Quantstamp

Primex Finance

Executive Summary

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

Type	Defi, trading platform
Timeline	2023-06-22 through 2023-08-17
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	PrimeX Guide Audit Docs (google drive)
Source Code	• primex-finance/primex_contracts #276d36f • primex_artifacts ##c840207 (initial deployment review commit)
Auditors	• Andy Lin Senior Auditing Engineer • Jennifer Wu Auditing Engineer • Hytham Farah Auditing Engineer • Adrian Koegl Auditing Engineer

Documentation quality	Medium
Test quality	Medium
Total Findings	59 Fixed: 36 Acknowledged: 15 Mitigated: 8
High severity findings	3 Fixed: 3
Medium severity findings	16 Fixed: 10 Acknowledged: 4 Mitigated: 2
Low severity findings	27 Fixed: 15 Acknowledged: 6 Mitigated: 6
Undetermined severity findings	1 Acknowledged: 1
Informational findings	12 Fixed: 8 Acknowledged: 4

Summary of Findings

PrimeX Finance aims to offer leveraged or spot trading on multiple decentralized exchanges (DEXs). Lenders have the option to provide liquidity to the "Buckets," allowing traders to borrow funds for leveraged trading. The position funds are held within the protocol's smart contract until they are closed and transferred.

The proper functioning of the PrimeX trading market relies, in large part, on the "Keeper" role. These are users who trigger transactions when certain protocol conditions are met, e.g. liquidations, limit orders, stop losses. For market operations to effectively function it is crucial to have a significant amount of computational power behind this role. Note that anybody can run their own Keeper and the team plans on open-sourcing an implementation of a Keeper, but that code is completely outside of the scope of this audit.

The project exhibits a significant level of complexity, with various modules and components for interaction. While the code is relatively easy to follow, the inherent complexity introduces several risks, and we have identified several issues during our audit. Additionally, due to the large contract codebase, we observed that validations for specific variants may be scattered across different locations, making it challenging to ensure certain protections are in place. Despite these challenges, the project demonstrates a well-architected structure, incorporating several existing safeguards, such as the oracle price divergence check. Overall, we had a positive experience working with the PrimeX Finance team. They were highly responsive and provided prompt answers to any questions or clarifications we sought during the process.

We strongly recommend addressing all issues. Each fix should ideally be accompanied by test cases verifying its resolution.

Fix Review Update: The PrimeX team has diligently addressed several fixes and changes, which we have reviewed. Their approach to resolving the issues demonstrates a high level of detail and precision. They have streamlined communication by providing us with comprehensive documentation and spreadsheets outlining all reported issues along with their corresponding pull requests. Additionally, they gave extensive, carefully reasoned responses when choosing to acknowledge and not address a particular issue. However, it is worth noting that some changes may be categorized as new features, even though they do, in fact, represent notable improvements. It is important to emphasize that this report exclusively reflects the status updates of the fixes to the issues originally included.

Deployment Review Update: Quantstamp conducted a deployment review for the PrimeX team, encompassing an examination of their deployment scripts, configurations, and on-chain values. Key areas of focus included verifying that the values used were reasonable and within expected parameters, confirming that authorization was correctly set to maintain security and access controls, and ensuring that the on-chain bytecode matched the codebase.

Throughout the review process, we encountered instances where re-deployment was necessary due to bug fixes and upgrades. Additionally, the PrimeX team expressed their intention to implement a 'testing period', which would involve post-deployment actions executed at a later stage. Consequently, our review primarily focused on the originally deployed version, with a brief examination of on-chain changes to ensure consistency and accuracy. We initiated the review with the `c37b34ee` commit of the `primex_contracts` repository and `c840207` of the `primex_artifacts` repository. After a few fixes and the re-deployment, we later reviewed some of the deployment changes in the `0ae0630` commit of the `primex_contracts` repository and the `e00f8c1` commit of the `primex_artifacts` repository. In the later review, we focused mainly on sanity-checking that the on-chain dependencies are linked correctly and also checked the values on the newly deployed Buckets. The correctness and intention of the values were not re-reviewed as those were done in the initial deployment review.

PRI-55 to PRI-59 represent our findings during the deployment review process, and the team has fixed or acknowledged all reported issues.

Repository Migration/Cloning: The PrimeX team duplicated the code from the `a3b0ce1` commit of the `primex_contracts` repository to a new repository named `primex-protocol` (accessible here) with the commit `a8a22bcd2a`. We have duly confirmed that the contracts remain identical between the `a3b0ce1` commit of `primex_contracts` and the `a8a22bcd2a` commit of `primex-protocol`.

Later, the PrimeX team also cloned the deployment artifacts from `primex_artifacts` to `primex-protocol` in `f9dee2f`. We checked the difference between the latest commit of `primex_artifacts` (`7b0dab8b` , see: link) and the `primex-protocol` repository (see: link), and noticed some minor differences in `PositionLibrary.json`. The difference is that some functions accept interfaces as input instead of addresses. The team clarified that this is to support the frontend and the keepers. The `primex_artifacts` are the artifacts generated during the deployment, and the team later updated the implementation address.

To our understanding, the repository changes were instigated by a legal recommendation to update the code's license, given that the original version of the codebase lacked a license declaration.

As an additional note, the file hashes provided in the Appendix pertain to the initial audit commit of the `primex_contracts` repository. It's worth noting that the final version may have undergone significant changes, as a majority of files required adjustments during the fix review process.

ID	DESCRIPTION	SEVERITY	STATUS
PRI-1	Traders May Lose Entire Position if Asset Was Removed From Bucket	High	Fixed
PRI-2	Volatility May Be Greater than One	High	Fixed
PRI-3	Slippage Vulnerability in PrimeX Protocol for Swap and Spot Trade Positions	High	Fixed
PRI-4	Insufficient Reward upon PMX Withdrawal	Medium	Fixed
PRI-5	Strict Oracle Tolerance in Spot Trade Can Prevent Traders From Closing Positions	Medium	Fixed
PRI-6	Close Conditions May Be Incorrectly Set in `BatchManager`	Medium	Fixed
PRI-7	Buggy `BestDexLens` Implementation	Medium	Fixed
PRI-8	Wash Trades to Steal Keeper and Spot Trading Rewards	Medium	Mitigated
PRI-9	Signature Replay Attack	Medium	Acknowledged
PRI-10	Accruing Debt in Deprecated Bucket	Medium	Mitigated
PRI-11	Assumption of Singular Bucket Updates in `_searchApproxIndex()` Function	Medium	Fixed
PRI-12	Risk of Using Outdated Token Price Feed Data	Medium	Acknowledged
PRI-13	Reserve May Fall Below the Minimum Percentage of Total Supply	Medium	Fixed
PRI-14	`ECDSA` Library Vulnerable to Signature Malleability	Medium	Fixed
PRI-15	Un-Executable Limit Orders	Medium	Acknowledged
PRI-16	Interest Accumulation Prior to Bucket Launch	Medium	Fixed

ID	DESCRIPTION	SEVERITY	STATUS
PRI-17	Deployers Can Upgrade Factory Deployed Contracts	Medium	Fixed
PRI-18	Overflow Blocking `ActivityRewardDistributor` From Functioning	Medium	Fixed
PRI-19	Front-Run Keeper Rewards	Medium	Acknowledged
PRI-20	PTokens Can Be Transfered to Blacklisted Address	Low	Fixed
PRI-21	Unlocked Pragma and Risk of Unsupported Opcode in the Latest Solidity Version	Low	Fixed
PRI-22	Risk of Inaccurate Reward Distribution Due to Error Handling	Low	Mitigated
PRI-23	Gas Manipulation for High Rewards	Low	Mitigated
PRI-24	Collect Less Protocol Fee with Volatile Asset and Updatable Protocol Fee	Low	Acknowledged
PRI-25	Silent Failure when Updating Bonus Leads to Unlogged Errors	Low	Acknowledged
PRI-26	Inconsistency in Pause Management when Opening Position	Low	Fixed
PRI-27	Inconsistency and Validation Gaps in Treasury Spending Limit Management	Low	Fixed
PRI-28	Loss of Points Due to Insufficient Precision	Low	Fixed
PRI-29	Liquidation Price Viewer Stops Functioning for Deactivated Buckets	Low	Acknowledged
PRI-30	Duplications and Inconsistencies in Referrers List Due to Absence of Validations in `setReferrers()` and `setReferrals()` Functions	Low	Fixed
PRI-31	Missing Input Validation	Low	Mitigated
PRI-32	Ownership Can Be Renounced	Low	Mitigated
PRI-33	Out of Gas Risk	Low	Mitigated
PRI-34	Missing Access Control in `PToken.setBucket()`	Low	Fixed
PRI-35	NFT Minted without Setting URI	Low	Acknowledged
PRI-36	`orderExist` Modifier May Incorrectly Return True	Low	Fixed
PRI-37	Privileged Roles and Ownership	Low	Mitigated
PRI-38	Spot Trade or Swap Order with the `depositedAsset` Being the Same as `positionAsset`	Low	Fixed
PRI-39	Unable to Clean Up Legacy Orders	Low	Fixed
PRI-40	User Coming Later Gets Less Reward from `ActivityRewardDistributor`	Low	Acknowledged
PRI-41	Ineffective Conditions Can Be Attached to Orders	Low	Fixed

ID	DESCRIPTION	SEVERITY	STATUS
PRI-42	Missing Authorization Check for `PTokensFactory.createPToken()`	Low	Fixed
PRI-43	Aave Integration Risk	Low	Acknowledged
PRI-44	`increaseDeposit()` May Revert if No Swap Is Performed	Informational	Fixed
PRI-45	Unclear Whether Native Currency Should Be Supported for Swapping	Informational	Fixed
PRI-46	Application Monitoring Can Be Improved by Emitting More Events	Informational	Fixed
PRI-47	Incompatibility with Non-Standard Tokens	Informational	Acknowledged
PRI-48	Upgradability	Informational	Acknowledged
PRI-49	Unresolved `TODO`	Informational	Fixed
PRI-50	`depositAndLock()` Not Directly Tied to a Deposit	Informational	Fixed
PRI-51	Infinite Token Approval	Informational	Fixed
PRI-52	Uninitialized Implementation Contract	Informational	Fixed
PRI-53	Using Deprecated Function	Informational	Fixed
PRI-54	Referral Program Susceptible to Sybil Attack	Undetermined	Acknowledged
PRI-55	`PositionManager` and `BatchManager` do not need `NO_FEE_ROLE`	Low	Fixed
PRI-56	Proposers Incorrectly Set to Executors	Low	Fixed
PRI-57	Missing `GUARDIAN_ADMIN` role	Low	Fixed
PRI-58	Missing DNS Address	Informational	Acknowledged
PRI-59	Different Solidity Compiler Used For Proxy and Implementation Contracts	Informational	Acknowledged

Assessment Breakdown

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

Disclaimer

Within the scope of this audit and fix review are only the features contained within the repositories at the specified commit hashes mentioned on the front page of the report. Additionally, we consider only the fixes for the issues outlined in this report. Any features or code changes introduced in future revisions are not covered in this assessment.

Following the initial audit, the team addressed findings from external auditors, alongside those from Quantstamp, implementing necessary fixes. We assisted in reviewing these changes, assuming they were all directed towards issue resolution, without access to their report. It's important to note that due to process limitations, there may be associated risks, and we do not assume full responsibility in such cases.

During the review of the deployment process, the team came across situations that necessitated redeployments or upgrades. However, due to time constraints, our attention was predominantly on the initial deployment. The subsequent deployment, as a result, underwent

a more restricted review, which may have inadvertently allowed for discrepancies to arise.

The client later migrated the repository to `primex-protocol`. It's crucial to reiterate that we specifically reviewed fixes from the initial audit commit in the `primex_contracts` repository to the final commit `a3b0ce1` they adopted. We maintained open communication and thoroughly engaged with them on all identified fixes and PRs. However, please be aware that there may be potential risks associated with changes we haven't assessed. We did not re-audit the entire codebase, focusing solely on fixes and PRs provided by the client.

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

Most contracts are in the scope with a few exceptions for testing.

Files Included

./src/contracts/ActivityRewardDistributor/ ./src/contracts/BatchManager.sol ./src/contracts/BonusExecutor/ ./src/contracts/Bucket/ ./src/contracts/Constants.sol ./src/contracts/DebtToken/ ./src/contracts/DexAdapter.sol ./src/contracts/EPMXToken.sol ./src/contracts/InterestRateStrategy.sol ./src/contracts/KeeperRewardDistributor/ ./src/contracts/LimitOrderManager/ ./src/contracts/LiquidityMiningRewardDistributor/ ./src/contracts/PMXBonusNFT/ ./src/contracts/PMXPriceFeed.sol ./src/contracts/PMXToken.sol ./src/contracts/PToken/ ./src/contracts/PositionManager/ ./src/contracts/PriceOracle/ ./src/contracts/PrimexDNS/ ./src/contracts/PrimexProxyAdmin.sol ./src/contracts/PrimexRegistry.sol ./src/contracts/PrimexTimelock.sol ./src/contracts/PrimexUpkeep.sol ./src/contracts/Redeemer.sol ./src/contracts/ReferralProgram/ ./src/contracts/Reserve/ ./src/contracts/SpotTradingRewardDistributor/ ./src/contracts/SwapManager.sol ./src/contracts/TraderBalanceVault/ ./src/contracts/Treasury/ ./src/contracts/WhiteBlackList/ ./src/contracts/conditionalManagers/ ./src/contracts/interfaces/ ./src/contracts/lens/ ./src/contracts/libraries/

Files Excluded

./src/contracts/TestnetServices/ ./src/contracts/UniswapInterfaceMulticall.sol ./src/contracts/mocks/

Findings

PRI-1

Traders May Lose Entire Position if Asset Was Removed From Bucket

High Fixed

Update

The health function, found in `PositionLibrary.health()` in commit `7d350f7`, returns the health value regardless of the asset's status. Additionally, the client updated the `BatchManager` in commit `4685290` to remove the ability to liquidate removed assets. The client resolved the issue by ensuring it is not possible to liquidate a healthy position if the asset has been removed from the approved list in the bucket.

File(s) affected: `Bucket.sol`, `PositionLibrary.sol`

Description: When an asset was removed from a bucket through `removeAsset()`, the assets of all open positions may be transferred to the treasury. The affected traders will lose their entire position.

The issue arises from the fact that `health()` in `PositionLibrary` returns 0 when `position.positionAsset` is not an allowed asset in the bucket. This can only occur if the asset was removed, as this position could not exist otherwise.

Because `health()` returns 0, those positions can be closed through `closePosition()` with `CloseReason.RISKY_POSITION`. Such closures will lead to the traders not receiving any profits (`topUpAvailableBalance()` will not be called) AND not receiving their deposit back, as `decreaseTraderDebt()` will be called with `params.primexDNS.treasury()` as a receiver.

Furthermore, when a position is closed due to liquidation, the trader will not receive any funds back. The deposit and potential profit will be sent to the treasury in the `PositionLibrary.closePosition()`.

Recommendation: Do not allow positions of removed assets to be closed with the `CloseReason.RISKY_POSITION`. This can be achieved by introducing another `CloseReason` of assets being removed and `health()` not returning 0 if the asset was removed.

Furthermore, the trader should be provided with timely notifications to safeguard from sudden position terminations.

PRI-2 Volatility May Be Greater than One

High Fixed

Update

The team has renamed `pairVolatility` to `pairPriceDrop`, defining it as "A drop of one token's price regarding another token that is possible during N seconds (for now, 600)." They have confirmed that they are not utilizing the Chainlink volatility feed. Initially, they will launch with hardcoded values and plan to collaborate with a decentralized oracle project for the specialized "price drop feed" in the future. Furthermore, the `PriceOracle.setPairPriceDrop()` function includes a require statement ensuring that `_pairPriceDrop` is less than `WadRayMath.WAD`, thus preventing it from exceeding one.

File(s) affected: `PriceOracle.sol`, `PositionLibrary.sol`, `PositionManager.sol`

Description: The Chainlink Volatility Feed measures the realized volatility of one asset against another. Note that realized volatility can be greater than or equal to 100% (see: formula here), and such an instance would create one of two problems:

- In the case where the volatility is exactly 100% any user holding this asset would have a health of 0 and hence be able to liquidate.
- In the case where the volatility is over 100% the calculation of a user's health will fail due to overflow and hence no one will be able to liquidate.

Recommendation: In the short term, several mitigations should be implemented. For instance:

1. Thoroughly vet the tokens for integration. Only those exhibiting stable volatility should be considered as bucket assets.
2. It is advisable to employ a longer-range volatility (e.g., 30 days) instead of a short-term volatility. This adjustment would reduce the risk of volatility surpassing unity.
3. Ensure the functionality of the `Position.health()` function even when volatility > 1. A temporary solution could involve capping the effective volatility at a designated threshold—let's call it the Volatility Cap (VC)—within the `health()` function. VC should ideally be set to a value below one (e.g., 0.5~0.9). This precaution will prevent the `health()` function from simply returning zero, which could trigger unwarranted liquidation in scenarios where a token experiences a sudden price surge, causing unnecessary disruptions.
4. Implement a monitoring system to track volatilities and facilitate prompt responsive measures. Potential actions may encompass pausing operations or executing immediate liquidations.

In the long term, it is recommended to reconsider the formula employed in the health function. For instance, the team might explore devising a function that remaps volatility back to the `[0, VC)` range and employ this recalibrated value in the calculations.

PRI-3 Slippage Vulnerability in PrimeX Protocol for Swap and Spot Trade Positions

High Fixed

Update

The team fixed the issue in the commit `ddf6d6a` with the allowing fine-grained control in the following places:

1. Added `maximumOracleTolerableLimit` and `needOracleTolerableLimitCheck` inputs to the `SwapManager.swap()` function .
2. Introduced `needOracleTolerableLimitCheck` to the `ClosePositionParams` and `OpenPositionVars` in `PositionLibrary`. The issue of the keeper attacking the limit order traders are solved in the following places:

3. Swap order: the `LimitOrderLibrary.openPositionByOrder()` calls `swapManager.swap()` with `pm.getOracleTolerableLimit()` and `needOracleTolerableLimitCheck` being `true`.
4. Spot trade order: `PositionLibrary.createPositionByOrder()` now sets the `needOracleTolerableLimitCheck` as `true` in the `OpenPositionVars`. This allows all order-created-positions to have the tolerance check.

In short, when opening a position by order regardless of spot or swap, there is always an oracle check. When closing (fully or partially) a position, there is always an oracle check if it is not closed by the trader or if there is borrowed amount. Note that if it is a spot trade position opened or closed by the user, it remains not having the oracle tolerance but they can specify their own `amountOutMin` when making such calls.

File(s) affected: `LimitOrderManager.sol`, `LimitOrderLibrary.sol`, `PrimexPricingLibrary.sol`, `PositionLibrary.sol`

Description: In the PrimeX protocol, the keeper can attack orders using spot trades or swaps. The protocol includes slippage protection by comparing the DEX execution price with the oracle price data within a certain gap. However, this protection is not applied to swap and spot trade positions. As a result, if a limit order is created for a swap or spot trade, there can be high slippage without the oracle price gap protection. The limit orders do not have the slippage protection by `amountOutMin` as well.

Exploit Scenario: For instance, the keeper can manipulate the DEX price through a flash loan and execute the order with a large slippage.

Recommendation: To fix this, please consider the following changes:

1. In `PositionLibrary.openPosition()` function, always call `PrimexPricingLibrary.multiSwap()` with `_needOracleTolerableLimitCheck` being true. It is set as `!data.isSpot` on L799 now.
2. In `LimitOrderLibrary.openPositionByOrder()` function, check the `exchangeRate` to be within the tolerable limit of the oracle (see: `PositionManager.getOracleTolerableLimit()` function).

PRI-4 Insufficient Reward upon PMX Withdrawal

Medium · Fixed

Update

The team fixed the issue as recommended in commit `0aeb163`.

File(s) affected: `ActivityRewardDistributor.sol`

Description: The `ActivityRewardDistributor.withdrawPmx()` function enables the admin to withdraw unused PMX tokens from the contract. During the execution of this function, the `bucketInfo.totalReward` is reduced by the withdrawal amount. However, the `bucketInfo.rewardPerDay` and `bucketInfo.endTimestamp` remain unchanged. As a result, there may not be enough PMX tokens available to cover the reward until the end timestamp.

Additionally, the `lastUpdatedRewardTimestamp` is not updated as expected. As a consequence, the time span since this timestamp was last refreshed is inaccurately accounted for during subsequent reward accumulation. Specifically, when rewards are withdrawn, the `bucketInfo.fixedReward` value increases based on the duration since the `lastUpdatedRewardTimestamp`. Failure to update the `lastUpdatedRewardTimestamp` can result in the period being double-counted if it is not properly updated.

Exploit Scenario: Here is a simplified scenario:

1. Given that the `ActivityRewardDistributor` contract only has Alice being the single user. The contract currently still has 10 PMX and the `rewardPerDay` is also 10.
2. The `BIG_TIMELOCK_ADMIN` calls `withdrawPmx()` and take away all the remaining 10 PMXs.
3. On the next day, when Alice wanted to claim, she expects to get the extra 10 while the contract would fail due to the lack of tokens to transfer.

Recommendation: In the `withdrawPmx()` function, please update the followings:

1. `bucketInfo.endTimestamp`: to reflect the updated `totalReward`.
2. `bucketInfo.lastUpdatedRewardTimestamp`: this should be updated when `bucketInfo.fixedReward` is modified within the function.

PRI-5 Strict Oracle Tolerance in Spot Trade Can Prevent Traders From Closing Positions

Medium · Fixed

Update

The team addressed the issue in commit `ddf6d6a`. The recently introduced variable `needOracleTolerableLimitCheck` assists in bypassing the oracle tolerance check on spot trades when they are closed by the trader.

File(s) affected: `PositionManager.sol`, `PositionLibrary.sol`

Description: When closing a spot trade position (or more specifically, on `vars.borrowedAmountIsNotZero` in `PositionManager._closePosition()#L682-690`) through `PositionManager._closePosition()` the `oracleTolerableLimit` is

set to zero between `depositAsset` and `positionAsset`. This means when closing the position through `PositionLibrary.closePosition()`, the swapped amount returned by the DEX in the `PrimexPricingLibrary.multiSwap()` must be equivalent to or better than the price provided by the price oracle. This check is validated in the `multiSwap()` where the `getOracleAmountsOut()` must be greater or equal to the amount obtained from the DEX because `_needOracleTolerableLimitCheck` is set to `true` in the `PositionLibrary.closePosition()` function.

This strict requirement can prevent spot traders from closing their positions.

Recommendation: Evaluate the necessity of implementing a zero `oracleTolerableLimit` for spot trades. This constraint might prevent traders from closing their positions, potentially impacting trading dynamics and user satisfaction.

PRI-6 Close Conditions May Be Incorrectly Set in `BatchManager`

Medium ⓘ Fixed

Update

The team fixed the issue as recommended in the commit `278fe7f`.

File(s) affected: `BatchManager.sol`

Description: When batch closing positions through `BatchManager.closeBatchPositions()`, each position listed in the array `_ids` is validated. If the position does not exist through `PositionManager.getPosition()`, the function will catch the error and continue to the next position id in the `_ids` array. Therefore the `i` variable may not track to `vars.length` and `vars.length` can be less than `i`.

```
if (
    _closeReason == PositionLibrary.CloseReason.BATCH_STOP_LOSS ||
    _closeReason == PositionLibrary.CloseReason.BATCH_TAKE_PROFIT
    )
        vars.closeConditions[vars.length] = positionManager.getCloseCondition(vars.ids[i],
_conditionIndexes[i]);
```

Therefore, the variable `i` used to obtain the position close condition can be incorrect. Since `i` can be ahead of `vars` the position id can be zero. This will return the close condition from position id `0` for the current position `_ids[i]` and can incorrectly close the current position being evaluated.

Recommendation: Correct the position id to the current position id being evaluated. The `vars.ids[i]` should be corrected to either `vars.ids[vars.length]` or `_ids[i]`. Furthermore, the tests should be revised to capture this scenario to ensure the correct close condition is used. There is only one test written to check the `continue` case.

