFlashLoanChecks() (gas: 35038)
[PASS] testDepositToHoldingPosition() (gas: 6221727)
[PASS] testExitPoolReverts() (gas: 767968)
[PASS] testFailTransferEthToFund() (gas: 18089)
[PASS] testIsDebt() (gas: 10142)
[PASS] testJoinPoolReverts() (gas: 936629)
[PASS] testSwaapFlashLoans() (gas: 674591)
[PASS] testSwaapProportionalExitPool(uint256) (runs: 256, μ: 841091, ~: 841228)
[PASS] testTotalAssetsAfterExit(uint256) (runs: 256, μ: 709973, ~: 710091)
[PASS] testTotalAssetsAfterJoin(uint256) (runs: 256, μ: 932507, ~: 932644)
Test result: ok. 12 passed; 0 failed; 0 skipped; finished in 17.52s

Running 29 tests for test/Fund.sol:FundTest
[PASS] testCachePriceRouter() (gas: 1653053)
[PASS] testCallerOfCallOnAdaptor() (gas: 1226513)
[PASS] testDebtTokensInFunds() (gas: 6708577)
[PASS] testDepeggedAssetNotUsedByFund() (gas: 1004538)
```

```
[PASS] testDepeggedAssetUsedByTheFund() (gas: 1412292)
[PASS] testDepeggedFundAsset() (gas: 2445013)
[PASS] testDepeggedHoldingPosition() (gas: 1979117)
[PASS] testDepositAndWithdraw(uint256) (runs: 256, μ: 2169445, ~: 2169281)
[PASS] testDepositMintWithdrawRedeemWithZeroInputs() (gas: 1804729)
[PASS] testEndPauseDurationButFundIsShutDownThenLiftShutdown() (gas: 6417148)
[PASS] testFundDNOSPerformanceFeesWithZeroShares() (gas: 1113311)
[PASS] testFundWithFundPositions() (gas: 11090191)
[PASS] testInitialization() (gas: 209289)
[PASS] testInteractingWithDistrustedAdaptors() (gas: 1286391)
[PASS] testInteractingWithDistrustedPositions() (gas: 492605)
[PASS] testLimits() (gas: 1431146)
[PASS] testManagingPositions() (gas: 2177768)
[PASS] testMintAndRedeem(uint256) (runs: 256, μ: 1768814, ~: 1768981)
[PASS] testProhibitedActionsWhileShutdown() (gas: 684607)
[PASS] testReentrancyAttack() (gas: 3670768)
[PASS] testRegistryPauseButEndDurationReached() (gas: 9144214)
[PASS] testRegistryPauseStoppingAllFundActions() (gas: 3319148)
[PASS] testSettingBadRebalanceDeviation() (gas: 14259)
[PASS] testShutdown() (gas: 31916)
[PASS] testTotalAssets(uint256,uint256,uint256,uint256,uint256) (runs: 256, μ: 2702391, ~: 2702388)
[PASS] testTrustPositionForUnsupportedAssetLocksAllFunds() (gas: 805141)
[PASS] testWithdrawInOrder() (gas: 2756966)
[PASS] testWithdrawWithDuplicateReceivedAssets() (gas: 8371400)
[PASS] testWithdrawingWhileShutdown() (gas: 684527)
Test result: ok. 29 passed; 0 failed; 0 skipped; finished in 6.47s

Running 13 tests for test/testAdaptors/AaveV2Morpho.t.sol:FundAaveV2MorphoTest
[PASS] testBlockExternalReceiver(uint256) (runs: 256, μ: 1375535, ~: 1375657)
[PASS] testDeposit(uint256) (runs: 256, μ: 525633, ~: 552949)
[PASS] testHealthFactorChecks() (gas: 2411158)
[PASS] testIntegrationRealYieldEth(uint256) (runs: 256, μ: 3168397, ~: 3168397)
[PASS] testIntegrationRealYieldUsd(uint256) (runs: 256, μ: 7412233, ~: 7413232)
[PASS] testIsBorrowingAnyFullRepay(uint256) (runs: 256, μ: 1923162, ~: 1918970)
[PASS] testRepayingDebtThatIsNotOwed(uint256) (runs: 256, μ: 1064423, ~: 1064532)
[PASS] testRepayingLoans(uint256) (runs: 256, μ: 2822391, ~: 2822497)
[PASS] testTakingOutLoans(uint256) (runs: 256, μ: 2614009, ~: 2614109)
[PASS] testTakingOutLoansInUntrackedPosition(uint256) (runs: 256, μ: 1428847, ~: 1428954)
[PASS] testTotalAssets(uint256) (runs: 256, μ: 572480, ~: 607804)
[PASS] testWithdraw(uint256) (runs: 256, μ: 1184545, ~: 1208043)
[PASS] testWithdrawalLogic(uint256) (runs: 256, μ: 2054657, ~: 2050315)
Test result: ok. 13 passed; 0 failed; 0 skipped; finished in 31.25s