PRI-7 Buggy `BestDexLens` Implementation

Medium ⓘ Fixed

Update

The team fixed the issue in the commit `436cbd5`. First, the `amountByDexByShare` is no longer immediately overridden. Second, the `gas` variable is renamed to `gasPriceInCheckedAsset`, meaning it will be in the same asset. Third, the casting is now using `SafeCast`. Fourth, the change introduces `VERY_POSITIVE_VALUE`. When `_params.isAmountToBuy` is true, the value will set to the `VERY_POSITIVE_VALUE`.

In the fix PR/commit, there is no test in place to validate the fix. It is highly recommended to incorporate tests to ensure accurate gas conversion and alignment with the optimization objective.

Description: The `BestDexLens.getBestMultipleDexes()` function filters out DEX results and attempts to optimize share splits across DEX results to return the best DEX routing path for the amount requested. However, several concerns arise from the current implementation:

1. **Overwriting the Negative Value**: Within the loop, a condition checks if `amount` equals 0 and assigns `VERY_NEGATIVE_VALUE` to `amountByDexByShare`. However, in the subsequent lines, this value is immediately overwritten without any related checks.
2. **Unsafe Gas Estimation**: The gas, represented in ETH units (`vars.gas`), is directly added or subtracted to/from the `amount`. Mixing gas estimations in ETH with an amount that might represent another token or asset unit could lead to inaccurate calculations or misrepresentations of value.
3. **Unsafe Casting to `int256`**: The `amount`, an unsigned integer, is directly cast to `int256`. Without ensuring the `amount` is within safe bounds, this could introduce potential overflows or unexpected negative values.
4. **Arbitrarily Large Negative Value Concern**: A value `VERY_NEGATIVE_VALUE` defined as `-1e72` is used to indicate an undesirable amount obtained from the DEX. The problem arises when `currentValue` is assigned this large negative value and subsequently compared to other values using the conditional `_params.isAmountToBuy ? (currentValue < bestValue || bestValue == 0) : currentValue > bestValue`. If `_params.isAmountToBuy` is `true`, then a `currentValue` of `VERY_NEGATIVE_VALUE` will almost always be considered smaller than `bestValue`, even if `bestValue` represents a more desirable outcome. This bug can lead to incomplete or suboptimal solutions.

Recommendation: The `BestDexLens.getBestMultipleDexes()` function filters out DEX results and attempts to optimize share splits across DEX results to return the best DEX routing path for the amount requested. This coupling of concerns makes the function complex and difficult to test.

We recommend making the following correction to the function's implementation:

1. Confirm if `VERY_NEGATIVE_VALUE` is needed; the current implementation overwrites after setting this value when the dex returns zero amount;
2. Re-evaluate the logic of mixing gas estimations with other token units. Consider separating these calculations or clearly documenting the reasoning behind this design;
3. Before casting to `int256`, ensure the value being cast is within the valid range or use `SafeCast`;
4. Review and handle `VERY_NEGATIVE_VALUE` with caution, ensuring that the use does not skew the optimization process unintentionally. Consider adding explicit checks for such values or utilizing other mechanisms to handle error states or undesirable outcomes.

In addition to addressing the bugs mentioned above, we recommend breaking down the `getBestMultipleDexes()` function into smaller, more specialized functions. This will help improve code readability, maintainability, and testability. We strongly recommend breaking down the function into individual functions and adding units to test each component.

PRI-8 Wash Trades to Steal Keeper and Spot Trading Rewards

Medium Mitigated

Update

The team made some mitigations to this issue:

1. commit `33f2c2b`: `CLOSE_BY_TRADER` reason is disallowed for `PositionManager.closePositionByCondition()`.
2. commit `88a0f56`: keeper reward will not be granted if the limit order or the position is updated in the same block/timestamp.

Aside from the above, the team also made the code change to enable different protocol fee rates on different types of trade in the commit `370bd57`. The change is less related to the issue but provides greater flexibility.

File(s) affected: `PositionManager.sol`, `PositionLibrary.sol`, `SpotTradingRewardDistributor.sol`

Description: Both keeper rewards and spot trading rewards are susceptible to being attacked through wash trading.

The function `PositionManager.closePositionByCondition()` allows anyone to close an existing position as long as the close reason condition is satisfied in `PositionLibrary.closePosition()`. Unlike the function `PositionManager.closePosition()` where the reason is restricted to `CLOSE_BY_TRADER`, the `CloseReason` can be decided by the keeper where the reason can be `CLOSE_BY_TRADER`, `RISKY_POSITION`, `BUCKET_DELISTED`, or `LIMIT_CONDITION`. Therefore, in addition to close conditions, this function can be also used to close positions for liquidations, bucket deprecation, and traders. Since the keeper rewards are paid by the protocol's treasury, this close reason flexibility in the function and the ability to open and close positions in the same transaction, the `closePositionByCondition()` function allows anyone to steal keeper rewards by repeatedly opening and closing positions to drain treasury rewards through `CLOSE_BY_TRADER` or `LIMIT_CONDITION`. The malicious trader can close the existing position using `CLOSE_BY_TRADER` or setting the close condition where the newly created position can be closed in the same block.

Similarly, the spot trading rewards can also be stolen through similar exploit scenarios by repeatedly opening and closing positions. When a user opens a position via `PositionManager.openPosition()`, their trading activity is recorded via `_updateTraderActivity()`. Crucially, the volume that they are treating in USD terms is added to their total trading activity in that period of time, which is a variable used to calculate the rewards they receive from the `SpotTradingRewardsDistributor` contract. However, when the position is closed by the user via `PositionManager.closePosition()`, the trading activity the user received before is not rescinded. Hence they can open and close positions multiple times in order to increase their trading activity without actually benefitting the protocol.

In this scenario, rewards are determined using the formula:

```
reward += (periodInfo.totalReward * periodInfo.traderActivity[trader]) / periodInfo.totalActivity;
```

The formula added intricacy when evaluating the economic feasibility of the attack, as rewards are influenced by the overall trading activity of other users as well.

This issue is mitigated by the fact that the protocol charges a protocol fee when opening the position. As long as the protocol fee is significantly larger than the sum of the rewards, the attack is mitigated because it would not be economically feasible. However, this also means that the deployment configuration value should be carefully vetted before launching the protocol, and some on-chain values need to be continuously monitored since some of the variants will not be under the protocol's control.

Exploit Scenario:

1. Trader opens a large position using a flash loan.
2. Trader closes the same large position by `closePositionByCondition()` with reason `CloseReason.CLOSE_BY_TRADER`.
3. The trader closes the position and gets the keeper reward with zero risk.
4. Continue until the treasury is drained.

Recommendation: Consider refactoring the function to limit the `CloseReason` flexibility. The function's naming implies the function can only be used for closing existing positions based on close conditions. Furthermore, the function should not allow `CLOSE_BY_TRADER` as a `CloseReason`. The trader is able to close the existing position using the `PositionManager.closePosition()`.

Also, please carefully set the deployment configuration and ensure continuous on-chain monitoring is in place:

1. The `KeeperRewardDistributor.positionSizeCoefficientA` should be significantly smaller than `PrimexDNS.protocolRate` and `PrimeDNS.protocolRateInPmx`. Additionally, `KeeperRewardDistributor.positionSizeCoefficientB` should not be excessively large.

2. The `PositionManager.minPositionSize` should be carefully set. The position size should be large enough to ensure the keeper's reward would not exceed the protocol fee. When the position size is low, the `positionSizeCoefficientB` multiples the gas cost might be higher than the protocol fee as there is no base fee.
3. The gas price should be monitored as `KeeperRewardDistributor` gives a reward based on the gas cost. When the gas price is very high and the gas cost ends up being higher than the position size, it can potentially give more reward than the charged protocol fee.
4. `SpotTradingRewardDistributor.rewardPerPeriod` should be carefully picked and continuously monitored. If the PMX token price is pumped or the `rewardPerPeriod` is too high, it would incentive attackers to wash trade.

PRI-9 Signature Replay Attack

Medium | Acknowledged

Update

The team acknowledged the issue with a detailed statement as follow:

> This issue touches several contracts, so comments are also separated: 1) NFT: We do not want to restrict the ability to mint NFTs for users who have received a signature from NFT_MINTER, so the absence of an expiration date is acceptable. We also have additional security measures in place that allow us to pause reward distribution and block incorrect NFTs according to protocol management rules.
>
> 2) Referral whitelist: We have decided to remove the requirement for whitelisting to become a referrer. Now, anyone can generate a signature that serves as a referral link and invite other users. The requirement for referrers to be whitelisted will be eliminated from the contracts.
>
> 3) Referral program: We have decided to independently launch the referral program on each network where Primex will be deployed. Additionally, we do not want to require the referrer and referee to be users of the same network. Therefore, referrers will generate universal signatures that can be used by referees on any EVM network. As a result, we do not need to add a chain Id. Referral links will be valid indefinitely, ensuring a positive experience for referrers and referees without any risks for misbehaving referees.
>
> The smart contracts for the referral program will solely serve the purpose of storing connections between users. The reward distribution for referrers will be calculated off-chain, taking into consideration referral connections across different networks. In cases of conflicts, the connection created earlier will be considered correct. The distribution algorithm will also be Sybil-resistant to protect the protocol from bots that generate fake referral connections. More information can be found in the answer to PRI-54.

File(s) affected: `WhiteBlackListReferral.sol`, `PMXBonusNFT.sol`, `ReferralProgram.sol`

Description: The following contracts are susceptible to cross-chain signature replay due to a lack of chain id and expiration mechanisms in their signed messages:

1. The PMX bonus NFT allows for the signing of minting parameters, dictating the type of bonus that can be activated by its owner. While the signature hashes the `MintParameters` which include the chain ID, it lacks an expiration date. This means the parameters remain valid indefinitely and new NFTs can be always minted with the signed parameters. Given that only those with the `NFT_MINTER` role can access the mint function, this issue serves primarily as an informational note.
2. The referral whitelist enables users to enroll as referrers in the referral program. The signed message contains the contract name, function, and referrer. However, if the whitelist contract is deployed across multiple chains using the same signer, the signature remains valid indefinitely on all chains. This is due to the omission of both chain ID and an expiration date in the signature.
3. The referral system permits signing referral messages without incorporating expiration dates or chain IDs. This design allows the message to be used across any EVM-compatible chain, and without a set expiration, the referral message remains valid indefinitely. It is crucial to inform referrers of these cross-chain signature replay risks when generating signed messages.

For the replay attack on the referral contracts to be successful, certain additional conditions must be met. For instance, the signer must be whitelisted or be the admin in multiple chains. Additionally, based on our discussion with the PrimeX team, even though the protocol aims to be deployed on multiple chains, they plan to run the referral program on only one chain. Therefore, the risk is relatively low in this scenario.

Nonetheless, we strongly urge addressing this issue despite the low risk, as the situation can change or the assumptions can be invalidated depending on the operation. It is crucial to proactively mitigate any potential vulnerabilities to ensure the long-term security and stability of the system. As a result, we escalated the severity to medium for this issue.

Recommendation: Consider following the EIP712 standard, which includes the chain ID and the contract address as part of the signature, to mitigate the risk of a replay attack. Alternatively, at the very least, incorporate the chain ID and the contract address into the signed messages if not adhering to the standard.

Additionally, we suggest incorporating an expiration timestamp into all signatures to limit the duration of signature validity.

PRI-10 Accruing Debt in Deprecated Bucket

Medium | Mitigated

Update

The team fixed the issue in the commit `711d76e`. An extra `position.bucket.isActive()` check occurs in the `PositionLibrary.decreaseDeposit()` function.

We have verified that currently, all places calling `Bucket.increaseDebt()` have the validation of the bucket status. However, as a best practice, it is better to consolidate the validation in the `Bucket` contract to ensure that future code changes will not miss the check.

File(s) affected: `PositionLibrary.sol`, `Bucket.sol`

Description: The current implementation of the `Bucket` contract permits users to interact with the contract even after it has been delisted. This raises significant concerns, especially regarding the `Bucket.increaseDebt()` function, as it places additional responsibility and burden on the delisted bucket. Consequently, it becomes possible to acquire more debt through existing positions by reducing deposits using the `PositionManager.decreaseDeposit()` function. In this scenario, anyone can close these positions by specifying the close reason as `BUCKET_DELISTED`.

It is important to note that the verification of the bucket status occurs in the `primexDNS.getBucketAddress()` function. This verification provides protection to the `Bucket.deposit()` function and the `PositionLibrary.createPosition()` functions implicitly, thereby limiting the attack factor to the `PositionManager.decreaseDeposit()` function.

Recommendation: Consider implementing a modification to the `Bucket` contract to stop the `increaseDebt()` functions when the bucket is delisted. Also, consider making the validation more explicit on the `deposit()` function for better readability. And/Or, before allowing the trader to increase debt from the bucket through the function `PositionManager.decreaseDeposit()`, the function should validate the status of the bucket.

Meanwhile, having the validation inside `primexDNS.getBucketAddress()` make it harder to read and realize this implicit intent. We recommend also having explicit validations.

PRI-11
Assumption of Singular Bucket Updates in `_searchApproxIndex()` Function

Medium | Fixed

Update
The team fixed the issue as recommended in the `bf8e378` commit.

File(s) affected: `FeeExecutor.sol`, `FeeDecreaser.sol`, `InterestIncreaser.sol`

Description: In the `FeeExecutor`, lists `updatedTimestamps` and `indexes` are used to track all index and timestamp updates for a bucket.

The function `FeeExecutor._searchApproxIndex()` is designed to return an approximate index corresponding to a given bonus deadline and the current index. However, its implementation, which employs a binary search approach, assumes that only one bucket updates are recorded in the `updatedTimestamps` and `indexes` lists. This assumption is incorrect because multiple buckets can update the lists using the function `FeeExecutor._updateIndex()`. Based on `FeeDecreaser` and `InterestIncreaser` functions, the `_updateIndex()` function pushes different bucket indexes which can be different based on the bucket's specific `bar` rate. This incorrect assumption can lead to inaccurate results returned by the `_searchApproxIndex()` function.

Recommendation: We recommend incorporating a mapping to track updates in `updatedTimestamps` and `indexes` on a per-bucket basis.

PRI-12 Risk of Using Outdated Token Price Feed Data

Medium | Acknowledged

Update
The team provided an analysis of the potential impact of an outdated token price feed and decided to keep it as it is. We agreed that in several cases, the result of not having an extra sanity check for the outdated oracle price feed is minimal or acceptable. A special case to point out is for liquidation. When the outdated oracle price is below the liquidation threshold, there are two scenarios to consider:

In case 1, if the real market price remains lower than the threshold, reverting to an outdated price will lead to a failed liquidation, posing a risk of an insolvent bucket. As a result, this should be the case where the sanity check for an outdated price feed should not be applied.

In case 2, if the price has recovered and liquidation is no longer necessary, using outdated data would result in unnecessary liquidation. Without a sanity check for an outdated price feed, it would put traders at risk during this scenario.

Since the team decided not to add the sanity check, users should be aware of the case 2 scenario above.

The following is the original statement from the team:

> There are several cases where oracles are used:
>
> 1. Liquidation: Liquidations cannot be blocked as it may result in significant losses for the protocol. If the oracle determines that the liquidation price has been achieved, corresponding positions must be closed quickly. Blocking liquidations after this point does not make much sense.
> 2. Stop loss: If the oracle price becomes lower than a specific stop loss (SL) price at any given moment, all positions with SL prices higher than the oracle price must be closed within a short timeframe. This ensures that a stuck oracle does

not result in the inability to close new positions. Positions that require processing with a new price missed by the oracle will not be processed. The same applies if we ignore the oracle with an outdated price.

3. Take profit: The executability of a take profit (TP) position is verified by the actual number of tokens received from decentralized exchanges (DEXs). After that, this price is not significantly lower than the oracle price. If the oracle gets stuck with a price higher than the TP, the position must have been closed when this price was set. If the oracle gets stuck with a price lower than the TP, Keepers will still be able to close the position once the real price on DEXs reaches the corresponding level.
4. Opening new margin positions or decreasing debts for existing ones: In this case, we validate the adequacy of the actual price after a swap on a DEX with the oracle price. If the oracle price is outdated, inadequate positions can be opened. We will manage this scenario with our emergency shutdown system, as described below.

We have an emergency pause service that allows us to pause borrowing funds from buckets, forbid opening new positions, etc. Different Chainlink (CL) feeds have different heartbeats, and the normal heartbeat value is not stored on-chain. Therefore, we will monitor the oracle behavior off-chain. If we detect an inadequate price update frequency, we will pause the corresponding components of the protocol until the oracle is stabilized. Additionally, we will provide information about outdated oracle prices to our users in the app. In future versions, we are considering incorporating reserve oracles for such critical situations. As for actions performed by traders with their own funds (spot and swap), we have removed oracle checks, so outdated data will not affect this type of trading.

Description: Chainlink updates the feeds periodically (heartbeat idle time). The application should check whether the timestamp of the latest answer is updated within the latest heartbeat or within the time limits acceptable for the application (see: Chainlink doc). The `getExchangeRate()` and `getOracleVolatility()` functions of the `PriceOracle` contract do not check the timestamp of the `answer` from the `IAggregatorV3(gasOracleAddress).latestRoundData()`.

Recommendation: In the `getExchangeRate()` function, ensure that the returned timestamp data from the `IAggregatorV3(gasOracleAddress).latestRoundData()` is updated recently enough.

For the `getOracleVolatility()` function, consider returning 0 or reverting the transaction if the data is outdated.

PRI-13
Reserve May Fall Below the Minimum Percentage of Total Supply

Medium · Fixed

Update
The team fixed the issue as recommended in the commit `0976587`.

File(s) affected: `Reserve.sol`

Description: The `Reserve` currently uses `pToken.scaledTotalSupply()` instead of actual token supply `pToken.totalSupply()`, when validating the minimum percentage of token supply in the `Reserve` during the function `transferToTreasury()`. This methodology could result in inaccuracies in the calculated minimum percentage, which can lead to a lower than expected minimum percentage in the reserve.

```
_require(
          reserveBalance >= (restristions.minAmountToBeLeft + amount) &&
              reserveBalance >=
              (pToken.scaledTotalSupply().wmul(restristions.minPercentOfTotalSupplyToBeLeft) + amount),
              Errors.NOT_SUFFICIENT_RESERVE_BALANCE.selector
          );
```

Recommendation: It is recommended to modify the calculation to use the actual token supply instead of the "scaled" token supply.

PRI-14 `ECDSA` Library Vulnerable to Signature Malleability

Medium · Fixed

Update
The team updated the Openzeppelin versions in the following commits: `ffed20b` and `1382573`.

File(s) affected: `PMXBonusNFT.sol`, `ReferralProgram.sol`, `WhiteBlackListReferral.sol`

Description: The protocol currently utilizes OpenZeppelin's `ECDSAUpgradeable` version `4.3.1` in contracts `PMXBonusNFT`, `ReferralProgram`, and `WhiteBlackListReferral` for validating registration using `ECDSAUpgradeable.recover(bytes, bytes)`. However, it is important to note that version `4.7.3` introduced a high-severity patch to address a vulnerability related to signature malleability in the `ECDSA.recover()` and `ECDSA.tryRecover()` functions. These functions were found to be vulnerable due to accepting EIP-2098 compact signatures in addition to the traditional 65-byte signature format. This vulnerability affected the functions that accept a single bytes argument, not the ones that take `r, v, s` or `r, vs` as separate arguments. As a result, the `PMXBonusNFT`, `ReferralProgram`, and `WhiteBlackListReferral` contracts are impacted since they utilize the single bytes argument version of `ECDSA`, and version `4.3.1` is affected by this vulnerability.

Recommendation: We strongly recommend upgrading the `ECDSAUpgradeable` library to at least version `4.7.3` or the latest stable version available to incorporate the security fixes.

PRI-15 Un-Executable Limit Orders

Medium | Acknowledged

Update

The team responded by stating that, as they support "third asset deposits," and considering a limit order can only lock the price of two tokens (the borrowing token and the position token), it is not possible to ascertain the price of the "third asset" at the time of order creation. Consequently, it becomes challenging to perform a comprehensive on-chain validation during order creation.

This issue can lead to the keeper wasting gas because it will fail during the condition check which occurs after going through `PositionManager.openPositionByOrder()` which will eventually fail at `PositionManager._openPosition()` where `maxPositionSize()` is also verified. The documentation should also warn the keeper to validate the position size before opening the limit order.

The following is the original statement:

> We allow traders to use a deposit in a third asset for margin trading. For example, a trader can use WBTC as their deposit, borrow USDC from a bucket, and buy ETH by selling both WBTC and USDC. For limit orders, we set the deposit size, leverage, and limit price. However, the price of the third asset can change independently, which means that the position size can also change during the order's lifespan and will only be fixed once the order is executed. In the frontend app, we prevent users from creating limit orders with an expected position size larger than the maximum allowed position size. Therefore, users can only create such orders by directly interacting with the contracts. We could implement this verification on-chain, but it would require additional gas and does not guarantee a 100% accuracy.

File(s) affected: `LimitPriceCOM.sol`, `PositionManager.sol`, `PositionLibrary.sol`

Description: It is possible for a user to open a limit order position that can never be closed. If it is ever the case that the user's `amountIn` (deposit * leverage) divided by their limit price is greater than the `maxPositionSize` between the `deposit` and `position` asset, then the limit order will never be closed due to the check of the `maxPositionSize` in the `LimitPriceCOM.canBeFilled()` function.

The severity of this issue is not as high since the user can still close the position themselves when they notice it. However, we strongly believe that this can be easily overlooked and can confuse users as to why the limit order is not executing. We suggest that the team fixes this issue for a better user experience and thus giving a medium severity.

The following is the detailed analysis:

Given the following scenario/assumptions:

1. Suppose a user deposits `depositAmount` of token A with leverage of `L`, and asks for a limit price `limitPrice` relative to token B.
2. Let `maxP = maxPositionSize[tokenA][tokenB]`.
3. The `amountIn` will be `L * depositAmount`, and suppose that `amountIn  / limitPrice > maxP`.

CLAIM: All calls to the `LimitPriceCOM.canBeFilled()` function will return `false` given the user's position.

To see this consider the following cases:

- Case 1: `amountOut > maxP`
  - The call will return `false` since the position is greater than the max size.
- Case 2: `amountOut <= maxP`
  - Note that `amountIn = price * amountOut`. Now suppose that the `price < limitPrice` (if this is not the case the call will return `false`). Then:
  - `amountIn = price * amountOut < limitPrice * amountOut` which implies:
  - `amountIn / limitPrice < amountOut <= maxP` which directly contradicts assumption 3.
  - Hence, it is impossible for the `price` to be lower than the limit price if `amountOut =< maxP`.

Exploit Scenario: Here is a sample scenario, assuming trying to open a position on the USDETH pair:

1. Given the current price of ETH is at a price of 2000, and the `maxPositionSize` for USDETH is 1 ETH. In other words, one can only open a position with a size not exceeding 1 ETH.
2. Alice put a limit order with the following setting: 2x leverage, with a deposit amount of 1000, and the limit order price being 1000. In other words, Alice wants to put a $2000 position on ETH when the price drops from 2000 to 1000.
3. When the price of ETH drops to 1000, and the keeper attempts to open the position for the limit order. However, since $2000 will become 2ETH now, the order will fail to open the position due to the `maxPositionSize` check.
4. The limit order is basically un-executable.

Recommendation: Consider doing a sanity check that the position can be opened on the limit price when creating the limit order.

PRI-16 Interest Accumulation Prior to Bucket Launch

Medium | Fixed

Update

The team fixed the issue as recommended in the commit `541096b`. Notably, the client also adjusted the bar to register as zero when "ur" holds a zero value, a modification unrelated to the current fix.

File(s) affected: `Bucket.sol`

Description: According to the documentation, it states that "During the stabilization period, Traders can now borrow funds from the Credit Bucket, and Lenders start earning interest on the liquidity they provide for margin trading." Based on this information, we assume that lenders only start earning interest after moving to the stabilization period, which occurs when the bucket launches. Additionally, the implementation of `_depositLM()` does not update indexes and rates. However, the `withdraw()` function allows lenders to withdraw funds before the bucket launches. Within this function, both `_updateIndexes()` and `_updateRates()` are called, causing the index to be updated and start accumulating before the bucket launches.

Exploit Scenario: Alice deposits 101 into a bucket and immediately withdraws 1 to trigger interest accumulation. The bucket starts owing Alice lending interest, despite no borrowing activities occurring to repay the accrued interest.

Recommendation: Add a condition check in the `withdraw()` function so that it only calls the `_updateIndexes()` and `_updateRates()` functions after the bucket has launched.

PRI-17 Deployers Can Upgrade Factory Deployed Contracts

Medium · Fixed

Update

The team added the capability to upgrade the beacon of the factories to the `PrimexProxyAdmin` contract. Additionally, in the deployment scripts, they promptly transferred the ownership of the factories to the `PrimexProxyAdmin` contract. These combined changes resolved the issue. Nevertheless, we recommend implementing the ownership transfer in the constructor for added confidence.

File(s) affected: `PTokensFactory.sol`, `BucketsFactory.sol`, `DebtTokenFactory.sol`

Description: The architecture graph suggests that exclusive roles should possess substantial privileges, which should be meticulously time-locked for enhanced security. The factories inherit the `UpgradeableBeacon` contract for beacon upgrades. However, the `UpgradeableBeacon` contract inherits the `Ownable` contract, making the deployer the owner and granting them the ability to upgrade implementations through the `UpgradeableBeacon.upgradeTo()` function (see: code). As a result, the deployer holds a significant privilege to upgrade all contracts, highlighting the need to secure this role and its associated responsibilities.

Recommendation: We would suggest doing the followings:

1. Consider renouncing the ownership directly in the `constructor()` of the factories.
2. Consider overriding the `UpgradeableBeacon.upgradeTo()` function in all factory contracts and apply the desired role restriction (e.g. timelock).

PRI-18 Overflow Blocking `ActivityRewardDistributor` From Functioning

Medium · Fixed

Update

The team applied the recommended change to the `ActivityRewardDistributor.getBucketAccumulatedReward()` function in the `60592e4` commit. The `_updateBucketInfo()` does not apply the fix because the callers of it (`updateUserActivity()` and `updateUsersActivities()`) both returns early when `bucketInfo.rewardPerDay == 0` before reaching it.

However, we recommend to add warning comments on the `_updateBucketInfo()` to avoid future code changes breaking the implicit assumption.

Meanwhile, instead of:

```
if (endTimestamp != type(uint256).max) {
    endTimestamp += unusedTime;
}
```

It would be preferable to write:

```
if (endTimestamp <= type(uint256).max - unusedTime) {
    endTimestamp += unusedTime;
} else {
    endTimestamp = type(uint256).max
}
```

File(s) affected: `ActivityRewardDistributor.sol`

Description: In both the `setupBucket()` and `decreaseRewardPerDay()` functions of the `ActivityRewardDistributor` contract, if `rewardPerDay` is equal to zero, the `bucketInfo.endTimestamp` is overridden to `type(uint256).max`. Additionally, the

`getBucketAccumulatedReward()` and `_updateBucketInfo()` functions have the logic of `endTimestamp += unusedTime` when `rewardPerDay == 0`. The line `endTimestamp += unusedTime` will overflow if the `endTimestamp` is set to `type(uint256).max`. Consequently, almost all functions relying on `getBucketAccumulatedReward()` and `_updateBucketInfo()` will fail.

From our discussion with the team, the `rewardPerDay` can be set to zero as emergency mitigation in case there is a bug to stop the accrual of incorrect rewards. Despite the likelihood of this occurring being low, it would cause extra chaos and worsen the emergency situation, which is why it is a medium-severity issue.

Exploit Scenario: After spotting a security issue, the `EMERGENCY_ROLE` wants to stop reward generation and thus calls the `decreaseRewardPerDay()` with `rewardPerDay` set as zero. Now, the functions on the `ActivityRewardDistributor` will fail.

Recommendation: One potential fix is to add a condition check before `endTimestamp += unusedTime` in the `getBucketAccumulatedReward()` and `_updateBucketInfo()` functions.

PRI-19 Front-Run Keeper Rewards

Medium | Acknowledged

Update

Since there is no easy solution for now, the team states that they might introduce keeper consensus and dynamic changing reward system in the future to mitigate this.

The following is the original statement from the team:

> We agree that it may be easier for a cheating "Keeper" to simply monitor pools and front-run transactions of other Keepers instead of hosting the Keeper software itself. It appears that there is no quick solution for this issue. However, in later versions, we plan to introduce a Keeper consensus algorithm where time will be divided into slots, with only a few Keepers being responsible for executing actions in each slot. The distribution of slots will depend on the stakes of the Keepers, and inactive Keepers will be penalized. This approach should reduce competition between Keepers and make front-running more difficult and costly.
>
> As a protocol, we prioritize the fast execution of the Keepers' tasks, so front-runners can potentially be beneficial since their transactions are executed more quickly. Additionally, we have set upper limits for rewards. Our Keepers also have the option to use private pools to conceal their transactions.
>
> Regarding the idea of decreasing Keeper rewards to only cover gas costs, or even less, it may not be the best solution in the early stages. This is because there may not be enough motivated users willing to contribute to the protocol, so the rewards should provide some level of profitability. In future versions, we are considering implementing a dynamically changing reward system based on the demand and supply of Keepers' services.

File(s) affected: `KeeperRewardDistributor.sol`, `BatchManager.sol`, `LimitOrderManager.sol`, `PositionManager.sol`

Description: The keeper reward mechanism exposes a risk to front-running. Currently, the `KeeperRewardDistributor.updateReward()` function is used to distribute keeper rewards. However, an attacker can exploit the system by monitoring the mempool and sending a transaction with a few input mutations, enabling them to fraudulently claim the reward without fulfilling the keeper effort.

The `KeeperRewardDistributor.updateReward()` function currently calculates rewards based on the gas cost multiplied by a coefficient and the position size. In other words, larger rewards are given for larger position sizes, as long as the reward value exceeds the transaction gas cost. This incentivizes attacking bots to engage in front-running.

The following is the list of impacted functions:

1. `BatchManager.closeBatchPositions()`
2. `LimitOrderManager.openPositionByOrder()`
3. `PositionManager.closePositionByCondition()`

Exploit Scenario: One can front-run the `PositionManager.closePositionByCondition()` function and get the reward simply by changing the `_keeper` input.

Recommendation: It does not appear to have a simple solution or perfect solution without trade-offs. There are some directions that can mitigate the issue:

1. The keeper should avoid sending transactions to the public mempool. For instance, they can use a flashbot to mitigate the front-run risk.
2. Ensure that the reward is slightly less than the gas cost or barely covers the gas cost. However, the downside is that this means that only people with some external motivation (e.g., lenders) will be willing to run the keeper as the protocol would not be rewarding enough. Additionally, this will require removing the part that gives larger rewards with larger position sizes.

PRI-20 PTokens Can Be Transfered to Blacklisted Address

Low | Fixed

Update

As per discussion with the client, this is an intentional design.

Blacklist contract is made in such a way that we can block only contracts. In the case of PToken and the `transfer` and `transferFrom` functions, this was not done due to the fact that it is an erc20 token that can be used in other protocols. In dex or something. Thus, we want to avoid distrust, because if we put a contract of the same dex in the blacklist, we will block the tokens forever

Update

The team implemented the blacklist check in the commit `510179d`. The reason for this addition, despite having been previously acknowledged, is explained below:

> We discussed it once more and decided to include this restriction as a precaution. This is in case there is an attack vector that could potentially allow an attacker to transfer a significant amount of pTokens to their address, such as from the Reserve. If we find that this measure is excessive, we will consider removing it through an upgrade.

A modifier `isRecipientNotBlacklisted()` has been applied to the `transfer()` and `transferFrom()` functions, but not to the `mint()` function in the `PToken`. The functions triggering the `Ptoken.mint()` function on the `Bucket` contract, namely `depositFromBucket()` and `deposit()`, include the `Bucket._notBlackListed()` check. However, it is recommended to implement this modifier using a `_beforeTokenTransfer()` hook rather than on each individual token transfer.

We would also like to highlight that the new changes do not prevent a holder, who was previously not on the blacklist, from transferring funds even if they later become blacklisted. This is because only the recipient is validated.

File(s) affected: `PToken.sol`

Description: In the `transfer()` function of `PToken`, there is no check to ensure that the recipient is not blacklisted. In accordance with these checks in other places, blacklisted addresses should not be able to receive any pTokens.

Note: We discussed this with the team during the audit, and the team has clarified that this is the intended behavior.

Recommendation: Consider adding the following check to the `transfer()` function:

```
_require(
        !IWhiteBlackList(bucket.whiteBlackList()).isBlackListed(recipient), ...
    );
```

PRI-21
Unlocked Pragma and Risk of Unsupported Opcode in the Latest Solidity Version

Low · Fixed

Update

The team locked the version to `0.8.18` in the commit `2a2dcc1` with the exceptions the interface files, which are suitable to leave them unlocked.

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".

Note that Solidity `0.8.20` might only work on Ethereum Mainnet at this moment as it introduced a new OP Code that other EVM-compatible chains might not support yet (see: this article for more information).

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

PRI-22
Risk of Inaccurate Reward Distribution Due to Error Handling

Low · Mitigated

Update

In commit `2933047`, the `ActivityRewardDistributor` contract now records the "old balance" of each user that the contract last saw. It utilizes the difference between the "new balance" and the "old balance" to update the reward per token and the fixed rewards for the users. This provides the system with the ability to automatically recalculate the reward based on the contract's last synchronization. This does not necessarily guarantee the "correct" reward, but it does help mitigate risk and ensures that the `ActivityRewardDistributor` will not run out of rewards to distribute.

Following the code change, we have some best practice suggestions:

1. In `ActivityRewardDistributor.updateUsersActivities()`, consider adding a comment for the line calling `_updateBucketInfo()` to explain that the condition `role == Role.TRADER` is in place to only allow `DebtToken.batchBurn()` to update the reward per token, without altering it for the `PToken.transfer()` and `transferFrom()` functions.
2. One concern regarding the new `_updateBucketInfo()` parameters is that they take two separate arguments, `users` and `length`, and then update the balance of `length` amount of users. There is a missing validation to ensure `length <= users.length`, and it is not entirely clear why a separate length parameter is necessary when `users.length` could suffice.
3. The nat spec should be updated for `BucketInfo` to account for the new variable.

File(s) affected: `PToken.sol`, `DebtToken.sol`

Description: The reward distribution mechanism in `PToken` and `DebtToken` is designed to update rewards based on changes in the token supply. The reward per token is recalculated to reflect any changes in the token supply, ensuring that token holders continue to accumulate rewards at a rate proportionate to their holdings. However, when an error occurs during the update of reward information, the try/catch block causes this step to be skipped. This means that when a change in token supply occurs, the reward per token could potentially remain unchanged. As a result, token holders may continue to accumulate rewards at the previous rate, which may not be accurate given the new token supply.

The implementation incorporates a try-catch pattern to isolate failures in the reward distribution contract from affecting the pToken and debt token contracts. However, there is uncertainty regarding the operational process to address the gap in missed rewards when an unexpected failure occurs. It is essential to establish a clear and defined procedure to rectify any missed rewards promptly.

Recommendation: A well-defined and thoroughly tested procedure for manual reward addition should also be established to handle exceptions. This may involve thorough monitoring, timely identification of issues, and implementing appropriate measures to reconcile and distribute any missed rewards.

PRI-23 Gas Manipulation for High Rewards

Low Mitigated

Update

The recent design change on the commit `bda7234` seems less related to the main issue, as the original concern focuses more on the gas price and the fix here is for the gas amount. However, it's important to note that these changes serve a valid purpose in preventing keepers from potentially exploiting the reward system within the `BatchManager`. Specifically, this concerns scenarios where they might spend a lot of gas trying to close positions that are inherently non-closable, yet still receive substantial rewards in return.

The code changes primarily involve managing how much gas is used based on conditions that make certain positions ineligible for closure when using the `BatchManager`. These gas-related adjustments are explained in the `decreasingGasByReason`. It's recommended to update the user documentation to include a warning about the possibility of a keeper doing work without getting paid if they use `BatchManager` to close positions, and some of the positions are actually non-closable. For lower risk, keepers may prefer using the single close function, which reduces the chances of encountering front-running and potential loss of rewards.

Description: After EIP1559 the gas cost per transaction is split into a base fee and a priority fee. The priority fee is awarded to the validator of that block, whereas the base fee is burned. The priority fee can be set as high as the user wants, usually with the intention of increasing their likelihood of getting included in a block.

In the scenario where a someone is both a keeper and a validator of a specific block, they have the opportunity to perform an attack which involves paying themselves an usually high priority fee in order to increase the rewards they receive as part of the gas refund.

Since the rewards are given directly in terms of the value of the native token, such an attack could seriously harm the protocol's treasury.

Note that, while such an attack is currently not possible on some L2s (such as Arbitrum) given their centralized sequencer model, it may be in the future if they transition to a more distributed architecture.

The issue is mitigated by the fact that `KeeperRewardDistributor.updateReward()` function has a cap on the `maxGasPriceForReward` to be related to the gas price from the oracle. Additionally, the protocol charges a protocol fee when opening the position. As long as the protocol fee is significantly larger than the reward, the attack is mitigated because it would not be economically feasible. As a result, we lowered the severity for this issue.

Exploit Scenario: In the following scenario, we are imagining that the keeper is also a validator and hence may arbitrarily order transactions according to their preference.

1. Using one address the attacker opens a position that can be immediately closed.
2. The attacker calls `PositionManager.closePositionByCondition()` and in this transaction awards himself a very high priority gas
3. This function then calls `KeeperRewardDistributor.updateReward()` which calculates the rewards using the following formula, where `gasPrice = tx.gasprice`:

```
uint256 reward = (gasAmount * gasPrice).wmul(uint256(positionSizeMultiplier));
```

4. The attacker profits from disproportionately high rewards.

Recommendation: Please carefully set the deployment configuration and ensure continuous on-chain monitoring is in place, for more detail, please check the recommendation on the issue: "Wash Trades to Steal Keeper and Spot Trading Rewards ".

PRI-24
Collect Less Protocol Fee with Volatile Asset and Updatable Protocol Fee

Low | Acknowledged

Update

The team confirmed that they would like to keep it as is for better UX to not require the users keeping specific tokens on their balance. They will also monitor and analyze the fee accumulated in the future. The following is the original statement from the team:

> We have decided to fix the protocol fee at the moment of a limit order creation. This decision was made to avoid the need for traders to constantly maintain a balance of the chain native token and ePMX on their protocol account in order to ensure the proper execution of their orders. The specific fee amounts are difficult to predict due to fluctuations in prices. Additionally, we understand that the fee must be recalculated after an order is edited, as it directly depends on the order size. We acknowledge that there is a possibility for traders to create limit orders under favorable market conditions and pay a lower fee, while others may create their orders during unfavorable market conditions and pay more. On average, we anticipate that the protocol fee will align with the rate we have set. We will actively monitor and analyze the accumulated fees in the protocol to optimize them in the future.

File(s) affected: `LimitOrderManager.sol`, `LimitOrderLibrary.sol`

Description: The `LimitOrderManager.createLimitOrder()` function allows traders to create limit orders and locks in the protocol fee charged to the trader based on the deposit size converted to native currency or the protocol token PMX at the time of order creation. Therefore, if the value of the deposit asset changes significantly before the order is executed, the effective protocol fee (in terms of the deposit asset) could differ from the originally locked-in fee. This could lead to a situation where the protocol either receives a lower-than-expected fee or where traders pay higher fees when the order is executed.

Furthermore, the trader can update the protocol fee as long as the order is not executed through the function `LimitOrderManager.updateOrder()`. This mechanism could be exploited by traders to lock in a more favorable protocol fee, particularly in times of high deposit asset volatility.

Recommendation: We recommend evaluating the potential consequences associated with the timing of protocol fee determination and its updatability to ensure fairness and economic stability for both the protocol and traders. Changes might include calculating the protocol fee at the time of order execution rather than at creation, and/or introducing restrictions on the ability of traders to update the protocol fee once locked in.

PRI-25
Silent Failure when Updating Bonus Leads to Unlogged Errors

Low | Acknowledged

Update

The team pointed out that our description was incorrect regarding the discrepancy between the `FeeDecreaser` and `InterestIncreaser` contracts. They intend not to throw an error to save some gas, as they are more interested in other kinds of errors. We agree with the mistake in the description and consider it acceptable not to throw an error here.

In the commit `9e51105`, the team made a change to trigger the global index update even if the user does not have such an NFT. This could help collect more data to yield better results for the `FeeExecutor._searchApproxIndex()` function. The change, although in the same area, is not a direct fix for the described issue. Therefore, we update the status to acknowledged here.

Note that we agree with the PR reviewer that it would be a best practice to add the validation `bucket == _bucket` in this comment, but we have not identified any potential exploits arising from this omission.

File(s) affected: `FeeDecreaser.sol`, `InterestIncreaser.sol`

Description: The `updateBonus()` function in both `FeeDecreaser` and `InterestIncreaser` is responsible for updating user bonuses. However, it will silently fail, without generating an error, when either the contract is paused or the bonus has not been activated yet in the `FeeDecreaser`. On the other hand, the `InterestIncreaser` contract will revert when the contract is paused or the bonus has not been activated yet.

This function is crucial to the operation of the `updateBonuses()` function, which is called by either `PToken` or `DebtToken` to revise user bonuses when changes occur in user balances. This silent failure on the `FeeDecreaser` contract can impede the proper functioning of `DebtToken`, which rely on the `updateBonuses()` function and a try-catch statement to log any potential errors. If no error is thrown when the bonus is paused or not activated, `DebtToken` will not be able to log the error event. This could lead to unexpected behavior and a loss of traceability for potential issues.

Recommendation: To ensure proper operation and error logging, it is recommended to modify the `updateBonus()` function in the `FeeDecreaser` contract. This modification involves throwing an error similar to the `InterestIncreaser` contract when the contract is paused or when the bonus is not yet activated. By making this adjustment, `DebtToken` will be able to accurately capture and log any error events that occur during the execution of the function.

PRI-26 Inconsistency in Pause Management when Opening Position

Low Fixed

Update

The team fixed the issue in the commit `a8c5d1d` by reverting the `LimitOrderManager.openPositionByOrder()` if the `PositionManager` contract is paused.

However, during our review of other code changes, which are outside the scope of this report, we noticed that in commit `3a038a7`, the pause check was moved to `PositionManager.openPositionByOrder()`. While the main purpose of commit `3a038a7` was to revert transactions with unexpected `msg.value` instead of returning it, we believe the inclusion of the pause mechanism may be unrelated to the commit's purpose.

File(s) affected: `PositionManager.sol`, `LimitOrderManager.sol`

Description: If `PositionManager` contract is paused, it is still possible to open new positions via the `LimitOrderManager` contract through the `LimitOrderManager.openPositionByOrder()` function.

However, this can be mitigated once the `LimitOrderManager` contract is paused as well.

Recommendation: It is important to verify whether this behavior is intentional. If not, it is recommended to improve the alignment of pause management across these contracts. Specifically, the `LimitOrderManager` contract should be adjusted to prevent the initiation of new positions when the `PositionManager` contract is paused. This will ensure that operational states are consistent across the system.

PRI-27 Inconsistency and Validation Gaps in Treasury Spending Limit Management

Low Fixed

Update

In the commits `c80e9ed` and `42f4e7a`, we reviewed that most recommended validations are applied. The `isSpenderExist` validation is applied in other commits but already shown in those two commits. The only validation missing is `minTimeBetweenTransfers` and we assume that the team would like to have the ability to have it be zero.

The `setMaxSpendingLimit` has been changed from a medium to a large time-lock along with the mentioned commit(s), although this change is not directly related to the fix.

File(s) affected: `Treasury.sol`

Description: The current structure of the `Treasury.setMaxSpendingLimit()` and `Treasury.decreaseLimits()` functions, crucial for managing the treasury spending, exhibit inconsistencies and lack the necessary validation checks.

The `setMaxSpendingLimit()` function allows the `Treasury` to add a new spender address with spending limits or update existing limits. This function does not validate the `_newSpendingLimits` used and but the function is locked behind a `MEDIUM_TIMELOCK_ADMIN`. Although the function name implies modifying the maximum spending limits for a spender, the implementation allows any data in `SpendingInfo.limits` to be updated for new and existing spenders.

The `Treasury.decreaseLimits()` function allows `Treasury` to update the spending limits of existing spenders. This function validates that the new limits are decreased and time durations are increased. However, the function does not validate the existence of the spender address. This lack of validation could allow for unintentional increases in the time duration for non-existent spenders in the `spenders` mapping.

The following input validations are recommended, but not limited to:

1. When adding or updating spending limits for a spender address,
    1. `maxTotalAmount` should exceed zero,
    2. `maxAmountPerTransfer` should exceed zero,
    3. `maxPercentPerTransfer` should be less than `WAD` or reasonably bounded,
    4. `minTimeBetweenTransfers` should exceed zero or reasonably bounded,
    5. `maxAmountDuringTimeFrame` should exceed zero or reasonably bounded.
2. When decreasing spending limits for a spender address,
    1. `isSpenderExist` should be validated,

Recommendation: We recommend introducing more input validations when adding a new spender address or updating existing spending limits. Furthermore, depending on the treasury's use case, consider adding a function to deactivate existing spenders and track the total spendings of spenders.

PRI-28 Loss of Points Due to Insufficient Precision

Low Fixed

Update

The team fixed the issue in the commit `b8ea360` and uses `WAD` to increase the precision.

File(s) affected: `LiquidityMiningRewardDistributor.sol`

Description: The `getLenderInfo()` function calculates the percentage of funds a lender holds with the formula:

```
(lenderInfo.points * 100) / buckets[_bucketName].totalPoints;
```

This calculation can lead to a loss of precision. For example, a lender with 1 point in a 1000-point bucket would be represented as holding 0% rather than the correct 0.1%. As the total points accumulate in `totalPoints`, the inaccuracy magnifies.

Recommendation: The `getLenderInfo()` function should be modified to employ a more precise calculation mechanism regardless of the lender's point allocation. To preserve precision, we recommend utilizing the `WadRayMath` to maintain precision during division.

PRI-29
Liquidation Price Viewer Stops Functioning for Deactivated Buckets

Low | Acknowledged

Update
The team clarified that the function we pointed out (`PrimexLens.getLiquidationPrice()` with 5 inputs) is only intended to be used *before* opening a position. Once the position is opened, another function `getLiquidationPrice(address _positionManager, uint256 _id)` should be used instead. Thus, if the bucket is inactive, it is expected for the function to revert.

File(s) affected: `PrimexLens.sol`

Description: The `PrimexLens.getLiquidationPrice()` function calculates the liquidation price for a given position amount and borrowed amount. The function looks up the bucket address based on the provided `_bucket` name using the function `primexDNS.getBucketAddress()`. However, this function reverts when the bucket is not active. Therefore, the external `getLiquidationPrice()` viewer becomes unusable when the bucket is not active.

Recommendation: Consider ensuring that the liquidation price viewer function is usable regardless of the bucket's status. One potential solution is to utilize the `PrimexDNS.buckets()` function, generated from the public storage mapping, instead of relying solely on the `PrimexDNS.getBucketAddress()` function.

Alternatively, another potential solution is to eliminate the validation within the `PrimexDNS.getBucketAddress()` function. It's important to note that these validations serve a purpose in various scenarios. Therefore, we advise exercising caution when contemplating the removal of these validations, unless all affected functions are updated simultaneously as well. The first option might be easier and less risky as a fix.

PRI-30
Duplications and Inconsistencies in Referrers List Due to Absence of Validations in `setReferrers()` and `setReferrals()` Functions

Low | Fixed

Update
In commit `da6b319` , `setReferrers()` function is removed and `setReferrals()` adds a condition check of `if (referrerOf[referralProgramUnits[i].referrals[j]] == address(0))` as the existence check.