Running 16 tests for
test/testAdaptors/FraxlendCollateralAndDebtV2.t.sol:FundFraxLendCollateralAndDebtTestV2
[PASS] testBlockExternalReceiver(uint256) (runs: 256, μ: 373728, ~: 373792)
[PASS] testDeposit(uint256) (runs: 256, μ: 594485, ~: 594607)
[PASS] testFailRemoveCollateralBecauseLTV(uint256) (runs: 256, μ: 1041838, ~: 982083)
[PASS] testLTV(uint256) (runs: 256, μ: 1556290, ~: 1556372)
[PASS] testLoanInUntrackedPosition(uint256) (runs: 256, μ: 862866, ~: 862966)
[PASS] testMultipleFraxlendPositions() (gas: 3081871)
[PASS] testRemoveAllCollateralWithTypeUINT256Max(uint256) (runs: 256, μ: 739544, ~: 739653)
[PASS] testRemoveCollateral(uint256) (runs: 256, μ: 739462, ~: 739548)
[PASS] testRemoveCollateralWithTypeUINT256MaxAfterRepay(uint256) (runs: 256, μ: 1453406, ~: 1453510)
[PASS] testRemoveSomeCollateral(uint256) (runs: 256, μ: 769810, ~: 769909)
[PASS] testRepayPartialDebt(uint256) (runs: 256, μ: 1237118, ~: 1237220)
[PASS] testRepayingDebtThatIsNotOwed(uint256) (runs: 256, μ: 385690, ~: 385745)
[PASS] testRepayingLoans(uint256) (runs: 256, μ: 1288137, ~: 1288218)
[PASS] testTakingOutLoanInUntrackedPositionV2(uint256) (runs: 256, μ: 457599, ~: 457689)
[PASS] testTakingOutLoansV2(uint256) (runs: 256, μ: 1009053, ~: 1009133)
[PASS] testTotalAssets(uint256) (runs: 256, μ: 649349, ~: 649439)
Test result: ok. 16 passed; 0 failed; 0 skipped; finished in 11.27s

Running 15 tests for test/testAdaptors/FraxLendFToken.t.sol:FraxLendFTokenAdaptorTest
[PASS] testAddInterest() (gas: 417677)
[PASS] testDeposit(uint256) (runs: 256, μ: 312055, ~: 312135)
[PASS] testDepositV2(uint256) (runs: 256, μ: 481812, ~: 481884)
[PASS] testDifferencesWhenAccountingForInterestV1() (gas: 12386115)
[PASS] testDifferencesWhenAccountingForInterestV2() (gas: 11745043)
[PASS] testLendingFrax(uint256) (runs: 256, μ: 396585, ~: 396658)
[PASS] testMultiplePositionsTotalAssets(uint256) (runs: 256, μ: 976631, ~: 976781)
[PASS] testMultiplePositionsUserWithdraw(uint256) (runs: 256, μ: 1303165, ~: 1303235)
```

```
[PASS] testRebalancingBetweenPairs(uint256) (runs: 256, μ: 813781, ~: 813891)
[PASS] testTotalAssets(uint256) (runs: 256, μ: 342587, ~: 342675)
[PASS] testUsingPairNotSetupAsPosition(uint256) (runs: 256, μ: 415620, ~: 415730)
[PASS] testWithdraw(uint256) (runs: 256, μ: 420692, ~: 420762)
[PASS] testWithdrawV2(uint256) (runs: 256, μ: 610644, ~: 610756)
[PASS] testWithdrawableFrom() (gas: 1198215)
[PASS] testWithdrawingFrax(uint256) (runs: 256, μ: 427899, ~: 427983)
Test result: ok. 15 passed; 0 failed; 0 skipped; finished in 28.99s