The team made other changes regarding the referral mechanism, although they are unrelated to the issue. In the commits `072d560` and `43faee2` , the referral no longer requires whitelisting, and the related functions/roles have been removed.

File(s) affected: `ReferralProgram.sol`

Description: The `setReferrers()` function permits the configuration of multiple referrers. However, it currently does not verify the existence of a referrer in the `referrers` list prior to the addition of a new referrer. This can result in the same referrer address being duplicated in the `referrers` list.

Furthermore, the `setReferrals()` function neglects to validate the existence of referrals or referrers before updating the `referralsOf` and `referrerOf` mappings. It also fails to add new referrers to the `referrers` list. This can lead to inconsistencies in the tracking of referrers and referrals.

Recommendation: Consider implementing existence checks in both `setReferrers()` and `setReferrals()` functions before any modifications to mappings or lists.

PRI-31 Missing Input Validation

Low | Mitigated

Update

> We checked the status of the validations under the finalized commit `4fa6401bd5`, despite most changes being done in commit `4063436`.
>
> We have directly updated the status of each item pointed out in the description.

File(s) affected: `All contracts`

Description: It is crucial to validate inputs, even if the inputs come from trusted addresses, to avoid human error. A lack of robust input validation can only increase the likelihood and impact in the event of mistakes.

Following is the list of places that can potentially benefit from stricter input validation:

1. Fixed `Bucket.initialize()#L17`: the interface of `liquidityMiningRewardDistributor` should be validated.
2. Acknowledged `Bucket.setMaxTotalDeposit()#L183`: the `_maxTotalDeposit` can be set lower than the current token supply. **Update:** The team clarified that this is a feature.
3. Acknowledged `Bucket.setReserveRate()#L149`: add a validation on `_fee`. **Update:** the team clarified that the fee can be zero.
4. Fixed `Bucket.setFeeBuffer()#156`: add a validation on `_feeBuffer`. **Update:** Added `_feeBuffer < WadRayMath.WAD + (WadRayMath.WAD / 100)` in the `Bucket` contract.
5. Fixed `BucketsFactory.constructor()#L31`: the interface of `_bucketImplementation` should be validated. **Update:** Also, all the overwrote `upgradeTo()` functions the factories have the same check.
6. Fixed `FeeExecutor.setTierBonus()#L44`: the `bonus[i].percent` should not be zero. The percent is verified in `_activateBonus` to confirm active status.
7. Acknowledged `BonusExecutor.setMaxBonusCount()#L58`: the `_maxCount` should be set lower than the existing bonus count. **Update:** The team clarified that this is a feature.
8. Fixed `PositionLibrary.increaseDeposit()#L211`: the `amountToTrader` should be greater than zero before updating the trader's balance in the trader balance vault. **Update:** Instead of adding a validation, a condition check is done before `traderBalanceVault.topUpAvailableBalance()`.
9. Fixed `PositionLibrary.decreaseDeposit()#L288`: the `position.amount` should be greater than zero before updating the trader's balance in the trader balance vault.
10. Fixed `PositionLibrary.closePosition()#L327`: the `vars.amountToReturn` should be greater than zero before updating the trader's balance in the trader balance vault. **Update:** Instead of adding a validation, a condition check is done before `traderBalanceVault.topUpAvailableBalance()`.
11. Acknowledged `PositionLibrary.createPosition()#L555`: the price feed oracle should exist before creating the leveraged position. **Update:** The existence of the oracle is checked when adding a new asset to the bucket.
12. Acknowledged `PositionLibrary.createPositionByOrder()#L631`:
    - the close condition parameters should be validated. **Update:** the team clarified that it is intended to not check the price of closing conditions. Other checks take place in the setCloseConditions in PositionLibrary.
    - check that `_params.order.depositedAsset != _params.order.positionAsset`. **Update:** the team clarified that the deposit asset of a margin position may be equal to the position asset.
    - make sure the token pair exists on DEX: `priceOracle.getPriceFeedsPair(_params.order.positionAsset, _params.order.depositedAsset)`. **Update:** the team clarified that adding new restrictions is unnecessary because the existence of the oracle between deposit and position assets will be checked on later stage in `PositionLibrary.openPosition()`.
    - if `!isSpot`, check that `position.bucket.allowedAssets(_params.order.positionAsset)`. **Update:** the team clarified that the allowedAssets is checked when creating an order.
13. Acknowledged `PositionLibrary.openPosition()#L699`: the close condition parameters should be validated. **Update:** It is intended to not check the price of closing conditions. Other checks take place in the setCloseConditions in PositionLibrary.
14. Fixed `PrimexPricingLibrary.getAmountIn()#L168`: the first token of `_params.routes.length` should be `_params.tokenA`. **Update:** Also, the team added the check that the last token is the tokenB.
15. Fixed `PToken.unlockDeposit()#L88`: the `deposits` length should be greater than zero; otherwise, the function can unexpectedly revert.
16. Fixed `PToken.depositProlongation()#L108`: the `deposits` length should be greater than zero; otherwise, it is possible to extend a non-existent locked deposit. **Update:** the `depositProlongation()` function no longer exists.
17. Fixed `PositionManager._onlyExists()#L767`: the `positions` length should be greater than zero; otherwise, it is possible to return true for a non-existent position when the positions list is empty.
18. Fixed `LimitOrderManager.orderExists()#L457`: the `orders[index]` length should be greater than zero; otherwise, it is possible to return true for a non-existent order when the `orders[index]` list is empty.
19. Fixed `LimitOrderLibrary.createLimitOrder()#L357`: the deadline should be validated when creating limit orders.
20. Acknowledged `LimitOrderLibrary.createLimitOrder()#L357`: the price feed oracle should exist before creating the leveraged limit order. **Update:** If the position asset is available for the bucket, the oracle is checked on `addAsset()`.
21. Fixed `Reserve.setTransferRestrictions()#L105`: values should be validated when setting reserve transfer restrictions. **Update:** Add the validation for the `minPercentOfTotalSupplyToBeLeft` field.
22. Acknowledged `Treasury.transferFromTreasury()#L36`: the existence of the `spender.isSpenderExist` should be validated before transferring funds. **Update:** the team states that it is unnecessary because if the spender doesn't exist, all limits will be 0 and the first validation will fail.
23. Fixed `Treasury.decreaseLimit()#L110`: the existence of the `spender.isSpenderExist` should be validated before decreasing the spending limit.
24. Fixed `KeeperRewardDistributor.setPmxPartInReward()#L144`: the `_pmxPartInReward` should be validated before setting the PMX's portion of the reward.
25. Fixed `KeeperRewardDistributor.setNativePartInReward()#L151`: the `_nativePartInReward` should be validated before setting the native currency's portion of the reward.
26. Fixed `DexAdapter._swapWithBalancer()#L236`: the `path` length should be two when performing a single swap. **Update:** A check `_require(path.length >= 2, ...)` is added. The function will return on the condition `if (path.length > 2) {}`, so the single swap can only be `path.length == 2`.

27. Fixed `PMXBonusNFT.blockNFT()#L44` : the activation status of the NFT should be validated before deactivating the bonus. **Update:** used a condition check instead of validation.
28. Fixed `PMXBonusNFT._mint()#L63` : the `_nftParameters.uris` length should be validated before updating the uris for the token id.
29. Fixed `ActivityRewardDistributor.updateUserActivity()#L53` : should check that the `bucket` indeed exists. For instance, add a check that the `bucketAddress` get from `dns.buckets()` are not zero address.
30. Fixed `Reserve.transferToTreasury()#L81` : should check that the `bucket` has already been launched. Otherwise, the `Bucket(bucket).withdraw()` call will fail as it will try to call `liquidityMiningRewardDistributor.removePoints()` , and the `Reserve` does not have any liquidity mining points.
31. Fixed `LimitOrderLibrary.createLimitOrder()#L48` : consider checking that `order.bucket.isActive()` in the `else` block of the `if (vars.isSpot) {}` condition after setting the `order.bucket` . The `LimitOrderLibrary.updateOrder()` checks this and the two should be consistent.
32. Fixed `TraderBalanceVault.batchTopUpAvailableBalance()#L130` : consider validating that `params.assets[i] != address(0)` and `params.traders[i] != address(0)` . Note that both fields are validated in a similar function `topUpAvailableBalance()` .
33. Acknowledged `SpotTradingRewardDistributor.setRewardPerPeriod()#L170` : consider adding a validation on `_rewardPerPeriod` . **Update:** The team stated that it may be almost any value.
34. Acknowledged `Redeemer.changeRate()#L38` : validate that the `_rate` argument is bellow a certain threshold. **Update:** The team added a validation that `rate > 0` , while the description asked to add an upper boundary. However, the team has decided not to include the `Redeemer` in the initial launch. The logic will be refined and re-audited in the upcoming releases.
35. Fixed `SwapManager.setSwapRate()#L61` : validate that `_swapRate` is below a certain threshold. **Update:** in the new logic, it's checked in `PrimexDNS._setFeeRate()` .
36. Fixed `SwapManager.setSwapRateInPmx()#L68` : validate that `_swapRateInPmx` is below a certain threshold. **Update:** in the new logic, it's checked in `PrimexDNS._setFeeRate()` .

Recommendation: Add the validations and checks listed in the description.

PRI-32 Ownership Can Be Renounced

Low Mitigated

Update

Instead of renouncing ownership, it is transferred to the `PrimexProxyAdmin` contract, which only the `BIG_TIMELOCK_ADMIN` can perform ownership-related features. Please note that the transfer of ownership occurs in the deployment script.

File(s) affected: `BucketFactory.sol` , `PTokenFactory.sol` , `DebtTokenFactory.sol`

Description: If the owner renounces their ownership, all ownable contracts will be left without an owner. As a result, any function guarded by the `onlyOwner` modifier will no longer be able to be executed, such as upgrading the `UpgradeableBeacon` implementation contract. Furthermore, critical role transfers are not following a two-step pattern. The two-step pattern ensures that there is an additional layer of verification and authorization before transferring critical roles, reducing the risk of unauthorized access or malicious actions.

Recommendation: Confirm that this is the intended behavior. If not, override and disable the `renounceOwnership()` function in the affected contracts. For extra security, consider using a two-step process when transferring the ownership of the contract (e.g. `Ownable2Step` from OpenZeppelin).

PRI-33 Out of Gas Risk

Low Mitigated

Update

Since it is by design, the gas risk cannot really be fixed. The team added code comment in the commit `9cead78` to ensure it would not be misused on chain.

File(s) affected: `BestDexLens.sol` , `PrimexUpkeep.sol`

Description: Numerous functions within the `BestDexLens` contract involve iterating through multiple dexes and conducting intricate calculations to determine the optimal route. This concern becomes particularly pronounced when an increasing count of dexes is integrated into the protocol, raising apprehensions about the computational load leading to potential out-of-gas errors.

The central problematic function is `getBestMultipleDexes()` , which iteratively traverses a dex list. This function, in turn, is invoked by `getBestDexForOpenablePosition()` , `getBestDexByPosition()` , and `getBestDexByOrder()` .

We consider the severity as low because, in the scope of this audit, the functions are only used in the `PrimexUpkeep.checkUpkeep()` function. From our understanding, the `checkUpkeep()` is only intended to be for a static call to simulate the order execution and retrieve a filtered result that can be performed. However, if any of the functions are used outside our knowledge unexpectedly, the severity can be higher.

Recommendation: If the functions are only intended for frontend usage, please add warning comments/docs to the functions to ensure they should not be accidentally used in an unexpected way. Also, consider moving the logic frontend or off-chain SDKs instead.

PRI-34 Missing Access Control in `PToken.setBucket()`

Low Fixed

Update

The team fixed the issue as recommended in the commit `24962f1`. The same issue for debt token is also fixed.

Description: Any user can call the `PToken.setBucket()` function and change the bucket corresponding to the `PToken` as long as they chose an address conforming to a valid PrimeX bucket interface.

However, once this bucket is set it cannot be changed. Thus, this attack has a very small window of opportunity but can be disruptive if successfully executed. Additionally, this issue is mitigated in the `BucketsFactory.createBucket()` function. Essentially, the pToken is deployed/created, and `setBucket()` is immediately called, ensuring that this scenario would not occur in practice due to their deployment mechanism.

Nonetheless, if the `PToken` contract is ever used independently, this issue should be a concern.

Recommendation: Add a modifier to ensure that only privileged users can call this function.

PRI-35 NFT Minted without Setting URI

Low Acknowledged

Update

The team stated that the NFT feature will **not** be part of the initial release:

> NFT will not be included in the first release, for now, we have necessary calls in other contracts that will work with them, but these calls will be turned off. NFT logic, including URI definition, will be improved, re-audited and introduced in the next releases.

File(s) affected: `PMXBonusNFT.sol`

Description: The `PMXBonusNFT._mint()` function mints an NFT with a default set of metadata. Notably, the `uri` field is left with an empty string `""`. This is bizarre given that there is even a base URI function which returns `primexURL/`, which also seems like a default value that has not been set.

Recommendation: The team should clarify whether this is intended and clearly state the intention for the URI field. If it is not intended, they should consider adding at least the base URI to the `NftMetadata` struct when minting a new NFT. Ideally, they should include data unique to the NFT, such as its ID, in conjunction with the base URI.

PRI-36 `orderExist` Modifier May Incorrectly Return True

Low Fixed

Update

The team fixed the issue as recommended in the commit `4063436` (as part of Missing Input Validation).

File(s) affected: `LimitOrderManager.sol`

Description: The `limitOrderManager.orderExists` modifier may run into a problem when `orders.length == 0`. In this case, if it is called with input `0` then it will return true since the check `orders[index].id == _orderId` essentially amounts to a check that `0 == 0`.

Recommendation: Perform a check that `orders.length != 0` before checking if the specific orderId exists.

PRI-37 Privileged Roles and Ownership

Low Mitigated

Update

The team provided detailed responses on the privileged roles. Aside from that, the team added a mechanism to have an extra sanity check that certain roles can only be a contract in the commit `db0f92b`. While the contract validation is valuable, it has limitations. Attackers can potentially bypass these checks by routing their calls through a contract if they can set specific roles. Nevertheless, having this check in place helps decrease the potential attack surface.

Since most actions are behind their well-considered time-lock contracts, we would like to set the status as mitigated. We provide a detailed status update on each of the highlighted areas in the description section. Please review it there as well.

The following is their detailed response (note that the client uses the phrase re-audit, but we are only conducting a fix-review here):

> Firstly, we'd like to answer your findings and then provide the general context of our role system that will also be included (after some polishing) in our documentation.
>
> 1.1 The role VAULT_ACCESS_ROLE will be assigned to several contracts in the protocol: ActivityRewardDistributor, BatchManager, LimitOrderManager, LiquidityMiningRewardDistributor, PositionManager, SpotTradingRewardDistributor,

SwapManager. This role is managed by BIG_TIMELOCK_ADMIN, we will add additional requirements that this and several other roles can be only assigned to smart contracts to decrease the risk of setting incorrect addresses.

2.1 BIG_TIMELOCK_ADMIN can't withdraw all PMX, only the undistributed ones: the variable `availablePMX` decreases every period, we will rename it to `undistributedPMX` to increase the code clarity.

2.2 BIG_TIMELOCK_ADMIN can cancel margin limit orders only after delisting the corresponding bucket. We will rename this method from `cancelLimitOrdersByAdmin` to `cancelDelistedLimitOrdersByAdmin`. Also, the Trader's deposit from such an order will be returned to the Trader, this change will be included in the commit for reaudit.

UPD 05.09. This method will be removed, the possibility to cancel limit orders from delisted buckets will be added to the method `cancelExpiredOrder` added for the issue PRI-39.

2.3 Redeemer contract has not been finished yet and will not be included in the first release, for now, we will remove these methods and automate ePMX burning after redeem to make it more transparent.

2.4 The upgradability system is described in the answer PRI-48

3.1 The method Reserve.transferToTreasury doesn't allow to transfer all funds from any bucket, the maximum amount is restricted by the number of bucket pTokens owned by the Reserve. Here, the Reserve serves as a regular lender withdrawing the underlying assets with no additional rights. pTokens are accumulated in the Reserve every time when the liquidity index of the corresponding bucket increases, in this way, the Reserve receives a part of the borrowing fee paid by traders.

4.1 Redeemer contract has not been finished yet and will not be included in the first release.

4.2 It's correct behaviour, EMERGENCY_ADMIN can't affect already distributed rewards but can decrease future rewards in the case of suspicious activity.

We will also perform two post-deployment parameter audits to make sure that all roles are granted correctly.

Here we'd like to provide more context about how roles are assigned. We use AccessControl registry for several purposes:

1) Contract management. Here we have 5 types of admins:

- BIG_TIMELOCK_ADMIN - have the biggest rights in the protocol, can assign other roles excluding SMALL and EMERGENCY admins managed by MEDIUM and SMALL admin correspondingly; can make critical changes such as contract upgrades, setting the most important parameters, etc. This role will be assigned to a timelock contract with a 10-day delay. When Primex becomes a DAO, the management of this role and 2 next timelock roles will be transferred to DAO. Initially, will be managed by multisig.
- MEDIUM_TIMELOCK_ADMIN - can manage SMALL admins, and change different parameters in the protocol. Has a 2-day delay; initially, will be managed by multisig.
- SMALL_TIMELOCK_ADMIN - can manage EMERGENCY admins, unpause contracts paused by EMERGENCY, and make some other actions. Has a 12-hour delay; initially, will be managed by multisig.
- EMERGENCY_ADMIN - can pause different contracts, mostly blocking depositing and borrowing, but it can't block withdrawal or repaying debts. This role will be granted to external monitoring services. We can't trust EMERGENCY_ADMIN, so their actions are strictly restricted and maximum damage in the case of EMERGENCY compromisation is the restriction of some protocol features until unpausing.
- GUARDIAN_ADMIN - can cancel admin proposals in all 3 timelocks. Can pause timelock contracts in the case of the admin compromisation. Can't change anything in the protocol. Will be managed by multisig (different from the multisig used in BIG/MEDIUM/SMALL admins).

2) Cross-contract interactions

- VAULT_ACCESS_ROLE - can put funds to TraderBalanceVault and take them from there, will be granted to the following contracts:
  - ActivityRewardDistributor
  - BatchManager
  - LimitOrderManager
  - LiquidityMiningRewardDistributor
  - PositionManager
  - SpotTradingRewardDistributor
  - SwapManager
- NO_FEE_ROLE - can use SwapManager.swap without paying fee, will be granted to Buckets and PositionManager
- LOM_ROLE - granted to LimitOrderManager
- PM_ROLE - granted to PositonManager
- BATCH_MANAGER_ROLE - granted to BatchManager

3) Other privileged roles

- TRUSTED_TOLERABLE_LIMIT_ROLE - a trusted keeper that will be able to liquidate positions with higher OracleTolerableLimit, will be used only in critical case when market liquidity drops significantly and a risky position can't be closed with a normal OracleTolerableLimit.
- NFT_MINTER - can provide signatures to mint NFTs; NFTs will not be included into the first release.

File(s) affected: `all contracts`

Description: The protocol incorporates several privileged roles, and it is important to highlight the associated centralization risk to users interacting with the protocol. The following is a list of the privileged roles and their corresponding privileges:

1. VAULT_ACCESS_ROLE
    1. Acknowledged May call `TraderBalanceVault.withdrawFrom()` to withdraw any amount of tokens from any trader.
2. BIG_TIMELOCK_ADMIN
    1. Acknowledged May call `SpotTradingRewardDistributor.withdrawPmx()` to withdraw all the PMX. **Update**: To clarify, only the `undistributedPMX` (previously `availablePMX`) can be withdrawn.
    2. Fixed May cancel limit orders by calling `cancelLimitOrdersByAdmin()`. **Update:** The `LimitOrderManager._unlockAssetsAndDeleteOrder()` function has been modified to return the funds to the trader of the order, rather than to the `msg.sender`, which would be the admin if called through the `cancelLimitOrdersByAdmin()` function.
    3. Fixed May claim an arbitrary amount of `earlyPMX` and `PMX` tokens from the `Redeemer` contract by calling `adminClaimEarlyTokens()` and `adminClaimRegularTokens()` respectively. **Update:** Those functions have been removed.
    4. Acknowledged May change the implementation of any upgradeable contract calling `PrimexProxyAdmin.upgrade()` or `upgradeAndCall()`. **Update:** Please refer to PRI-48 for a detailed response regarding the upgradeability.
3. MEDIUM_TIMELOCK_ADMIN
    1. Fixed May transfer tokens as long from any bucket as long as they leave `transferRestrictions[].minAmountToBeLeft` tokens in the bucket, by calling `Reserve.transferToTreasury()`. However, the same admin can also change the `transferRestrictions[]` struct by calling `Reserve.setTransferRestriction()`. Hence, they have the ability to transfer the entire funds of any bucket to the protocol treasury. **Update:** The maximum amount is determined by the number of pTokens owned by the Reserve. This design enables the Reserve to receive a portion of the borrowing fee paid by traders. We have marked the status as "fixed" as this is not inherently an issue.
4. EMERGENCY_ADMIN
    1. Acknowledged May prevent all users from redeeming early `PMX` tokens by calling `Redeemer.pause()`. Note that the `SMALL_TIMELOCK_ADMIN` could then unpause. **Update:** The client states that the Redeemer will not be included in the first release.
    2. Acknowledged May decrease the amount of rewards by calling `SpotTradingRewardDistributor.decreaseRewardPerPeriod()`. **Update:** The team clarifies that this is the intended behavior, enabling the emergency role to respond to suspicious activity. This trade-off represents the team's decision between decentralization and operational security.

Recommendation: We understand that the team has carefully designed the privileged roles and attempted to balance the operational risk and centralization risk. However, it is crucial to make the centralization of power clear to the users.

PRI-38 Spot Trade or Swap Order with the `depositedAsset` Being the Same as `positionAsset`

• Low Fixed

> **Update**
> The team added the suggested validation in the commit `7c255f7`.

File(s) affected: `LimitOrderManager.sol`, `LimitOrderLibrary.sol`

Description: An order for a swap or spot trade should not have the `depositedAsset` being the same as the `positionAsset`, as there is no need for them to be identical. The `LimitOrderLibrary.createLimitOrder()` function includes validation on the assets when `vars.isSpot` to prevent this scenario. However, it is important to note that when using the `LimitOrderManager.updateOrder()` function, one can change the `depositedAsset` without the same validation. As a result, it is possible to create a spot trade or swap order with the `depositedAsset` being the same as the `positionAsset`, which may lead to unexpected behavior or unnecessary operations.

Recommendation: With the current codebase, the direct solution would be to add validation against the assets in the `LimitOrderManager.updateOrder()` function when the order is for a spot trade or a swap. Additionally, we recommend following the `FREI-PI` pattern (see: blog) which suggests adding an invariants validation at the end of the call. In this case, the invariants of the order can be validated at the end to ensure consistency and prevent any unexpected behavior.

PRI-39 Unable to Clean Up Legacy Orders

• Low Fixed

> **Update**
> The team added a `cancelExpiredLimitOrders()` function in the commit `c5444db`. It is important to note that this function also allows the cancellation of orders if the bucket is under `isWithdrawAfterDelistingAvailable()` status.
>
> Given that the cancellation of orders under the `isWithdrawAfterDelistingAvailable()` condition is no longer an admin action, the team should consider renaming the `adminDeadline` variable in the `Bucket` contract.

File(s) affected: `LimitOrderManager.sol`

Description: The `LimitOrderManager` contract currently only allows the trader to cancel an order with the `cancelLimitOrder()` function, unless the bucket has been delisted. However, if the `deadline` of the order has passed, it may be kept in the contract indefinitely unless the trader actively cancels it. These legacy orders will be stored in the contract, including arrays such as `traderOrderIds` and `bucketOrderIds`. When there are too many legacy orders, it can potentially block certain features, such as the `getBucketOrders()`

function, from executing due to running out of gas. Additionally, directly accessing the public array may also raise gas concerns, although this can be configurable through the node (refer to the gas limit on `eth_call` discussion here).

Recommendation: Consider providing a function for others to clean up orders that have passed the deadline.

PRI-40
User Coming Later Gets Less Reward from `ActivityRewardDistributor`

Low | Acknowledged

> **Update**
> We discussed this with the team during the audit, and the team has acknowledged and decided to keep it as it is. The team states that the liquidity index increases rather slowly, and the liquidity mining program will not be too lengthy.

File(s) affected: `ActivityRewardDistributor.sol`

Description: The `ActivityRewardDistributor` contract calculates the reward based on "scaled" balances. Using "scaled" balances means that users who come later will receive fewer PMX rewards. For example, if a user deposits the same amount of the borrowing asset into the bucket and stays for the same amount of time, they will earn less reward compared to users who came earlier. This is because the "scaled" balance for the same amount decreases over time as "index" grows.

Recommendation: Check if this is intended, if not, consider redesigning the rewards to be based on the non-scaled amount.

PRI-41 Ineffective Conditions Can Be Attached to Orders

Low | Fixed

> **Update**
> The team removed the ability to update `shouldOpenPosition` in the commit `83a3614` .

File(s) affected: `LimitOrderManager.sol` , `LimitOrderLibrary.sol`

Description: An order for a swap does not require, and should not have, a close condition attached to it. This is because a swap order does not open a position, so there is no need for a way to close it. The `LimitOrderLibrary.setCloseConditions()` function includes a check on `_order.shouldOpenPosition` to prevent this situation. However, it is important to note that this check can be bypassed if a user changes a spot trade order to a swap order using the `LimitOrder.updateOrder()` function. In such cases, the originally attached closing condition will remain attached, potentially leading to unexpected behavior.

Recommendation: With the current codebase, the direct solution would be to either clean up the closing conditions or reject the change when `shouldOpenPosition` is modified from true to false in the `LimitOrder.updateOrder()` function. Additionally, we recommend following the `FREI-PI` pattern (see: blog) which suggests adding an invariants validation at the end of the call. In this case, the invariants of the order can be validated at the end to ensure consistency and prevent any unexpected behavior.

PRI-42 Missing Authorization Check for `PTokensFactory.createPToken()`

Low | Fixed

> **Update**
> Both the `DebtTokenFactory.createDebtToken()` and `PTokenFactory.createPToken()` functions now feature an `onlyBucketsFactory()` modifier. This modifier ensures that the designated caller is the address stored in the `bucketsFactory` state variable, which can be set by the `BIG_TIMELOCK_ADMIN` .
>
> However, validation is missing: checking that `_bucketsFactory` is not set to `address(0)` .

File(s) affected: `PTokensFactory.sol`

Description: The `PTokensFactory.createPToken()` function seems only needed for the `BucketFactory.createBucket()` function. However, there is no authorization check for the `PTokensFactory.createPToken()` function. In other words, anyone can call the function and deploy a new pToken. We do not find any exploit due to this, but if any of the off-chain components are to listen to the `PTokenCreated` event, they might be wrongly notified and have unexpected behavior.

Recommendation: Consider adding an authorization check so that only the `BucketFactory` contract can call the function.

PRI-43 Aave Integration Risk

Low | Acknowledged

> **Update**
> The team acknowledged the issue with the following statement:

We will continuously monitor the AAVE protocol to detect any suspicious activity. Regarding the issue of not having enough funds to withdraw liquidity from AAVE, we anticipate that the initial amount of liquidity we collect will be significantly smaller than what is typically available in AAVE. As a result, the level of risk involved is minimal.

As for the monitoring of the actual difference in tokens expected and returned from AAVE, we have decided to maintain the existing logic. However, a proper solution would involve accounting for the liquidity withdrawn from AAVE, as well as the remaining liquidity, and implementing a mechanism that allows the administrator to withdraw liquidity from AAVE in smaller portions. This would help mitigate the risk of having low liquidity in AAVE. We plan to incorporate such a mechanism in future versions.

It is important to note that if AAVE is compromised, the funds that have been deposited will still be at risk. However, we are willing to accept this risk. Additionally, when it comes to swaps on different decentralized exchanges (DEXs), we consider actual balance changes since we are integrating various DEXs, and some of them may not be as secure as AAVE.

Description: The protocol integrates with Aave, with generally low risk, and is only used prior to bucket launch. However, there are a few potential risks to highlight:

1. Aave may not have enough liquidity to withdraw if the token is lent out.
2. There is a non-zero risk of a blackswarm event causing Aave to become insolvent.
3. The `Bucket._withdrawAllLiquidityFromAave()` function does not check the real returned balance or balance difference between the `IPool(aavePool).withdraw()` call.

Recommendation: For the first two risks, we suggest continuously monitoring the Aave conditions to be able to take immediate action if anything unexpected occurs.

For the third risk, it is safer to use the difference of the token balance in between the call `IPool(aavePool).withdraw()` rather than relying on `IAToken(aToken).balanceOf(address(this))`.

PRI-44 `increaseDeposit()` May Revert if No Swap Is Performed

Informational · Fixed

Update

The team fixed the issue as recommended in the commit `20cdeb5`.

File(s) affected: `PositionLibrary.sol`

Description: When increasing the deposit of a position through `PositionLibrary.increaseDeposit()`, the following check is performed:

```
_require(depositedAmountInBorrowed >= params.amountOutMin, ...)
```

However, this check will also be performed in case `params.asset == borrowedAsset`, meaning that no swap is executed. Therefore, this swap may revert if the user has provided `_amountOutMin` in `increaseDeposit(..)` of `PositionManager.sol` anyway.

Recommendation: Move the check into the `if (params.asset != borrowedAsset) {}` block such that it is only performed when a swap is actually executed.

PRI-45 Unclear Whether Native Currency Should Be Supported for Swapping

Informational · Fixed

Update

The team removed the ability to lock `NATIVE_CURRENCY` in the `TraderBalanceVault.lockAsset()` function, which will stop the `SwapManager.swap()` from accepting `NATIVE_CURRENCY`. The change is in the commit `19b757a`.

File(s) affected: `SwapManager.sol`

Description: When swapping from the wallet through the `swap()` function, the input token `params.tokenA` can be `NATIVE_CURRENCY`. The current implementation does not accept `NATIVE_CURRENCY` when swapping directly from the wallet through the `TokenTransfersLibrary.doTransferFromTo()` function. However, the implementation accepts `NATIVE_CURRENCY` when using funds from the `traderBalanceVault.lockAsset()`.

Nonetheless, the call to `PrimexPricingLibrary.multiSwap()` will eventually fail. However, it might be great to clarify whether native token should be supported or not and add the corresponding validation.

Recommendation: Clarify if this is an intentional design; otherwise, update the function to allow swapping `NATIVE_CURRENCY` with the `WETH` contract automatically.

PRI-46
Application Monitoring Can Be Improved by Emitting More Events

Informational | Fixed

> **Update**
> The team had add all suggested events in the commit `5fdd555` .

File(s) affected: `all contracts`

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 transitions 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 application management:

1. `Bucket.setReserveRate()`
2. `Bucket.setWithdrawalFee()`
3. `Bucket.setInterestRateStrategy()`
4. `PositionManager.setKeeperRewardDistributor()`
5. `PositionManager.setMinPositionSize()`
6. `PositionManager.setOracleTolerableLimitMultiplier()`
7. `Reserve.setTransferRestrictions()`
8. `PrimexDNS.setConditionalManager()`
9. `PrimexDNS.setPMX()`
10. `PrimexDNS.setProtocolRate`
11. `PrimexDNS.setProtocolRateInPMX`
12. `PrimexDNS.setAavePool()`
13. `PrimexDNS.activateBucket()`
14. `PrimexDNS.freezeBucket()`
15. `PrimexDNS.setDexAdapter()`
16. `PrimexDNS.activateDEX()`
17. `PrimexDNS.freezeDEX()`
18. `PriceOracle.updatePriceFeed()`
19. `PriceOracle.updateVolatilityFeed()`
20. `PriceOracle.setGasPriceFeed()`
21. `KeeperRewardDistributor.setDefaultMaxGasPrice()`
22. `KeeperRewardDistributor.setOracleGasPriceTolerance()`
23. `KeeperRewardDistributor.setMaxGasAmount`
24. `KeeperRewardDistributor.setPmxPartInReward()`
25. `KeeperRewardDistributor.setNativePartInReward()`
26. `KeeperRewardDistributor.setPositionSizeCoefficients()`
27. `KeeperRewardDistributor.setAdditionalGas()`
28. `SpotTradingRewardDistributor.decreaseRewardPerPeriod()`
29. `SpotTradingRewardDistributor.setRewardPerPeriod()`
30. `SpotTradingRewardDistributor.topUpAvailablePmxBalance()`
31. `SpotTradingRewardDistributor.withdrawPmx()`
32. `Treasury.setMaxSpendingLimit()`
33. `Treasury.decreaseLimits()`
34. `ReferralProgram.setReferrals()`
35. `DexAdapter.setQuoter()`
36. `DexAdapter.setDexType()`
37. `Redeemer.changeRate()`
38. `Redeemer.adminClaimEarlyTokens()`
39. `Redeemer.adminClaimRegularTokens()`
40. `SwapManager.setSwapRate()`
41. `SwapManager.setSwapRateInPmx()`
42. `PMXBonusNFT.setExecutor()`
43. `PMXBonusNFT.blockNFT()`
44. `PMXBonusNFT.unblockNFT()`

Recommendation: Check whether the team can benefit from emitting the events as suggested.

PRI-47 Incompatibility with Non-Standard Tokens

Informational | Acknowledged

> **Update**
> The team acknowledged the issue with the following statement:
>
> > Acceptable. Yes, we understand that the current version of the protocol only works with ERC20 tokens with <=18 decimals, without a fee on transfer, and without tokens with arbitrary changeable balances. The support for more "exotic" tokens will be introduced in future versions.

File(s) affected: `TraderBalanceVault.sol`, `TokenTransfersLibrary.sol`, `BatchManager.sol`, `LimitPriceCOM.sol`, `TakeProfitStopLossCCM.sol`, `BestDexLens.sol`, `PositionLibrary.sol`

Description: In general, the protocol has some limits on what the token can be supported to interact:

1. The protocol only supports tokens with at most 18 decimals: the protocol aligns the "multiplier" on the decimals to 18 in several places: `BatchManager`, `LimitPriceCOM`, `TakeProfitStopLossCCM`, `BestDexLens`, and `PositionLibrary` contracts. The 18 decimals are also how the `WadRayMath` library handles the WAD precision.
2. The protocol does not support token charging fees on transfers: despite that the `TokenTransfersLibrary.doTransferFromTo()` function returns the balance difference as the output, all places calling the function do not use the output. In other words, all places calling the function assume to get the full amount on transfer.
3. The protocol does not support rebaseable tokens: the contract `TraderBalanceVault` maintains the accounting of token balances in `balances`. Tokens with elastic supply (also known as rebasing tokens) and tokens with dishonest implementation would not be compatible with the logic intended by this contract.

Recommendation: It is imperative that the team understands the implementation of the ERC20 tokens used with this protocol, and attests to their compatibility before allowing the tokens in the protocol.

PRI-48 Upgradability

• Informational ⓘ Acknowledged

Update

The team provided the following explanation of their upgrade mechanism:

> The upgradability of all upgradable contracts will be managed through the PrimexProxyAdmin contract, which allows only the role BIG_TIMELOCK_ADMIN to make upgrades. This role will be granted only to the timelock contract with a 10-day delay. Actions can be proposed and executed only by a multisig contract managed by the Primex core team.
>
> Users will be able to learn about an upcoming upgrade 10 days before it occurs. As a means to display this information, external dashboards such as Dune analytics will be set up. It's important to note that there are no mechanisms in Primex contracts to prevent users from withdrawing their funds. Pausing blocks only depositing and borrowing, with no impact on withdrawal or returning funds. Therefore, if a user does not like a proposed upgrade, they have sufficient time to withdraw their assets.
>
> A special role called GUARDIAN_ADMIN, granted to another multisig, can cancel any proposal sent to the BIG/MEDIUM/SMALL timelock contract or pause any of these contracts in critical situations, such as when the admin multisig is compromised. GUARDIAN_ADMIN serves as a last protection, keeping the protocol in a stable state and blocking potentially malicious changes. However, GUARDIAN_ADMIN cannot propose or execute proposals in the timelocks.
>
> Documentation will include information about the protocol management system.

File(s) affected: `all upgradeable contracts`

Description: While upgradability is not a vulnerability in itself, token holders should be aware that the token contract can be upgraded at any given time. This audit does not guarantee the behavior of future contracts that the token may be upgraded to.

Here is the list of upgradable contracts:

```
▶ grep -rl --exclude="I*" --exclude-dir="mocks" "function initialize" contracts
contracts/SpotTradingRewardDistributor/SpotTradingRewardDistributor.sol
contracts/TraderBalanceVault/TraderBalanceVault.sol
contracts/PMXBonusNFT/PMXBonusNFT.sol
contracts/ActivityRewardDistributor/ActivityRewardDistributor.sol
contracts/PToken/PToken.sol
contracts/PriceOracle/PriceOracle.sol
contracts/WhiteBlackList/WhiteBlackListReferral/WhiteBlackListReferral.sol
contracts/WhiteBlackList/WhiteBlackList/WhiteBlackList.sol
contracts/LimitOrderManager/LimitOrderManager.sol
contracts/KeeperRewardDistributor/KeeperRewardDistributor.sol
contracts/Bucket/Bucket.sol
contracts/LiquidityMiningRewardDistributor/LiquidityMiningRewardDistributor.sol
contracts/DebtToken/DebtToken.sol
contracts/Reserve/Reserve.sol
contracts/ReferralProgram/ReferralProgram.sol
contracts/PrimexDNS/PrimexDNS.sol
contracts/BonusExecutor/FeeExecutor.sol
contracts/BonusExecutor/FeeDecreaser.sol
contracts/Treasury/Treasury.sol
contracts/PositionManager/PositionManager.sol
```

Recommendation: The fact that the contract can be upgraded and reasons for future upgrades should be communicated to users beforehand.

PRI-49 Unresolved `TODO`

Informational · Fixed

Update
The team fixed the issue by removing the function as recommended in the commit `0b87b9e`.

File(s) affected: `PrimexPricingLibrary.sol`

Description: The `PrimexPricingLibrary.getLiquidiationPriceByOrder()` function has an unresolved `TODO` stating: "This function is only used in tests. Maybe it can be removed."

Recommendation: Address the TODO by either removing the function or removing the comment. If it is just for testing, it might be worth moving the functionality into testing/mocking contracts instead.

PRI-50 `depositAndLock()` Not Directly Tied to a Deposit

Informational · Fixed

Update
The function is renamed as `increaseLockedBalance()` in the commit `ccbdaf7`.

File(s) affected: `TraderBalanceVault.sol`

Description: The `TraderBalanceVault.depositAndLock()` function allows the `VAULT_ACCESS_ROLE` to add any amount of any asset to any trader's locked balance. However, this function does not actually require that a deposit takes place, contrary to what the function name suggests.

Recommendation: Consider moving the token `transferFrom()` logic inside this function, and also check the `msg.value` if it is for `NATIVE_CURRENCY`.

Alternatively, consider renaming the function to better describe its intention, and add some warning comments on the pre-requisite of the token transfer before calling this function.

PRI-51 Infinite Token Approval

Informational · Fixed

Update
Infinite approvals are removed in the commit `afe7eda`.

File(s) affected: `ActivityRewardDistributor.sol`, `Bucket.sol`, `LiquidityMiningRewardDistributor.sol`, `SpotTradingRewardDistributor.sol`

Description: Several contracts approves token infinitely for another contract. While we do not find any exploit from this, and the risk is mitigated as it is only approved to another contract within the protocol, it increase the risk surface as a vulnerability can cause the token on the approved contracts to be all taken away. We found the pattern in the following places:

1. `ActivityRewardDistributor.initialize()`
2. `Bucket.initialize()`
3. `LiquidityMiningRewardDistributor.initialize()`
4. `SpotTradingRewardDistributor.initialize()`

Recommendation: Consider avoiding the pattern and approving just the needed amount on demand.

PRI-52 Uninitialized Implementation Contract

Informational · Fixed

Update
The team fixed the issue as recommended in the commit `e9d04de`.

File(s) affected: `all upgradeable contracts`

Description: Leaving an implementation contract uninitialized poses a security risk. An uninitialized implementation contract can be exploited by attackers, potentially compromising the associated proxy. To mitigate this risk, it is recommended to invoke the `_disableInitializers()` function in the constructor during deployment (see: OpenZeppelin doc). This action will automatically lock the implementation contract, preventing any unauthorized usage.

The following is the list of impacted contracts found with the `grep` command:

```
▶ grep -rl --include="*.sol" --exclude="I*.sol" --exclude-dir="mocks" "initializer" .
./SpotTradingRewardDistributor/SpotTradingRewardDistributor.sol
./TraderBalanceVault/TraderBalanceVault.sol
./PMXBonusNFT/PMXBonusNFT.sol
./ActivityRewardDistributor/ActivityRewardDistributor.sol
./PToken/PToken.sol
./PriceOracle/PriceOracle.sol
./WhiteBlackList/WhiteBlackListReferral/WhiteBlackListReferral.sol
./WhiteBlackList/WhiteBlackList/WhiteBlackList.sol
./LimitOrderManager/LimitOrderManager.sol
./KeeperRewardDistributor/KeeperRewardDistributor.sol
./Bucket/Bucket.sol
./LiquidityMiningRewardDistributor/LiquidityMiningRewardDistributor.sol
./DebtToken/DebtToken.sol
./Reserve/Reserve.sol
./ReferralProgram/ReferralProgram.sol
./PrimexDNS/PrimexDNS.sol
./BonusExecutor/FeeDecreaser.sol
./Treasury/Treasury.sol
./PositionManager/PositionManager.sol
```

Recommendation: Add the following code to the implementation contracts as listed in the description section.

```
constructor() {
    _disableInitializers();
}
```

PRI-53 Using Deprecated Function

Informational | Fixed

Update

`_grantRole()` is used in the commit `8c74eb8`.

File(s) affected: `AccessControl.sol`

Description: The function `AccessControl._setupRole()` has been deprecated in favor of `AccessControl._grantRole()`, which performs the same operations.

Recommendation: Consider replacing the usage of `_setupRole()` with `_grantRole()`.

PRI-54 Referral Program Susceptible to Sybil Attack

Undetermined | Acknowledged

Update

The team clarified that they will distribute the reward base on the real on-chain activities, and the contract simply serve the purpose of recording the linkage.

Here is the detailed explanation from the team:

> Partially related to PRI-9, referral program smart contracts serve only one purpose - to store connections between users. The reward distribution for referrers will be calculated off-chain, taking into account the referral connections in different networks. In case of any conflicts, the connection created earlier will be considered correct. The distribution of rewards mainly depends on on-chain activity, such as provided liquidity and trading volumes. Therefore, the rewards will be based on the real value that referees bring to the protocol in comparison to the value brought by all Primex users.
>
> In the "worst" scenario, all our users will be referees of several or a single referrer who invited them all. However, this scenario is almost impossible, considering that anyone can become a referrer. In order to receive the majority of referral rewards, a cheating referrer would have to generate the majority of protocol volume, provide the majority of liquidity, and pay the majority of fees. This would be difficult and expensive in a competitive environment.
>
> This reward distribution mechanism, based on real volumes, will be Sybil-resistant to bots that only generate referral connections.

File(s) affected: `ReferralProgram.sol`

Description: Any user can register a referrer by calling the `ReferralProgram.register()` function. This could potentially be dangerous as there is no way to verify that a single account belongs to a single human. Thus, if a reward was associated with referrals, a user would be incentivized to create and refer fake accounts to amass rewards.

Since the reward mechanism for the referral program is unclear to us, we consider the severity to be undetermined.

Recommendation: The team should clarify if this should be intended behavior or not. If not, the following are some potential directions to mitigate this:

1. Implement a proof of humanity mechanism alongside the referral program.
2. Do not attach valuable rewards to referrals.

PRI-55 `PositionManager` and `BatchManager` do not need `NO_FEE_ROLE`

Low Fixed

Update

The client has fixed the issue and revoked `NO_FEE_ROLE` from both contracts in commit `0763077`.

File(s) affected: `batchManager.deploy.js`, `positionManager.deploy.js`

Description: The deployment process grants `NO_FEE_ROLE` to `PositionManager` and `BatchManager` in addition to `LimitOrderManager` and `Bucket`. The `NO_FEE_ROLE` is given to contracts to avoid protocol fees when swapping with `SwapManager`. The `PositionManager` and `BatchManager` do not need `NO_FEE_ROLE` because the contracts do not use `SwapManager` to conduct swaps.

Recommendation: Revoke `NO_FEE_ROLE` from PositionManager and BatchManager contracts.

PRI-56 Proposers Incorrectly Set to Executors

Low Fixed

Update

The team clarified their intention to use the same address for both the executors and the proposers on the initial launch. However, they did fix the script in commit `027fae1`.

File(s) affected: `PrimexTimelock.deploy.js`

Description: During the deployment process of `PrimexTimelock`, the proposers are incorrectly parsed from executors. Therefore the proposers are set as the executors regardless of the proposers provided in the task parameters.

Recommendation: When deploying `PrimexLock`, the proposers should be parsed from proposers provided in the task parameters as best practice.

PRI-57 Missing `GUARDIAN_ADMIN` role

Low Fixed

Update

The team clarified that this is an intentional setup. This role will be given to guardian multi-sig, which did not exist at the time of the deployment review. When executing scripts for full transfer of access roles, this role will be assigned to the guardian multi-sig.

Update

The team has granted the role in the tx:
https://polygonscan.com/tx/0×6efc469e513bc54d847db61896008856f1c63c6d5d8459549e1b7ba8715e7676.

The role is assigned to `0x081079f40CCd0A75344E843976bfaf176a3cb228`, which is a GnosisSafe multi-sig.

File(s) affected: `Registry.deploy.js`, `addresses.json`

Description: During `Registry` deployment, if `guardianAddress` is undefined from the `addresses.json` configuration, the role `GUARDIAN_ADMIN` will not be added to the `Registry`. The `guardianAddress` is not specified in the configuration. The guardian role is needed in PrimexTimelock to pause and unpause functionalities.

Recommendation: Add `GUARDIAN_ADMIN` address role in Registry.

PRI-58 Missing DNS Address

Informational Acknowledged

Update

The client stated that this variable is not used in the contract and stated that the state variable should be removed in the future version of `SpotTradingRewardDistributor` that will be deployed together with the PMX token and will work with this token.

File(s) affected: `SpotTradingRewardDistributor.sol`, `SpotTradingRewardDistributorStorage.sol`

Description: The `SpotTradingRewardDistributorStorage` contract declares the `dns` variable, but the `SpotTradingRewardDistributor.initialized()` function does not set its value. The `dns` variable is not used in `SpotTradingRewardDistributor`, but if there is a function that requires the use of `dns`, the variable should be initialized during a contract upgrade.

Recommendation: No action is required, but the client should be aware of the missing DNS address should a future functionality require DNS lookup.

PRI-59
Different Solidity Compiler Used For Proxy and Implementation Contracts

Informational | Acknowledged

Update
The team clarified that they use a hardhat-deploy plugin that stores pre-compiled proxy contracts (see: link) with v0.8.10.

File(s) affected: `Proxy Contracts`

Description: The proxy contracts used in the Transparent Upgradeable Proxy pattern are compiled using solidity 0.8.10, but the implementation contracts are compiled using solidity 0.8.18.

Recommendation: It is unclear why 0.8.10 solidity compilation is used for proxy contracts instead of 0.8.18 matching the implementation contract. While using different compiler versions for proxy and implementation contracts might not necessarily impose immediate issues, as best practices, the client should use the same 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).