Running 28 tests for test/testAdaptors/BalancerPoolAdaptor.t.sol:BalancerPoolAdaptorTest
[PASS] testAssetsUsed() (gas: 14598)
[PASS] testBalancerFlashLoanChecks() (gas: 32440)
[PASS] testBalancerFlashLoans() (gas: 723942)
[PASS] testClaimRewards() (gas: 1830583)
[PASS] testConstructorReverts() (gas: 80238)
[PASS] testDepositToHoldingPosition() (gas: 5887879)
[PASS] testExitBoostedPool(uint256) (runs: 256, μ: 1486046, ~: 1486439)
[PASS] testExitBoostedPoolProportional(uint256) (runs: 256, μ: 1781279, ~: 1781316)
[PASS] testExitPoolReverts() (gas: 2380807)
[PASS] testExitPoolSlippageCheck(uint256) (runs: 256, μ: 1313374, ~: 1313512)
[PASS] testExitVanillaPool(uint256) (runs: 256, μ: 1272099, ~: 1268407)
[PASS] testExitVanillaPoolProportional(uint256) (runs: 256, μ: 1392031, ~: 1392165)
[PASS] testFailTransferEthToFund() (gas: 18419)
[PASS] testIsDebt() (gas: 10626)
[PASS] testJoinBoostedPool(uint256) (runs: 256, μ: 1174945, ~: 1175028)
[PASS] testJoinBoostedPoolWithMultipleTokens(uint256) (runs: 256, μ: 1920979, ~: 1920996)
[PASS] testJoinPoolNoSwapsReverts() (gas: 1201772)
[PASS] testJoinPoolSlippageCheck(uint256) (runs: 256, μ: 1130044, ~: 1130152)
[PASS] testJoinPoolWithSwapsReverts() (gas: 1097566)
[PASS] testJoinVanillaPool(uint256) (runs: 256, μ: 1042629, ~: 1050784)
[PASS] testJoinVanillaPoolWithMultiTokens(uint256) (runs: 256, μ: 1517228, ~: 1517309)
[PASS] testNonStableCoinJoinMultiTokens(uint256) (runs: 256, μ: 1349028, ~: 1342721)
[PASS] testStakeBpt(uint256) (runs: 256, μ: 1491839, ~: 1491998)
[PASS] testStakeUint256Max(uint256) (runs: 256, μ: 1492865, ~: 1492989)
[PASS] testTotalAssets(uint256) (runs: 256, μ: 1974961, ~: 1975047)
[PASS] testUnstakeBpt(uint256) (runs: 256, μ: 1461848, ~: 1461998)
[PASS] testUnstakeUint256Max(uint256) (runs: 256, μ: 1462444, ~: 1462556)
[PASS] testUserWithdrawPullFromGauge(uint256,uint256) (runs: 256, μ: 2110104, ~: 2090764)
Test result: ok. 28 passed; 0 failed; 0 skipped; finished in 31.70s

Ran 39 test suites: 383 tests passed, 0 failed, 0 skipped (383 total tests)
```

# Code Coverage

| File | % Lines | % Statements | % Branches | % Funcs |
|---|---|---|---|---|
| **src/**Deployer.sol | 84.62% (**11/**13) | 85.00% (**17/**20) | 75.00% (**3/**4) | 75.00% (**3/**4) |
| **src/**Registry.sol | 97.00% (**97/**100) | 88.00% (**132/**150) | 76.56% (**49/**64) | 95.45% (**21/**22) |
| **src/base/**ERC4626.sol | 0.00% (**0/**4) | 0.00% (**0/**5) | 100.00% (**0/**0) | 0.00% (**0/**4) |
| **src/base/**Fund.sol | 91.40% (**255/**279) | 90.67% (**340/**375) | 86.54% (**90/**104) | 97.06% (**66/**68) |
| **src/base/permutations/**FundWithShareLockFlashLoansWhitelisting.sol | 100.00% (**24/**24) | 95.12% (**39/**41) | 93.75% (**15/**16) | 100.00% (**8/**8) |
| **src/base/permutations/**FundWithShareLockPeriod.sol | 100.00% (**26/**26) | 100.00% (**35/**35) | 85.71% (**12/**14) | 100.00% (**8/**8) |

| File | % Lines | % Statements | % Branches | % Funcs |
|---|---|---|---|---|
| **src/modules/adaptors/Aave/V3/**AaveV3ATokenManagerAdaptor.sol | 91.04% (**61/**67) | 86.84% (**99/**114) | 56.67% (**17/**30) | 92.31% (**12/**13) |
| **src/modules/adaptors/Aave/V3/**AaveV3AccountExtension.sol | 50.00% (**3/**6) | 50.00% (**3/**6) | 100.00% (**0/**0) | 50.00% (**3/**6) |
| **src/modules/adaptors/Aave/V3/**AaveV3AccountHelper.sol | 89.66% (**26/**29) | 93.02% (**40/**43) | 50.00% (**5/**10) | 100.00% (**8/**8) |
| **src/modules/adaptors/Aave/V3/**AaveV3DebtManagerAdaptor.sol | 81.48% (**22/**27) | 82.05% (**32/**39) | 33.33% (**2/**6) | 69.23% (**9/**13) |
| **src/modules/adaptors/**AggregatorBaseAdaptor.sol | 96.55% (**28/**29) | 96.08% (**49/**51) | 75.00% (**3/**4) | 85.71% (**6/**7) |
| **src/modules/adaptors/Balancer/**BalancerFlashLoanHelper.sol | 100.00% (**1/**1) | 100.00% (**1/**1) | 100.00% (**0/**0) | 100.00% (**1/**1) |
| **src/modules/adaptors/Balancer/**BalancerPoolAdaptor.sol | 94.53% (**121/**128) | 93.93% (**201/**214) | 88.64% (**39/**44) | 87.50% (**14/**16) |
| **src/modules/adaptors/**BaseAdaptor.sol | 69.57% (**16/**23) | 75.00% (**27/**36) | 50.00% (**6/**12) | 61.54% (**8/**13) |
| **src/modules/adaptors/**ERC20Adaptor.sol | 100.00% (**16/**16) | 100.00% (**25/**25) | 75.00% (**3/**4) | 85.71% (**6/**7) |
| **src/modules/adaptors/**ERC4626Adaptor.sol | 88.89% (**32/**36) | 90.38% (**47/**52) | 50.00% (**4/**8) | 90.00% (**9/**10) |
| **src/modules/adaptors/Paraswap/**ParaswapAdaptor.sol | 100.00% (**2/**2) | 100.00% (**3/**3) | 100.00% (**0/**0) | 100.00% (**2/**2) |
| **src/modules/adaptors/**PositionlessAdaptor.sol | 0.00% (**0/**6) | 0.00% (**0/**7) | 100.00% (**0/**0) | 0.00% (**0/**6) |
| **src/modules/adaptors/Swaap/**SwaapFundAdaptor.sol | 100.00% (**35/**35) | 96.15% (**50/**52) | 50.00% (**4/**8) | 100.00% (**10/**10) |
| **src/modules/adaptors/Swaap/**SwaapV2Adaptor.sol | 97.78% (**44/**45) | 98.57% (**69/**70) | 87.50% (**14/**16) | 100.00% (**7/**7) |
| **src/modules/fees/**FeesManager.sol | 95.60% (**87/**91) | 91.23% (**104/**114) | 81.25% (**26/**32) | 93.75% (**15/**16) |
| **src/modules/fees/**ManagementFeesLib.sol | 92.31% (**12/**13) | 95.00% (**19/**20) | 83.33% (**5/**6) | 100.00% (**2/**2) |
| **src/modules/fees/**PerformanceFeesLib.sol | 92.86% (**13/**14) | 94.74% (**18/**19) | 83.33% (**5/**6) | 100.00% (**2/**2) |
| **src/modules/**price-**router/Extensions/**ERC4626Extension.sol | 100.00% (**7/**7) | 100.00% (**14/**14) | 50.00% (**1/**2) | 100.00% (**2/**2) |
| **src/modules/**price-**router/Extensions/Swaap/**SwaapSafeguardPoolExtension.sol | 100.00% (**24/**24) | 95.35% (**41/**43) | 66.67% (**4/**6) | 100.00% (**3/**3) |

| File | % Lines | % Statements | % Branches | % Funcs |
|---|---|---|---|---|
| **src/modules/**price-**router/**PriceRouter.sol | 94.35% (**167/**177) | 92.73% (**255/**275) | 83.33% (**85/**102) | 92.86% (**26/**28) |
| **src/utils/**Math.sol | 0.00% (**0/**14) | 0.00% (**0/**19) | 0.00% (**0/**6) | 0.00% (**0/**6) |
| **src/utils/**SigUtils.sol | 0.00% (**0/**3) | 0.00% (**0/**4) | 100.00% (**0/**0) | 0.00% (**0/**2) |
| **src/utils/**Uint32Array.sol | 0.00% (**0/**12) | 0.00% (**0/**24) | 0.00% (**0/**6) | 0.00% (**0/**3) |

# Changelog

- 2024-04-02 - Initial report
- 2024-04-16 - 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

protocols, platforms, content, and materials, will be at your sole risk. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE. This report is based on the scope of materials and documentation provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and may not be represented as such. No third party is entitled to rely on the report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, or any related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third party. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.