Code Documentation

1. Fixed In `Reserve.transferToTreasury()`, there is a typo on the variable name `restristions` instead of `restrictions`.
2. Fixed Consider adding more explanation to the fields of the `IBucketStorage.LiquidityMiningParams` struct. For instance, add a comment stating that `isLaunch` indicates the launching status of the bucket. Additionally, specify what "amount" the variable `amountPerUser` is accounting for. **Update:** The team did rename some variables to make them more self-explanatory. Additionally, a scheme with periods of liquidity mining was added in one of the previous commits.
3. Fixed Consider adding comment or rename the `IPrimexDNSStorage.adminDeadline`. The deadline is for the admin to call `Bucket.withdrawAfterDelisting()`.
4. Fixed In `WadRayMath.rmul()`, fix the comment `//rounds to zero if x*y < WAD / 2` to `RAY / 2`.
5. Add a code document explaining the purpose of the `IBucket.depositFromBucket()` function. This function is not commonly used and may be difficult to understand why it is necessary.
6. Fixed In `PositionManager._updateTraderActivity()` function, consider adding a comment stating that the condition `bucket == IBucket(address(0))` is to check whether it is a spot trade or not. Alternatively, use a local variable like `isSpotTrade = bucket ==`

`IBucket(address(0))` to make the intention clear.

7. Some terminology suggestions:
   - Fixed `PositionManager.updatePosition()` would be better named `updatePositionConditions()`
   - Fixed `TraderBalanceVault.lockAsset()` does lock assets, but that's a technical description. What it actually does, is releasing assets from the vault for different kind of operations. A more suiting name would be `useTraderAssets()` or `transferTraderAssets()`.
   - Fixed `CloseBatchPositionVars` holds `length` as variable: confusing naming and should could be `numberOfPositions`
   - Acknowledged `LimitOrderManager.openPositionByOrder()` would be better named `LimitOrderManager.openPositionOrPerformSwapByPriceLimitOrder()`.
   - Acknowledged `LimitOrderLibrary.openPositionByOrder()` would be better named `LimitOrderLibrary.openPositionOrPerformSwapByOrder()`.
   - Fixed `LimitOrderLibrary.hasNoAddressDuplicates()` to `LimitOrderLibrary.hasNoConditionManagerTypeDuplicates()`
   - Acknowledged `Treasury.setMaxSpendingLimit()` to `Treasury.addOrUpdateSpenderSpendingLimit()`
   - Fixed `PMXPriceFeed.sol` to `EPMXPriceFeed.sol`

Adherence to Best Practices

1. Acknowledged Generally, it is good practice to separate the responsibilities of contracts in a clear structure. This includes access control to quickly identify which functionalities are restricted to which roles. In this project, we noticed that the division of responsibilities and access control is unclear and convoluted. As an example, `increaseDeposit()` and `decreaseDeposit()` access control is performed in `PositionLibrary`. However, `updatePosition()` and `partiallyClosePosition()` is checked in `PositionManager`. **Update:** The client prefers to keep it as it is for now to minimize the need for refactoring.
2. Fixed The following function names do not follow the Solidity Style Guide naming convention. Internal functions should start with an underscore:
   1. `DexAdapter.getBalancerSwapSteps()`
   2. `TakeProfitStopLossCCM.calcTakeProfitStopLossAmounts()`
   3. `TrailingStopCCM.getPriceFromFeeds()`
3. Fixed There are two separate errors `BUCKET_IS_INACTIVE` and `BUCKET_IS_NOT_ACTIVE` acting the same purpose, we recommend consolidating the errors. **Update:** The unused error `BUCKET_NOT_ACTIVE` has been removed. Additionally, the client clarifies that "BUCKET_IS_NOT_ACTIVE" indicates that the bucket is either inactive or deprecated, while `BUCKET_IS_INACTIVE` specifically denotes the inactive status. Therefore, these error codes serve distinct purposes.
4. Acknowledged The `BatchManager.closeBatchPositions()` function allows anyone to close positions in batches. This function allows anyone to close positions in batches for a variety of `CloseReason`, making the function complex and reducing readability and maintainability. Consider breaking down the function into smaller, more specialized functions that handle specific tasks. This can help improve code readability, maintainability, and testability. **Update:** The client has stated that they will address this in the future.
5. Fixed Generally, it's advisable to verify transaction conditions before engaging in computations or memory and storage writes, as this approach consumes less gas in case of transaction failures. However, there are exceptions to this rule. For instance, in `LimitOrderLibrary.createLimitOrder()`, a `LimitOrder` struct is committed to memory even prior to validating the provided arguments.
6. Fixed Within the `Treasury` contract, a mapping is defined to correlate token addresses with spender details. If the provided token address is `address(0)`, it signifies the chain's `NATIVE_CURRENCY`. Yet, it's more prudent to employ a distinct address to denote the native currency, lessening the chances of unintentionally assigning a non-zero address. **Update:** Added a distinct address to denote the native currency: `address(uint160(bytes20(keccak256("NATIVE_CURRENCY"))))`
7. Fixed The name `TraderBalanceVault.depositAndLock()` can be misleading, as it implies the function involves both deposit and locking actions. However, it solely augments the `_trader`'s `lockedBalance` by the specified `_amount`, without ensuring an actual deposit transpired. **Update:** `depositAndLock()` has been renamed to `increaseLockedBalance()`.
8. Acknowledged The `onlyRole()` modifier has been replicated across multiple contracts. It's recommended to create a shared contract or library that all other contracts inherit from. The same principle applies to the `_getCurrentPeriod()` function.
9. Fixed The rationale behind granting users the capacity to self-lock their deposits and the potential benefits remains unclear. If no specific purpose warrants this capability, it might be preferable to eliminate this functionality from the `PToken.lockDeposit()` function. **Update:** The team has restricted traders from locking their deposits. Now, this action can only occur in the case of reinvesting after failed liquidity mining.
10. Acknowledged Consider using `IAccessControl` as the type of the `BucketStorage.registry` instead of `address` to make the variable more self-descriptive and enable basic type checks.
11. Fixed Consider using a more appropriate variable name for the `Bucket.setReserveRate()` function input. The `_fee` input variable could be renamed as `_reserveRate` or simply `rate` to better reflect its purpose and improve code readability.
12. Acknowledged The `IBonusExecutor.deactivateBonus()` should be under `IFeeExecutor` interface instead. The `deactivateBonus()` function is implemented in the `FeeExecutor` instead of the `BonusExecutor` contract. **Update:** The team has decided to keep it as it is. They would like to have a `deactivateBonus` method in the `IBonusExecutor` interface, but the state in which it operates is stored in `FeeExecutor`.
13. Fixed Consider removing the unused functions in the `V3Path` library: `hasMultiplePools()`, `decodeFirstPool()`, `getFirstPool()`, and `skipToken()`.
14. Fixed Consider removing the unused functions in the `WadRayMath` library: `rpow()`, `sqrt()`, and `fracPower()`.
15. Fixed At `Bucket.deposit()#L208`, instead of overriding `_pTokenReceiver` to the `msg.sender` in the `else` block, it is better to enforce the `_pTokenReceiver` to be correctly set (e.g., validating `_pTokenReceiver == msg.sender`) instead. This will avoid magic handling, making it hard to understand in the future.
16. Acknowledged While most contracts in the project support ERC165 and have implemented the `supportInterface()` function, there is a possibility that some interfaces that should be supported might have been overlooked in the implementation. We noticed that the `IxxxStorage` interface is not considered as supported in the `supportsInterface()` function while it is inherited. The following is a list

of contracts that might miss the support for interfaces: `Reserve`, `LimitOrderManager`, `ReferralProgram`, `PriceOracle`, `Treasury`. Please consider adding the support of the suggested interfaces.

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

- `f5e...d0a ./contracts/PMXPriceFeed.sol`
- `86f...dbe ./contracts/Redeemer.sol`
- `39e...dbd ./contracts/PrimexUpkeep.sol`
- `325...073 ./contracts/PrimexProxyAdmin.sol`
- `346...306 ./contracts/BatchManager.sol`
- `a7a...1e6 ./contracts/PMXToken.sol`
- `902...f53 ./contracts/DexAdapter.sol`
- `893...5f8 ./contracts/PrimexRegistry.sol`
- `f9d...0f9 ./contracts/PrimexTimelock.sol`
- `9ab...c43 ./contracts/EPMXToken.sol`
- `7aa...1d3 ./contracts/Constants.sol`
- `eb1...d39 ./contracts/InterestRateStrategy.sol`
- `6fa...354 ./contracts/SwapManager.sol`
- `2c9...1a3 ./contracts/UniswapInterfaceMulticall.sol`
- `a0c...adc ./contracts/PositionManager/IBasePositionManager.sol`
- `6a8...1d7 ./contracts/PositionManager/PositionManagerStorage.sol`
- `659...a13 ./contracts/PositionManager/IPositionManagerStorage.sol`
- `b56...bbc ./contracts/PositionManager/IPositionManager.sol`
- `36f...6e4 ./contracts/PositionManager/PositionManager.sol`
- `0a6...1bd ./contracts/interfaces/IPrimexUpkeep.sol`
- `54c...d11 ./contracts/interfaces/IConditionalOpeningManager.sol`
- `6b3...819 ./contracts/interfaces/IWhitelist.sol`
- `279...ac1 ./contracts/interfaces/IBatchManager.sol`
- `10a...5c5 ./contracts/interfaces/IERC20Mock.sol`
- `275...b45 ./contracts/interfaces/IBestDexLens.sol`
- `318...e24 ./contracts/interfaces/IPMXPriceFeed.sol`
- `533...b8f ./contracts/interfaces/ILimitPriceCOM.sol`
- `7ff...50a ./contracts/interfaces/ITrailingStopCCM.sol`
- `baf...a6f ./contracts/interfaces/IPrimexLens.sol`
- `ddd...102 ./contracts/interfaces/IPrimexRegistry.sol`
- `650...a01 ./contracts/interfaces/IPriceFeedUpdaterTestService.sol`
- `1f8...7a2 ./contracts/interfaces/IConditionalClosingManager.sol`
- `7fe...b66 ./contracts/interfaces/ISwapManager.sol`
- `63a...bad ./contracts/interfaces/IDexAdapter.sol`
- `abc...e3f ./contracts/interfaces/ITakeProfitStopLossCCM.sol`
- `5ae...aab ./contracts/interfaces/IPrimexPricingLibraryMock.sol`
- `c8f...be9 ./contracts/interfaces/IPausable.sol`
- `f43...f8e ./contracts/interfaces/IBalancer.sol`
- `b9e...5a4 ./contracts/interfaces/IEPMXToken.sol`
- `4a1...5ca ./contracts/interfaces/IInterestRateStrategy.sol`
- `b03...dc6 ./contracts/interfaces/IAsset.sol`
- `d33...c70 ./contracts/interfaces/IRedeemer.sol`
- `507...571 ./contracts/interfaces/routers/ICurveRouter.sol`
- `187...470 ./contracts/interfaces/routers/ICurvePool.sol`

- 723...3e1 ./contracts/interfaces/routers/ICurveRegistry.sol
- 2d7...ab3 ./contracts/interfaces/routers/ICurveCalc.sol
- 719...5a9 ./contracts/Treasury/ITreasury.sol
- 0b3...896 ./contracts/Treasury/TreasuryStorage.sol
- fef...986 ./contracts/Treasury/ITreasuryStorage.sol
- 0a9...481 ./contracts/Treasury/Treasury.sol
- 3ca...7e5 ./contracts/conditionalManagers/TakeProfitStopLossCCM.sol
- 665...d87 ./contracts/conditionalManagers/LimitPriceCOM.sol
- 1b7...3b7 ./contracts/conditionalManagers/TrailingStopCCM.sol
- 691...9dc ./contracts/BonusExecutor/IFeeExecutor.sol
- dce...8c1 ./contracts/BonusExecutor/IFeeExecutorStorage.sol
- 82e...c66 ./contracts/BonusExecutor/FeeExecutor.sol
- 1c1...6ed ./contracts/BonusExecutor/InterestIncreaser.sol
- 816...298 ./contracts/BonusExecutor/FeeDecreaser.sol
- dab...606 ./contracts/BonusExecutor/BonusExecutor.sol
- 5b6...b89 ./contracts/BonusExecutor/FeeExecutorStorage.sol
- 4df...26e ./contracts/BonusExecutor/IBonusExecutor.sol
- 441...e35 ./contracts/PrimexDNS/IPrimexDNS.sol
- 97c...835 ./contracts/PrimexDNS/PrimexDNS.sol
- 3bb...56b ./contracts/PrimexDNS/PrimexDNSStorage.sol
- 34a...612 ./contracts/PrimexDNS/IPrimexDNSStorage.sol
- b92...21e ./contracts/ReferralProgram/IReferralProgram.sol
- 901...fe0 ./contracts/ReferralProgram/ReferralProgram.sol
- be5...aa2 ./contracts/ReferralProgram/IReferralProgramStorage.sol
- 1a3...80a ./contracts/ReferralProgram/ReferralProgramStorage.sol
- c86...aef ./contracts/Reserve/IReserveStorage.sol
- f8d...d06 ./contracts/Reserve/ReserveStorage.sol
- f6d...9bb ./contracts/Reserve/IReserve.sol
- d61...a53 ./contracts/Reserve/Reserve.sol
- 86f...880 ./contracts/DebtToken/DebtToken.sol
- 276...a83 ./contracts/DebtToken/IDebtTokenStorage.sol
- 1dd...ae1 ./contracts/DebtToken/DebtTokenFactory.sol
- 19a...32e ./contracts/DebtToken/IDebtToken.sol
- 5b3...33b ./contracts/DebtToken/IDebtTokensFactory.sol
- b1b...4c1 ./contracts/DebtToken/DebtTokenStorage.sol
- f5d...d4d ./contracts/LiquidityMiningRewardDistributor/LiquidityMiningRewardDistributor.sol
- f05...90d ./contracts/LiquidityMiningRewardDistributor/ILiquidityMiningRewardDistributor.sol
- 6c4...352 ./contracts/LiquidityMiningRewardDistributor/ILiquidityMiningRewardDistributorStorage.sol
- db9...aea ./contracts/LiquidityMiningRewardDistributor/LiquidityMiningRewardDistributorStorage.sol
- 7a6...5f7 ./contracts/Bucket/IBucketStorage.sol
- 7b4...2f3 ./contracts/Bucket/IBucketsFactory.sol
- 5f4...429 ./contracts/Bucket/BucketStorage.sol
- bff...6c3 ./contracts/Bucket/IBucket.sol
- 9b6...b39 ./contracts/Bucket/Bucket.sol
- 394...c55 ./contracts/Bucket/BucketsFactory.sol
- fd1...57d ./contracts/KeeperRewardDistributor/IKeeperRewardDistributorStorage.sol
- 2f0...36f ./contracts/KeeperRewardDistributor/KeeperRewardDistributorStorage.sol
- d86...8f6 ./contracts/KeeperRewardDistributor/KeeperRewardDistributor.sol
- 24a...d99 ./contracts/KeeperRewardDistributor/IKeeperRewardDistributor.sol
- 461...98a ./contracts/lens/BestDexLens.sol
- 567...a3a ./contracts/lens/PrimexLens.sol
- 1fb...fdc ./contracts/LimitOrderManager/LimitOrderManager.sol
- ebf...469 ./contracts/LimitOrderManager/ILimitOrderManager.sol
- 013...b07 ./contracts/LimitOrderManager/ILimitOrderManagerStorage.sol
- 81c...ecd ./contracts/LimitOrderManager/LimitOrderManagerStorage.sol

- 9c6...231 ./contracts/WhiteBlackList/WhiteBlackListBase/WhiteBlackListBase.sol
- cc0...64f ./contracts/WhiteBlackList/WhiteBlackList/IWhiteBlackList.sol
- b67...422 ./contracts/WhiteBlackList/WhiteBlackList/WhiteBlackList.sol
- c74...f15 ./contracts/WhiteBlackList/WhiteBlackListReferral/WhiteBlackListReferral.sol
- 65e...6d6 ./contracts/WhiteBlackList/WhiteBlackListReferral/IWhiteBlackListReferral.sol
- b82...6b1 ./contracts/WhiteBlackList/WhiteBlackListReferral/WhiteBlackListReferralStorage.sol
- cb2...538 ./contracts/PriceOracle/IPriceOracleStorage.sol
- baa...42d ./contracts/PriceOracle/PriceOracle.sol
- 420...369 ./contracts/PriceOracle/IPriceOracle.sol
- e36...ce2 ./contracts/PriceOracle/PriceOracleStorage.sol
- 645...505 ./contracts/PToken/PTokensFactory.sol
- 2ec...9f3 ./contracts/PToken/IPTokensFactory.sol
- 6fa...884 ./contracts/PToken/IPToken.sol
- 1cc...478 ./contracts/PToken/PToken.sol
- 750...edd ./contracts/PToken/IPTokenStorage.sol
- ac1...d3d ./contracts/PToken/PTokenStorage.sol
- 4e9...b9e ./contracts/ActivityRewardDistributor/IActivityRewardDistributor.sol
- 84c...779 ./contracts/ActivityRewardDistributor/IActivityRewardDistributorStorage.sol
- d11...b24 ./contracts/ActivityRewardDistributor/ActivityRewardDistributor.sol
- 848...d5b ./contracts/ActivityRewardDistributor/ActivityRewardDistributorStorage.sol
- 1c5...040 ./contracts/libraries/PositionLibrary.sol
- 645...b64 ./contracts/libraries/Errors.sol
- 317...918 ./contracts/libraries/PrimexPricingLibrary.sol
- c58...aa1 ./contracts/libraries/TokenTransfersLibrary.sol
- ce0...f27 ./contracts/libraries/LimitOrderLibrary.sol
- be7...9f8 ./contracts/libraries/utils/WadRayMath.sol
- cc6...903 ./contracts/libraries/utils/V3Path.sol
- f46...690 ./contracts/libraries/utils/BytesLib.sol
- 797...46a ./contracts/PMXBonusNFT/PMXBonusNFT.sol
- 409...029 ./contracts/PMXBonusNFT/IPMXBonusNFT.sol
- 39a...250 ./contracts/PMXBonusNFT/IPMXBonusNFTStorage.sol
- b4a...7c1 ./contracts/PMXBonusNFT/PMXBonusNFTStorage.sol
- b74...f4b ./contracts/TraderBalanceVault/TraderBalanceVaultStorage.sol
- bac...70f ./contracts/TraderBalanceVault/TraderBalanceVault.sol
- 94e...1a9 ./contracts/TraderBalanceVault/ITraderBalanceVaultStorage.sol
- 2fb...fce ./contracts/TraderBalanceVault/ITraderBalanceVault.sol
- 05b...422 ./contracts/SpotTradingRewardDistributor/SpotTradingRewardDistributor.sol
- 5b7...1c3 ./contracts/SpotTradingRewardDistributor/ISpotTradingRewardDistributor.sol
- f55...a64 ./contracts/SpotTradingRewardDistributor/SpotTradingRewardDistributorStorage.sol
- b35...51f ./contracts/SpotTradingRewardDistributor/ISpotTradingRewardDistributorStorage.sol
- 315...cb2 ./contracts/TestnetServices/PrimexAggregatorV3TestService.sol
- fad...1d5 ./contracts/TestnetServices/BalancerBotLens.sol
- d0d...747 ./contracts/TestnetServices/SynchronizationBotLensQuickswapTestService.sol
- 8cd...a3e ./contracts/TestnetServices/CurveBotLens.sol
- f0a...57e ./contracts/TestnetServices/SynchronizationBotLens.sol
- 186...810 ./contracts/TestnetServices/SynchronizationBotLensUniV3TestService.sol
- 23b...cf1 ./contracts/TestnetServices/PriceFeedUpdaterTestService.sol
- 399...d4c ./contracts/TestnetServices/interfaces/ISynchronizationBotLensQuickswapTestService.sol
- 94a...d6c ./contracts/TestnetServices/interfaces/IPrimexAggregatorV3TestService.sol
- d8b...0ea ./contracts/TestnetServices/interfaces/IBalancerBotLens.sol
- 041...4b7 ./contracts/TestnetServices/interfaces/ICurveBotLens.sol
- 560...5bf ./contracts/TestnetServices/interfaces/ISynchronizationBotLensUniV3TestService.sol
- 3c3...cd5 ./contracts/TestnetServices/interfaces/ISynchronizationBotLens.sol
- 395...8d4 ./contracts/hardhat-dependency-compiler/@openzeppelin/contracts/token/ERC20/IERC20.sol
- df9...66e ./contracts/mocks/BucketMock.sol

- 2d3...19a ./contracts/mocks/MaliciousDexMock.sol
- a8a...032 ./contracts/mocks/ERC20Mock.sol
- b41...639 ./contracts/mocks/AttackerBucket.sol
- 782...8ea ./contracts/mocks/TokenTransfersLibraryMock.sol
- ba8...fad ./contracts/mocks/PrimexPricingLibraryMock.sol
- 6a2...1d7 ./contracts/mocks/MockNearestSearch.sol
- fbf...e23 ./contracts/mocks/upgradeMocksInterfaces/ILimitOrderManagerV2.sol
- 6cb...fef ./contracts/mocks/upgradeMocksInterfaces/IDebtTokenV2.sol
- 512...eee ./contracts/mocks/upgradeMocksInterfaces/IPriceOracleV2.sol
- bc0...a93 ./contracts/mocks/upgradeMocksInterfaces/IFeeDecreaserV2.sol
- 90f...e0d ./contracts/mocks/upgradeMocksInterfaces/ITreasuryV2.sol
- c7d...f79 ./contracts/mocks/upgradeMocksInterfaces/ITraderBalanceVaultV2.sol
- c76...7af ./contracts/mocks/upgradeMocksInterfaces/IPMXBonusNFTV2.sol
- d85...26d ./contracts/mocks/upgradeMocksInterfaces/IBucketV2.sol
- 729...b27 ./contracts/mocks/upgradeMocksInterfaces/IKeeperRewardDistributorV2.sol
- 683...22e ./contracts/mocks/upgradeMocksInterfaces/IReserveV2.sol
- d1f...6ff ./contracts/mocks/upgradeMocksInterfaces/IReferralProgramV2.sol
- 4f9...a06 ./contracts/mocks/upgradeMocksInterfaces/IPTokenV2.sol
- 8bf...924 ./contracts/mocks/upgradeMocksInterfaces/ISpotTradingRewardDistributorV2.sol
- 6c4...cac ./contracts/mocks/upgradeMocksInterfaces/ILiquidityMiningRewardDistributorV2.sol
- 3e5...6ca ./contracts/mocks/upgradeMocksInterfaces/IPrimexDNSV2.sol
- f9c...c4f ./contracts/mocks/upgradeMocksInterfaces/IInterestIncreaserV2.sol
- 675...c64 ./contracts/mocks/upgradeMocksInterfaces/IPositionManagerV2.sol
- 98b...64f ./contracts/mocks/upgradeMocksInterfaces/IWhiteBlackListReferralV2.sol
- a93...84e ./contracts/mocks/upgradeMocks/KeeperRewardDistributorV2.sol
- cab...d62 ./contracts/mocks/upgradeMocks/LiquidityMiningRewardDistributorV2.sol
- afe...999 ./contracts/mocks/upgradeMocks/ReserveV2.sol
- a1d...0b1 ./contracts/mocks/upgradeMocks/TraderBalanceVaultV2.sol
- 354...435 ./contracts/mocks/upgradeMocks/InterestIncreaserV2.sol
- 98c...b5e ./contracts/mocks/upgradeMocks/ReferralProgramV2.sol
- d24...72a ./contracts/mocks/upgradeMocks/SpotTradingRewardDistributorV2.sol
- 52d...f92 ./contracts/mocks/upgradeMocks/FeeDecreaserV2.sol
- 5e6...7f7 ./contracts/mocks/upgradeMocks/PositionManagerV2.sol
- 7d9...f66 ./contracts/mocks/upgradeMocks/PMXBonusNFTV2.sol
- 5ee...a6d ./contracts/mocks/upgradeMocks/PriceOracleV2.sol
- 0db...435 ./contracts/mocks/upgradeMocks/DebtTokenV2.sol
- c56...b49 ./contracts/mocks/upgradeMocks/PTokenV2.sol
- 4b8...c48 ./contracts/mocks/upgradeMocks/WhiteBlackListReferralV2.sol
- 5ca...37b ./contracts/mocks/upgradeMocks/BucketV2.sol
- 51c...8d6 ./contracts/mocks/upgradeMocks/LimitOrderManagerV2.sol
- be3...c1f ./contracts/mocks/upgradeMocks/PrimexDNSV2.sol
- 5f2...1af ./contracts/mocks/upgradeMocks/TreasuryV2.sol
- 1c7...408 ./contracts/mocks/mocksInterfaces/IMockNearestSearch.sol
- 5a0...45f ./contracts/mocks/mocksInterfaces/IAttackerBucket.sol
- 96b...b39 ./contracts/mocks/mocksInterfaces/IMaliciousDexMock.sol
- ab8...d36 ./contracts/mocks/mocksInterfaces/IBucketMock.sol

Tests

- 560...39f ./test/PrimexLens.test.js
- 147...3b2 ./test/DepositAsset_isPositionAsset.test.js
- d80...a3c ./test/BestDexLens.test.js
- 98d...91d ./test/PositionBatchClose.test.js
- 205...de6 ./test/CrossDexClosePosition.test.js
- f80...17c ./test/PrimexUpkeep.test.js

- bcb...baa ./test/DebtToken.test.js
- 6d0...079 ./test/DexAdapterSwapPath.test.js
- cef...bcd ./test/CurveBotLens.test.js
- 7d7...652 ./test/DexAdapter.test.js
- dc4...df2 ./test/DepositAsset_isAssetNotFromSwapPair.test.js
- 0fd...6cb ./test/PToken.test.js
- 106...bcb ./test/ReferralProgram.test.js
- ddd...fc6 ./test/PriceFeedsAccessControl.test.js
- 160...db0 ./test/PrimexDNS.test.js
- c4e...546 ./test/PhaseSwitching.test.js
- eab...f61 ./test/BucketsFactory.test.js
- c3f...33a ./test/Bucket.test.js
- 8b9...e18 ./test/LimitedMintTokens.test.js
- c3c...7e1 ./test/PositionManager.test.js
- 0fd...4ed ./test/PMXToken.test.js
- 1eb...e8f ./test/SpotTrading.test.js
- c79...f32 ./test/Balancer.test.js
- 982...7b4 ./test/LimitOrderManager.test.js
- 956...1a5 ./test/ChainLinkUpKeep.test.js
- 189...ea3 ./test/PriceFeedUpdaterTestService.test.js
- d17...8f7 ./test/utils/constants.js
- 9dd...885 ./test/utils/hardhatUtils.js
- 011...a81 ./test/utils/setBadOraclePrice.js
- 1ec...28e ./test/utils/activityRewardDistributorMath.js
- 252...9c5 ./test/utils/defaultBarCalcParams.js
- 231...6a9 ./test/utils/makeWalletFromPrivateKey.js
- 614...ad6 ./test/utils/bnMath.js
- ed8...cd2 ./test/utils/dexOperations.js
- 2f1...09a ./test/utils/conditionParams.js
- 343...7ad ./test/utils/math.js
- d09...379 ./test/utils/eventValidation.js
- b3a...a89 ./test/utils/addressFromEvent.js
- aa5...7ee ./test/utils/encodePriceSqrt.js
- 4f9...ed7 ./test/utils/generateSignature.js
- 1fc...ed6 ./test/utils/waffleMocks.js
- 8a1...6ee ./test/integration/TraderRewardDistributor.test.js
- 571...fb6 ./test/integration/PrimexPricingLibrary.test.js
- a88...f36 ./test/integration/InterestRateStrategy.test.js
- f72...757 ./test/integration/Upgradability.test.js
- 58c...4a2 ./test/integration/OperationWithBalancer.test.js
- 8dd...1e6 ./test/integration/SwapManager.test.js
- 945...b04 ./test/integration/Reserve.test.js
- 4d5...053 ./test/integration/WhiteBlackList.test.js
- d6b...436 ./test/integration/KeeperRewardDistributor.test.js
- 833...4ce ./test/integration/LiquidityMiningRewardDistributor.test.js
- 8df...fff ./test/integration/TakeProfitStopLossCCM.test.js
- 5aa...5ae ./test/integration/TraderBalanceVault.test.js
- c55...f81 ./test/integration/Redeemer.test.js
- 696...188 ./test/integration/LenderRewardDistributor.test.js
- f6d...aa3 ./test/integration/OperationWithCurve.test.js
- f2e...b6e ./test/integration/PriceOracle.test.js
- 436...08f ./test/integration/SpotTradingRewardDistributor.test.js
- 674...47b ./test/integration/LimitPriceCOM.test.js
- 6c5...bda ./test/integration/PrimexTimelocks.test.js
- 948...2da ./test/integration/TrailingStopCCM.test.js

- `bce...678 ./test/integration/Treasury.test.js`
- `e6e...eed ./test/integration/PMXBonusNFT.test.js`
- `1e9...09c ./test/unit/PrimexPricingLibrary.test.js`
- `906...1dd ./test/unit/InterestRateStrategy.test.js`
- `64d...592 ./test/unit/SwapManager.test.js`
- `9d6...201 ./test/unit/ActivityRewardDistributor.test.js`
- `97d...991 ./test/unit/Reserve.test.js`
- `6a0...7c0 ./test/unit/WhiteBlackList.test.js`
- `32f...de3 ./test/unit/EPMXToken.test.js`
- `b1e...cd6 ./test/unit/KeeperRewardDistributor.test.js`
- `fd2...4d2 ./test/unit/LiquidityMiningRewardDistributor.test.js`
- `40d...125 ./test/unit/ReferralProgram.test.js`
- `b29...128 ./test/unit/TakeProfitStopLossCCM.test.js`
- `e94...ff2 ./test/unit/TraderBalanceVault.test.js`
- `317...15d ./test/unit/Redeemer.test.js`
- `9f5...99d ./test/unit/PMXTokenOracle.test.js`
- `e14...726 ./test/unit/PriceOracle.test.js`
- `a62...bf1 ./test/unit/SpotTradingRewardDistributor.test.js`
- `9b6...d5a ./test/unit/PrimexProxyAdmin.test.js`
- `711...9cd ./test/unit/LimitPriceCOM.test.js`
- `21d...a98 ./test/unit/PositionManager.test.js`
- `63a...42c ./test/unit/PrimexTimelocks.test.js`
- `6e2...f79 ./test/unit/BonusExecutor.test.js`
- `d5b...2fd ./test/unit/TrailingStopCCM.test.js`
- `028...00b ./test/unit/Treasury.test.js`
- `6ce...2f5 ./test/unit/PrimexRegistry.test.js`
- `ab2...2b6 ./test/unit/PMXBonusNFT.test.js`

Toolset

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

Setup

Tool Setup:

- Slither v0.9.3

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 identified 1421 issues during the analysis of the project. However, most of them were false positives or out of scope, and only valid ones have been included in the report.

Test Suite Results

We followed the README, and run the following commands:

```
cd src
yarn install
yarn test
```

```
yarn run v1.22.19
$ yarn lint && yarn hardhat test
$ yarn prettier:solidity && yarn prettier:js && yarn eslint:all-fix && yarn solhint:all
$ ./node_modules/.bin/prettier --write contracts/*.sol contracts/**/*.sol contracts/**/**/*.sol
contracts/BatchManager.sol 973ms
contracts/Constants.sol 21ms
contracts/DexAdapter.sol 996ms
contracts/EPMXToken.sol 104ms
contracts/InterestRateStrategy.sol 33ms
contracts/PMXPriceFeed.sol 46ms
contracts/PMXToken.sol 7ms
contracts/PrimexProxyAdmin.sol 52ms
contracts/PrimexRegistry.sol 18ms
contracts/PrimexTimelock.sol 32ms
contracts/PrimexUpkeep.sol 231ms
contracts/Redeemer.sol 33ms
contracts/SwapManager.sol 91ms
contracts/UniswapInterfaceMulticall.sol 25ms
contracts/ActivityRewardDistributor/ActivityRewardDistributor.sol 253ms
contracts/ActivityRewardDistributor/ActivityRewardDistributorStorage.sol 4ms
contracts/ActivityRewardDistributor/IActivityRewardDistributor.sol 24ms
contracts/ActivityRewardDistributor/IActivityRewardDistributorStorage.sol 5ms
contracts/BonusExecutor/BonusExecutor.sol 41ms
contracts/BonusExecutor/FeeDecreaser.sol 76ms
contracts/BonusExecutor/FeeExecutor.sol 132ms
contracts/BonusExecutor/FeeExecutorStorage.sol 3ms
contracts/BonusExecutor/IBonusExecutor.sol 3ms
contracts/BonusExecutor/IFeeExecutor.sol 5ms
contracts/BonusExecutor/IFeeExecutorStorage.sol 4ms
contracts/BonusExecutor/InterestIncreaser.sol 55ms
contracts/Bucket/Bucket.sol 361ms
contracts/Bucket/BucketStorage.sol 6ms
contracts/Bucket/BucketsFactory.sol 81ms
contracts/Bucket/IBucket.sol 15ms
contracts/Bucket/IBucketStorage.sol 11ms
contracts/Bucket/IBucketsFactory.sol 5ms
contracts/DebtToken/DebtToken.sol 90ms
contracts/DebtToken/DebtTokenFactory.sol 9ms
contracts/DebtToken/DebtTokenStorage.sol 2ms
contracts/DebtToken/IDebtToken.sol 5ms
contracts/DebtToken/IDebtTokenStorage.sol 2ms
contracts/DebtToken/IDebtTokensFactory.sol 2ms
contracts/KeeperRewardDistributor/IKeeperRewardDistributor.sol 5ms
contracts/KeeperRewardDistributor/IKeeperRewardDistributorStorage.sol 9ms
contracts/KeeperRewardDistributor/KeeperRewardDistributor.sol 70ms
contracts/KeeperRewardDistributor/KeeperRewardDistributorStorage.sol 3ms
contracts/LimitOrderManager/ILimitOrderManager.sol 11ms
contracts/LimitOrderManager/ILimitOrderManagerStorage.sol 3ms
contracts/LimitOrderManager/LimitOrderManager.sol 288ms
contracts/LimitOrderManager/LimitOrderManagerStorage.sol 7ms
contracts/LiquidityMiningRewardDistributor/ILiquidityMiningRewardDistributor.sol 9ms
contracts/LiquidityMiningRewardDistributor/ILiquidityMiningRewardDistributorStorage.sol 6ms
contracts/LiquidityMiningRewardDistributor/LiquidityMiningRewardDistributor.sol 150ms
contracts/LiquidityMiningRewardDistributor/LiquidityMiningRewardDistributorStorage.sol 5ms
contracts/PMXBonusNFT/IPMXBonusNFT.sol 5ms
contracts/PMXBonusNFT/IPMXBonusNFTStorage.sol 3ms
contracts/PMXBonusNFT/PMXBonusNFT.sol 73ms
contracts/PMXBonusNFT/PMXBonusNFTStorage.sol 7ms
contracts/PToken/IPToken.sol 6ms
contracts/PToken/IPTokenStorage.sol 3ms
contracts/PToken/IPTokensFactory.sol 3ms
contracts/PToken/PToken.sol 145ms
contracts/PToken/PTokenStorage.sol 3ms
contracts/PToken/PTokensFactory.sol 9ms
contracts/PositionManager/IBasePositionManager.sol 2ms
contracts/PositionManager/IPositionManager.sol 16ms
contracts/PositionManager/IPositionManagerStorage.sol 4ms
contracts/PositionManager/PositionManager.sol 245ms
contracts/PositionManager/PositionManagerStorage.sol 8ms
contracts/PriceOracle/IPriceOracle.sol 6ms
```

```
contracts/PriceOracle/IPriceOracleStorage.sol 2ms
contracts/PriceOracle/PriceOracle.sol 69ms
contracts/PriceOracle/PriceOracleStorage.sol 3ms
contracts/PrimexDNS/IPrimexDNS.sol 6ms
contracts/PrimexDNS/IPrimexDNSStorage.sol 4ms
contracts/PrimexDNS/PrimexDNS.sol 68ms
contracts/PrimexDNS/PrimexDNSStorage.sol 3ms
contracts/ReferralProgram/IReferralProgram.sol 5ms
contracts/ReferralProgram/IReferralProgramStorage.sol 2ms
contracts/ReferralProgram/ReferralProgram.sol 38ms
contracts/ReferralProgram/ReferralProgramStorage.sol 2ms
contracts/Reserve/IReserve.sol 4ms
contracts/Reserve/IReserveStorage.sol 5ms
contracts/Reserve/Reserve.sol 38ms
contracts/Reserve/ReserveStorage.sol 2ms
contracts/SpotTradingRewardDistributor/ISpotTradingRewardDistributor.sol 5ms
contracts/SpotTradingRewardDistributor/ISpotTradingRewardDistributorStorage.sol 3ms
contracts/SpotTradingRewardDistributor/SpotTradingRewardDistributor.sol 62ms
contracts/SpotTradingRewardDistributor/SpotTradingRewardDistributorStorage.sol 3ms
contracts/TestnetServices/BalancerBotLens.sol 46ms
contracts/TestnetServices/CurveBotLens.sol 32ms
contracts/TestnetServices/PriceFeedUpdaterTestService.sol 52ms
contracts/TestnetServices/PrimexAggregatorV3TestService.sol 23ms
contracts/TestnetServices/SynchronizationBotLens.sol 64ms
contracts/TestnetServices/SynchronizationBotLensQuickswapTestService.sol 24ms
contracts/TestnetServices/SynchronizationBotLensUniV3TestService.sol 22ms
contracts/TraderBalanceVault/ITraderBalanceVault.sol 5ms
contracts/TraderBalanceVault/ITraderBalanceVaultStorage.sol 2ms
contracts/TraderBalanceVault/TraderBalanceVault.sol 69ms
contracts/TraderBalanceVault/TraderBalanceVaultStorage.sol 7ms
contracts/Treasury/ITreasury.sol 4ms
contracts/Treasury/ITreasuryStorage.sol 2ms
contracts/Treasury/Treasury.sol 47ms
contracts/Treasury/TreasuryStorage.sol 2ms
contracts/conditionalManagers/LimitPriceCOM.sol 65ms
contracts/conditionalManagers/TakeProfitStopLossCCM.sol 48ms
contracts/conditionalManagers/TrailingStopCCM.sol 26ms
contracts/interfaces/IAsset.sol 1ms
contracts/interfaces/IBalancer.sol 6ms
contracts/interfaces/IBatchManager.sol 4ms
contracts/interfaces/IBestDexLens.sol 9ms
contracts/interfaces/IConditionalClosingManager.sol 2ms
contracts/interfaces/IConditionalOpeningManager.sol 2ms
contracts/interfaces/IDexAdapter.sol 5ms
contracts/interfaces/IEPMXToken.sol 3ms
contracts/interfaces/IERC20Mock.sol 3ms
contracts/interfaces/IInterestRateStrategy.sol 2ms
contracts/interfaces/ILimitPriceCOM.sol 5ms
contracts/interfaces/IPMXPriceFeed.sol 2ms
contracts/interfaces/IPausable.sol 2ms
contracts/interfaces/IPriceFeedUpdaterTestService.sol 2ms
contracts/interfaces/IPrimexLens.sol 11ms
contracts/interfaces/IPrimexPricingLibraryMock.sol 3ms
contracts/interfaces/IPrimexRegistry.sol 2ms
contracts/interfaces/IPrimexUpkeep.sol 10ms
contracts/interfaces/IRedeemer.sol 3ms
contracts/interfaces/ISwapManager.sol 6ms
contracts/interfaces/ITakeProfitStopLossCCM.sol 3ms
contracts/interfaces/ITrailingStopCCM.sol 2ms
contracts/interfaces/IWhitelist.sol 3ms
contracts/lens/BestDexLens.sol 243ms
contracts/lens/PrimexLens.sol 204ms
contracts/libraries/Errors.sol 15ms
contracts/libraries/LimitOrderLibrary.sol 141ms
contracts/libraries/PositionLibrary.sol 244ms
contracts/libraries/PrimexPricingLibrary.sol 231ms
contracts/libraries/TokenTransfersLibrary.sol 17ms
contracts/mocks/AttackerBucket.sol 3ms
contracts/mocks/BucketMock.sol 24ms
contracts/mocks/ERC20Mock.sol 14ms
contracts/mocks/MaliciousDexMock.sol 4ms
contracts/mocks/MockNearestSearch.sol 7ms
```

```
contracts/mocks/PrimexPricingLibraryMock.sol 9ms
contracts/mocks/TokenTransfersLibraryMock.sol 2ms
contracts/TestnetServices/interfaces/IBalancerBotLens.sol 2ms
contracts/TestnetServices/interfaces/ICurveBotLens.sol 2ms
contracts/TestnetServices/interfaces/IPrimexAggregatorV3TestService.sol 1ms
contracts/TestnetServices/interfaces/ISynchronizationBotLens.sol 3ms
contracts/TestnetServices/interfaces/ISynchronizationBotLensQuickswapTestService.sol 2ms
contracts/TestnetServices/interfaces/ISynchronizationBotLensUniV3TestService.sol 2ms
contracts/WhiteBlackList/WhiteBlackList/IWhiteBlackList.sol 3ms
contracts/WhiteBlackList/WhiteBlackList/WhiteBlackList.sol 7ms
contracts/WhiteBlackList/WhiteBlackListBase/WhiteBlackListBase.sol 31ms
contracts/WhiteBlackList/WhiteBlackListReferral/IWhiteBlackListReferral.sol 2ms
contracts/WhiteBlackList/WhiteBlackListReferral/WhiteBlackListReferral.sol 13ms
contracts/WhiteBlackList/WhiteBlackListReferral/WhiteBlackListReferralStorage.sol 1ms
contracts/interfaces/routers/ICurveCalc.sol 2ms
contracts/interfaces/routers/ICurvePool.sol 2ms
contracts/interfaces/routers/ICurveRegistry.sol 2ms
contracts/interfaces/routers/ICurveRouter.sol 3ms
contracts/libraries/utils/BytesLib.sol 19ms
contracts/libraries/utils/V3Path.sol 11ms
contracts/libraries/utils/WadRayMath.sol 39ms
contracts/mocks/mocksInterfaces/IAttackerBucket.sol 2ms
contracts/mocks/mocksInterfaces/IBucketMock.sol 2ms
contracts/mocks/mocksInterfaces/IMaliciousDexMock.sol 2ms
contracts/mocks/mocksInterfaces/IMockNearestSearch.sol 2ms
contracts/mocks/upgradeMocks/BucketV2.sol 2ms
contracts/mocks/upgradeMocks/DebtTokenV2.sol 2ms
contracts/mocks/upgradeMocks/FeeDecreaserV2.sol 1ms
contracts/mocks/upgradeMocks/InterestIncreaserV2.sol 1ms
contracts/mocks/upgradeMocks/KeeperRewardDistributorV2.sol 2ms
contracts/mocks/upgradeMocks/LimitOrderManagerV2.sol 2ms
contracts/mocks/upgradeMocks/LiquidityMiningRewardDistributorV2.sol 2ms
contracts/mocks/upgradeMocks/PMXBonusNFTV2.sol 2ms
contracts/mocks/upgradeMocks/PTokenV2.sol 2ms
contracts/mocks/upgradeMocks/PositionManagerV2.sol 1ms
contracts/mocks/upgradeMocks/PriceOracleV2.sol 2ms
contracts/mocks/upgradeMocks/PrimexDNSV2.sol 1ms
contracts/mocks/upgradeMocks/ReferralProgramV2.sol 1ms
contracts/mocks/upgradeMocks/ReserveV2.sol 1ms
contracts/mocks/upgradeMocks/SpotTradingRewardDistributorV2.sol 2ms
contracts/mocks/upgradeMocks/TraderBalanceVaultV2.sol 1ms
contracts/mocks/upgradeMocks/TreasuryV2.sol 1ms
contracts/mocks/upgradeMocks/WhiteBlackListReferralV2.sol 2ms
contracts/mocks/upgradeMocksInterfaces/IBucketV2.sol 2ms
contracts/mocks/upgradeMocksInterfaces/IDebtTokenV2.sol 1ms
contracts/mocks/upgradeMocksInterfaces/IFeeDecreaserV2.sol 1ms
contracts/mocks/upgradeMocksInterfaces/IInterestIncreaserV2.sol 2ms
contracts/mocks/upgradeMocksInterfaces/IKeeperRewardDistributorV2.sol 1ms
contracts/mocks/upgradeMocksInterfaces/ILimitOrderManagerV2.sol 1ms
contracts/mocks/upgradeMocksInterfaces/ILiquidityMiningRewardDistributorV2.sol 1ms
contracts/mocks/upgradeMocksInterfaces/IPMXBonusNFTV2.sol 1ms
contracts/mocks/upgradeMocksInterfaces/IPTokenV2.sol 2ms
contracts/mocks/upgradeMocksInterfaces/IPositionManagerV2.sol 1ms
contracts/mocks/upgradeMocksInterfaces/IPriceOracleV2.sol 1ms
contracts/mocks/upgradeMocksInterfaces/IPrimexDNSV2.sol 2ms
contracts/mocks/upgradeMocksInterfaces/IReferralProgramV2.sol 1ms
contracts/mocks/upgradeMocksInterfaces/IReserveV2.sol 1ms
contracts/mocks/upgradeMocksInterfaces/ISpotTradingRewardDistributorV2.sol 1ms
contracts/mocks/upgradeMocksInterfaces/ITraderBalanceVaultV2.sol 2ms
contracts/mocks/upgradeMocksInterfaces/ITreasuryV2.sol 1ms
contracts/mocks/upgradeMocksInterfaces/IWhiteBlackListReferralV2.sol 1ms
$ ./node_modules/.bin/prettier --write deploy/**/*.js deploy/**/*.deploy.js test/**/*.js
test/**/*.test.js test/*.test.js tasks/*.js tasks/**/*.js tasks/**/*.deploy.js tasks/**/**/*.js
tasks/**/**/**/*.js config/*.js
deploy/TestnetServices/BalancerBotLens.deploy.js 61ms
deploy/TestnetServices/CurveBotLens.deploy.js 5ms
deploy/TestnetServices/PriceFeedUpdaterTestService.deploy.js 13ms
deploy/TestnetServices/SynchronizationBotLens.deploy.js 3ms
deploy/TestnetServices/SynchronizationBotLensQuickswap.deploy.js 5ms
deploy/TestnetServices/SynchronizationBotLensUniV3.deploy.js 6ms
deploy/core/ActivityRewardDistributor.deploy.js 9ms
deploy/core/BatchManager.deploy.js 5ms
```

```
deploy/core/BucketImplementation.deploy.js 4ms
deploy/core/DebtTokenImplementation.deploy.js 4ms
deploy/core/DexAdapter.deploy.js 16ms
deploy/core/EPMXToken.deploy.js 7ms
deploy/core/InterestRateStrategy.deploy.js 4ms
deploy/core/KeeperRewardDistributor.deploy.js 10ms
deploy/core/LimitOrderManager.deploy.js 6ms
deploy/core/LiquidityMiningRewardDistributor.deploy.js 7ms
deploy/core/PMXToken.deploy.js 5ms
deploy/core/PMXTokenOracle.deploy.js 7ms
deploy/core/PTokenImplementation.deploy.js 3ms
deploy/core/PositionManager.deploy.js 10ms
deploy/core/PriceOracle.deploy.js 7ms
deploy/core/PrimexDNS.deploy.js 7ms
deploy/core/PrimexProxyAdmin.deploy.js 3ms
deploy/core/PrimexTimelocks.deploy.js 14ms
deploy/core/PrimexUpkeep.deploy.js 7ms
deploy/core/Redeemer.deploy.js 5ms
deploy/core/Registry.deploy.js 5ms
deploy/core/Reserve.deploy.js 4ms
deploy/core/SpotTradingRewardDistributor.deploy.js 4ms
deploy/core/SwapManager.deploy.js 8ms
deploy/core/TraderBalanceVault.deploy.js 5ms
deploy/core/Treasury.deploy.js 4ms
deploy/core/WhiteBlackList.deploy.js 4ms
deploy/nft/FeeDecreaser.deploy.js 5ms
deploy/nft/InterestIncreaser.deploy.js 4ms
deploy/nft/PMXBonusNFT.deploy.js 2ms
deploy/referralProgram/ReferralProgram.deploy.js 3ms
deploy/referralProgram/WhiteBlackList.deploy.js 5ms
deploy/test/Bucket.deploy.js 8ms
deploy/test/Dexes.deploy.js 5ms
deploy/test/EPMXTokenAccess.js 4ms
deploy/test/PrimexAggregatorV3TestService.deploy.js 3ms
deploy/test/TestTokens.deploy.js 4ms
test/integration/InterestRateStrategy.test.js 74ms
test/integration/KeeperRewardDistributor.test.js 81ms
test/integration/LenderRewardDistributor.test.js 42ms
test/integration/LimitPriceCOM.test.js 43ms
test/integration/LiquidityMiningRewardDistributor.test.js 34ms
test/integration/OperationWithBalancer.test.js 52ms
test/integration/OperationWithCurve.test.js 32ms
test/integration/PMXBonusNFT.test.js 60ms
test/integration/PriceOracle.test.js 9ms
test/integration/PrimexPricingLibrary.test.js 75ms
test/integration/PrimexTimelocks.test.js 14ms
test/integration/Redeemer.test.js 9ms
test/integration/Reserve.test.js 29ms
test/integration/SpotTradingRewardDistributor.test.js 25ms
test/integration/SwapManager.test.js 30ms
test/integration/TakeProfitStopLossCCM.test.js 21ms
test/integration/TraderBalanceVault.test.js 11ms
test/integration/TraderRewardDistributor.test.js 30ms
test/integration/TrailingStopCCM.test.js 25ms
test/integration/Treasury.test.js 21ms
test/integration/Upgradability.test.js 12ms
test/integration/WhiteBlackList.test.js 10ms
test/unit/ActivityRewardDistributor.test.js 85ms
test/unit/BonusExecutor.test.js 152ms
test/unit/EPMXToken.test.js 16ms
test/unit/InterestRateStrategy.test.js 7ms
test/unit/KeeperRewardDistributor.test.js 22ms
test/unit/LimitPriceCOM.test.js 9ms
test/unit/LiquidityMiningRewardDistributor.test.js 90ms
test/unit/PMXBonusNFT.test.js 31ms
test/unit/PMXTokenOracle.test.js 8ms
test/unit/PositionManager.test.js 8ms
test/unit/PriceOracle.test.js 35ms
test/unit/PrimexPricingLibrary.test.js 18ms
test/unit/PrimexProxyAdmin.test.js 6ms
test/unit/PrimexRegistry.test.js 5ms
test/unit/PrimexTimelocks.test.js 9ms
```

```
test/unit/Redeemer.test.js 8ms
test/unit/ReferralProgram.test.js 8ms
test/unit/Reserve.test.js 14ms
test/unit/SpotTradingRewardDistributor.test.js 38ms
test/unit/SwapManager.test.js 14ms
test/unit/TakeProfitStopLossCCM.test.js 7ms
test/unit/TraderBalanceVault.test.js 18ms
test/unit/TrailingStopCCM.test.js 7ms
test/unit/Treasury.test.js 18ms
test/unit/WhiteBlackList.test.js 23ms
test/utils/activityRewardDistributorMath.js 4ms
test/utils/addressFromEvent.js 2ms
test/utils/bnMath.js 6ms
test/utils/conditionParams.js 4ms
test/utils/constants.js 4ms
test/utils/defaultBarCalcParams.js 2ms
test/utils/dexOperations.js 51ms
test/utils/encodePriceSqrt.js 6ms
test/utils/eventValidation.js 9ms
test/utils/generateSignature.js 6ms
test/utils/hardhatUtils.js 5ms
test/utils/makeWalletFromPrivateKey.js 2ms
test/utils/math.js 12ms
test/utils/setBadOraclePrice.js 3ms
test/utils/waffleMocks.js 13ms
test/Balancer.test.js 20ms
test/BestDexLens.test.js 95ms
test/Bucket.test.js 186ms
test/BucketsFactory.test.js 14ms
test/ChainLinkUpKeep.test.js 7ms
test/CrossDexClosePosition.test.js 44ms
test/CurveBotLens.test.js 11ms
test/DebtToken.test.js 15ms
test/DepositAsset_isAssetNotFromSwapPair.test.js 194ms
test/DepositAsset_isPositionAsset.test.js 151ms
test/DexAdapter.test.js 114ms
test/DexAdapterSwapPath.test.js 40ms
test/LimitOrderManager.test.js 353ms
test/LimitedMintTokens.test.js 8ms
test/PMXToken.test.js 4ms
test/PToken.test.js 403ms
test/PhaseSwitching.test.js 83ms
test/PositionBatchClose.test.js 120ms
test/PositionManager.test.js 256ms
test/PriceFeedUpdaterTestService.test.js 20ms
test/PriceFeedsAccessControl.test.js 4ms
test/PrimexDNS.test.js 29ms
test/PrimexLens.test.js 102ms
test/PrimexUpkeep.test.js 204ms
test/ReferralProgram.test.js 12ms
test/SpotTrading.test.js 182ms
tasks/index.js 2ms
tasks/support.js 7ms
tasks/ChainLink/CounterUpKeep.deploy.js 2ms
tasks/ChainLink/KeeperRegistry.deploy.js 3ms
tasks/ChainLink/index.js 3ms
tasks/ChainLink/registerUpkeepAndAddFunds.js 3ms
tasks/ChainLink/setKeepers.js 4ms
tasks/FrontendSetup/PriceFeedsSetup.js 8ms
tasks/FrontendSetup/Pricefeeds.deploy.js 5ms
tasks/FrontendSetup/addBuckets.js 8ms
tasks/FrontendSetup/addLiquidityAB.js 11ms
tasks/FrontendSetup/addLiquidityABBalancer.js 16ms
tasks/FrontendSetup/addLiquidityABCurve.js 16ms
tasks/FrontendSetup/addLiquidityABMeshswap.js 6ms
tasks/FrontendSetup/addLiquidityABQuickswapV3.js 7ms
tasks/FrontendSetup/addLiquidityABUniswapV3.js 8ms
tasks/FrontendSetup/chainLinkSetup.js 5ms
tasks/FrontendSetup/createLimitNetworkTestnets.js 14ms
tasks/FrontendSetup/index.js 6ms
tasks/FrontendSetup/setupEarlyRewardsInBuckets.js 5ms
tasks/FrontendSetup/setupPairsConfig.js 7ms
```

```
tasks/FrontendSetup/setupSpotTradingRewards.js 3ms
tasks/FrontendSetup/tokenERC20AddLimit.js 2ms
tasks/FrontendSetup/tokenERC20Mint.js 12ms
tasks/TestnetServices/index.js 2ms
tasks/closablePositions/closablePositionByLiquidationPrice.js 6ms
tasks/closablePositions/closablePositionBySL.js 4ms
tasks/closablePositions/closablePositionByTP.js 5ms
tasks/closablePositions/closePosition.js 3ms
tasks/closablePositions/index.js 2ms
tasks/closablePositions/taskSetup.js 12ms
tasks/core/index.js 2ms
tasks/deployScripts/deployCore.js 3ms
tasks/deployScripts/deployCoreAndTestnetServices.js 4ms
tasks/deployScripts/deployFullDevnode1.js 3ms
tasks/deployScripts/deployObscuro.js 2ms
tasks/deployScripts/index.js 2ms
tasks/dexes/index.js 1ms
tasks/nft/index.js 1ms
tasks/phaseSwitching/index.js 4ms
tasks/referral/index.js 1ms
tasks/utils/accountAddresses.js 3ms
tasks/utils/accounts.js 4ms
tasks/utils/addRole.js 3ms
tasks/utils/configureObscuroWalletExtension.js 3ms
tasks/utils/customErrorDecode.js 2ms
tasks/utils/decodeFunctionData.js 2ms
tasks/utils/encodeFunctionData.js 2ms
tasks/utils/index.js 3ms
tasks/utils/miner.js 2ms
tasks/utils/syncContractDataWithEthernal.js 4ms
tasks/utils/upgradeContract.js 3ms
tasks/TestnetServices/BalancerBotLens/BalancerBotLens.deploy.js 2ms
tasks/TestnetServices/BalancerBotLens/index.js 2ms
tasks/TestnetServices/CurveBotLens/CurveBotLens.deploy.js 2ms
tasks/TestnetServices/CurveBotLens/index.js 2ms
tasks/TestnetServices/ERC20Mock/erc20Mock.deploy.js 3ms
tasks/TestnetServices/ERC20Mock/index.js 2ms
tasks/TestnetServices/PriceFeedUpdaterTestService/PriceFeedUpdaterTestService.deploy.js 4ms
tasks/TestnetServices/PriceFeedUpdaterTestService/index.js 2ms
tasks/TestnetServices/PrimexAggregatorV3TestService/index.js 2ms
tasks/TestnetServices/PrimexAggregatorV3TestService/primexAggregatorV3TestService.deploy.js 3ms
tasks/TestnetServices/SynchronizationBotLens/SynchronizationBotLens.deploy.js 3ms
tasks/TestnetServices/SynchronizationBotLens/index.js 2ms
tasks/TestnetServices/SynchronizationBotLensQuickswap/SynchronizationBotLensQuickswapTestService.deploy.j
s 2ms
tasks/TestnetServices/SynchronizationBotLensQuickswap/index.js 3ms
tasks/TestnetServices/SynchronizationBotLensUniV3/SynchronizationBotLensUniV3TestService.deploy.js 2ms
tasks/TestnetServices/SynchronizationBotLensUniV3/index.js 1ms
tasks/core/ActivityRewardDistributor/ActivityRewardDistributor.deploy.js 5ms
tasks/core/ActivityRewardDistributor/index.js 2ms
tasks/core/ActivityRewardDistributor/setupBucket.js 3ms
tasks/core/BatchManager/batchManager.deploy.js 5ms
tasks/core/BatchManager/index.js 2ms
tasks/core/Bucket/bucket.deploy.js 12ms
tasks/core/Bucket/index.js 3ms
tasks/core/DexAdapter/dexAdapter.deploy.js 6ms
tasks/core/DexAdapter/dexAdapter.setDexType.js 2ms
tasks/core/DexAdapter/index.js 3ms
tasks/core/EPMXToken/AddPrimexAddressesToWhitelist.js 2ms
tasks/core/EPMXToken/EPMXToken.deploy.js 4ms
tasks/core/EPMXToken/index.js 1ms
tasks/core/InterestRateStrategy/InterestRateStrategy.deploy.js 2ms
tasks/core/InterestRateStrategy/index.js 2ms
tasks/core/KeeperRewardDistributor/KeeperRewardDistributor.deploy.js 4ms
tasks/core/KeeperRewardDistributor/index.js 2ms
tasks/core/LimitOrderManager/index.js 2ms
tasks/core/LimitOrderManager/limitOrderManager.deploy.js 4ms
tasks/core/LiquidityMiningRewardDistributor/LiquidityMiningRewardDistributor.deploy.js 6ms
tasks/core/LiquidityMiningRewardDistributor/index.js 2ms
tasks/core/PMXPriceFeed/PMXPriceFeed.deploy.js 2ms
tasks/core/PMXPriceFeed/index.js 1ms
tasks/core/PMXToken/PMXToken.deploy.js 1ms
```

```
tasks/core/PMXToken/index.js 1ms
tasks/core/PositionManager/index.js 2ms
tasks/core/PositionManager/positionManager.deploy.js 4ms
tasks/core/PriceOracle/index.js 2ms
tasks/core/PriceOracle/priceOracle.deploy.js 2ms
tasks/core/PriceOracle/rinkebyChainlinkPriceFeeds.js 7ms
tasks/core/PriceOracle/updateFeedsFromConfig.js 4ms
tasks/core/PriceOracle/updatePriceFeed.js 3ms
tasks/core/PriceOracle/updateVolatilityFeed.js 3ms
tasks/core/PrimexDNS/PrimexDNS.addDEX.js 3ms
tasks/core/PrimexDNS/PrimexDNS.deploy.js 3ms
tasks/core/PrimexDNS/index.js 2ms
tasks/core/PrimexDNS/setAavePoolAddress.js 1ms
tasks/core/PrimexProxyAdmin/PrimexProxyAdmin.deploy.js 2ms
tasks/core/PrimexProxyAdmin/index.js 1ms
tasks/core/PrimexTimelock/PrimexTimelock.deploy.js 3ms
tasks/core/PrimexTimelock/index.js 2ms
tasks/core/PrimexUpkeep/PrimexUpkeep.deploy.js 3ms
tasks/core/PrimexUpkeep/index.js 2ms
tasks/core/ProtocolFeeCollector/ProtocolFeeCollector.deploy.js 2ms
tasks/core/ProtocolFeeCollector/index.js 1ms
tasks/core/Redeemer/Redeemer.deploy.js 3ms
tasks/core/Redeemer/index.js 2ms
tasks/core/Registry/index.js 2ms
tasks/core/Registry/registry.deploy.js 2ms
tasks/core/Reserve/Reserve.deploy.js 3ms
tasks/core/Reserve/index.js 2ms
tasks/core/Reserve/setTransferRestrictionsByConfig.js 3ms
tasks/core/SpotTradingRewardDistributor/SpotTradingRewardDistributor.deploy.js 3ms
tasks/core/SpotTradingRewardDistributor/index.js 2ms
tasks/core/SwapManager/index.js 2ms
tasks/core/SwapManager/swapManager.deploy.js 6ms
tasks/core/TraderBalanceVault/index.js 1ms
tasks/core/TraderBalanceVault/traderBalanceVault.deploy.js 2ms
tasks/core/Treasury/Treasury.deploy.js 3ms
tasks/core/Treasury/index.js 3ms
tasks/core/Treasury/setTreasurySpendersByConfig.js 6ms
tasks/core/WhiteBlackList/index.js 2ms
tasks/core/WhiteBlackList/whiteBlackList.deploy.js 3ms
tasks/core/conditionalManagers/index.js 1ms
tasks/core/factories/index.js 1ms
tasks/core/lens/index.js 1ms
tasks/core/libraries/index.js 2ms
tasks/deployScripts/deployEnvironment/deployEnv.js 3ms
tasks/deployScripts/deployEnvironment/deployEnvArbitrum.js 2ms
tasks/deployScripts/deployEnvironment/deployEnvDevNode1.js 2ms
tasks/deployScripts/deployEnvironment/deployEnvDevnode2.js 2ms
tasks/deployScripts/deployEnvironment/deployEnvDevnode3.js 2ms
tasks/deployScripts/deployEnvironment/deployEnvGoerli.js 2ms
tasks/deployScripts/deployEnvironment/deployEnvMoonbaseAlpha.js 2ms
tasks/deployScripts/deployEnvironment/deployEnvMumbai.js 1ms
tasks/deployScripts/deployEnvironment/deployEnvObscuro.js 2ms
tasks/deployScripts/deployEnvironment/deployEnvPolygonZKtestnet.js 2ms
tasks/deployScripts/deployEnvironment/deployEnvZkSync2.js 2ms
tasks/deployScripts/deployEnvironment/index.js 2ms
tasks/deployScripts/phaseSwitching/index.js 2ms
tasks/deployScripts/phaseSwitching/phase-1-initial-deploy.js 2ms
tasks/deployScripts/phaseSwitching/phase-2-execution-spot-trading-rewards.js 2ms
tasks/deployScripts/phaseSwitching/phase-2-proposal-spot-trading-rewards.js 2ms
tasks/deployScripts/phaseSwitching/phase-3-execution-early-lenders-rewards.js 2ms
tasks/deployScripts/phaseSwitching/phase-3-proposal-early-lenders-rewards.js 1ms
tasks/deployScripts/phaseSwitching/phase-4-execution-early-traders-rewards.js 2ms
tasks/deployScripts/phaseSwitching/phase-4-proposal-early-traders-rewards.js 2ms
tasks/deployScripts/phaseSwitching/phase-5-execution-enable-nft.js 1ms
tasks/deployScripts/phaseSwitching/phase-5-proposal-enable-nft.js 2ms
tasks/deployScripts/phaseSwitching/phase-6-execution-migrate-epmx-to-pmx.js 2ms
tasks/deployScripts/phaseSwitching/phase-6-proposal-migrate-epmx-to-pmx.js 2ms
tasks/dexes/Aave/Aave.deploy.js 2ms
tasks/dexes/Aave/addLiquidity.js 3ms
tasks/dexes/Aave/index.js 2ms
tasks/dexes/Balancer/Balancer.deploy.js 3ms
tasks/dexes/Balancer/CreateWeightedPool.js 5ms
```

```
tasks/dexes/Balancer/addLiquidity.js 4ms
tasks/dexes/Balancer/index.js 4ms
tasks/dexes/Balancer/utils.js 2ms
tasks/dexes/Curve/Curve.deploy.js 7ms
tasks/dexes/Curve/addLiquidity.js 6ms
tasks/dexes/Curve/createPool.js 13ms
tasks/dexes/Curve/index.js 3ms
tasks/dexes/Curve/swapExactTokensForTokens.js 5ms
tasks/dexes/Meshswap/Meshswap.deploy.js 5ms
tasks/dexes/Meshswap/createPoolAndAddLiquidity.js 4ms
tasks/dexes/Meshswap/index.js 4ms
tasks/dexes/QuickswapV3/CreatePool.js 4ms
tasks/dexes/QuickswapV3/QuickswapV3.deploy.js 4ms
tasks/dexes/QuickswapV3/SwapExactInputSingle.js 4ms
tasks/dexes/QuickswapV3/addLiquidity.js 5ms
tasks/dexes/QuickswapV3/index.js 3ms
tasks/dexes/QuickswapV3/utils.js 1ms
tasks/dexes/Router/addLiquidity.js 6ms
tasks/dexes/Router/addLiquidityETH.js 6ms
tasks/dexes/Router/index.js 7ms
tasks/dexes/Router/removeLiquidity.js 4ms
tasks/dexes/Router/removeLiquidityETH.js 4ms
tasks/dexes/Router/swapExactETHForTokens.js 3ms
tasks/dexes/Router/swapExactTokensForETH.js 4ms
tasks/dexes/Router/swapExactTokensForTokens.js 3ms
tasks/dexes/UniswapV2/UniswapV2.deploy.js 3ms
tasks/dexes/UniswapV2/index.js 2ms
tasks/dexes/UniswapV3/CreatePool.js 4ms
tasks/dexes/UniswapV3/SwapExactInputSingle.js 4ms
tasks/dexes/UniswapV3/UniswapMulticall.deploy.js 3ms
tasks/dexes/UniswapV3/UniswapV3.deploy.js 6ms
tasks/dexes/UniswapV3/addLiquidity.js 6ms
tasks/dexes/UniswapV3/index.js 3ms
tasks/nft/FeeDecreaser/FeeDecreaser.deploy.js 4ms
tasks/nft/FeeDecreaser/index.js 2ms
tasks/nft/InterestIncreaser/InterestIncreaser.deploy.js 3ms
tasks/nft/InterestIncreaser/index.js 2ms
tasks/nft/PMXBonusNFT/PMXBonusNFT.deploy.js 3ms
tasks/nft/PMXBonusNFT/index.js 1ms
tasks/phaseSwitching/phase-2/getPhaseArguments.js 2ms
tasks/phaseSwitching/phase-2/setSpotTradingRewardDistributor.js 3ms
tasks/phaseSwitching/phase-2/setupSpotTradingRewardDistributorInPM.js 2ms
tasks/phaseSwitching/phase-2/topUpSpotTradingRewardDistributor.js 2ms
tasks/phaseSwitching/phase-3-4/getPhaseArguments.js 4ms
tasks/phaseSwitching/phase-3-4/setupBucketsWithRewardsConfiguration.js 2ms
tasks/phaseSwitching/phase-3-4/setupEarlyRewardsInBuckets.js 3ms
tasks/phaseSwitching/phase-3-4/topUpActivityRewardDistributor.js 2ms
tasks/phaseSwitching/phase-5/activateBonusNFTs.js 2ms
tasks/phaseSwitching/phase-5/getPhaseArguments.js 3ms
tasks/phaseSwitching/phase-6/deployRewardDistributors.js 4ms
tasks/phaseSwitching/phase-6/getPhaseArguments.js 6ms
tasks/phaseSwitching/phase-6/migrateEpmxToPmx.js 2ms
tasks/phaseSwitching/phase-6/updateRewardConfigurationsInBuckets.js 5ms
tasks/phaseSwitching/phase-6/updateRewardDistributors.js 3ms
tasks/referral/ReferralProgram/index.js 2ms
tasks/referral/ReferralProgram/referralProgram.deploy.js 2ms
tasks/referral/WhiteBlackList/index.js 2ms
tasks/referral/WhiteBlackList/whiteBlackListReferral.deploy.js 2ms
tasks/core/conditionalManagers/LimitPriceCOM/LimitPriceCOM.deploy.js 3ms
tasks/core/conditionalManagers/LimitPriceCOM/index.js 1ms
tasks/core/conditionalManagers/TakeProfitStopLossCCM/TakeProfitStopLossCCM.deploy.js 2ms
tasks/core/conditionalManagers/TakeProfitStopLossCCM/index.js 2ms
tasks/core/conditionalManagers/TrailingStopCCM/TrailingStopCCM.deploy.js 2ms
tasks/core/conditionalManagers/TrailingStopCCM/index.js 2ms
tasks/core/factories/BucketsFactory/BucketImplementation.deploy.js 3ms
tasks/core/factories/BucketsFactory/BucketsFactory.deploy.js 2ms
tasks/core/factories/BucketsFactory/index.js 1ms
tasks/core/factories/DebtTokensFactory/DebtTokenImplementation.deploy.js 2ms
tasks/core/factories/DebtTokensFactory/DebtTokensFactory.deploy.js 5ms
tasks/core/factories/DebtTokensFactory/index.js 2ms
tasks/core/factories/PTokensFactory/PTokenImplementation.deploy.js 3ms
tasks/core/factories/PTokensFactory/PTokensFactory.deploy.js 3ms
```

```
tasks/core/factories/PTokensFactory/index.js 1ms
tasks/core/lens/BestDexLens/bestDexLens.deploy.js 2ms
tasks/core/lens/BestDexLens/index.js 1ms
tasks/core/lens/PrimexLens/index.js 2ms
tasks/core/lens/PrimexLens/primexLens.deploy.js 2ms
tasks/core/libraries/Errors/Errors.deploy.js 2ms
tasks/core/libraries/Errors/index.js 2ms
tasks/core/libraries/LimitOrderLibrary/LimitOrderLibrary.deploy.js 3ms
tasks/core/libraries/LimitOrderLibrary/index.js 3ms
tasks/core/libraries/PositionLibrary/PositionLibrary.deploy.js 3ms
tasks/core/libraries/PositionLibrary/index.js 2ms
tasks/core/libraries/PrimexPricingLibrary/PrimexPricingLibrary.deploy.js 2ms
tasks/core/libraries/PrimexPricingLibrary/index.js 2ms
tasks/core/libraries/TokenTransfersLibrary/TokenTransfersLibrary.deploy.js 2ms
tasks/core/libraries/TokenTransfersLibrary/index.js 2ms
config/configUtils.js 11ms
$ yarn eslint --fix deploy/**/*.js deploy/**/*.deploy.js test/**/*.js test/**/*.test.js test/*.test.js
tasks/*.js tasks/**/*.js tasks/**/*.deploy.js tasks/**/**/*.js tasks/**/**/**/*.js config/*.js
$ /Users/poanlin/Downloads/primex_finance-primex_contracts-276d36f7002-
github/src/node_modules/.bin/eslint --fix deploy/TestnetServices/BalancerBotLens.deploy.js
deploy/TestnetServices/CurveBotLens.deploy.js
deploy/TestnetServices/PriceFeedUpdaterTestService.deploy.js
deploy/TestnetServices/SynchronizationBotLens.deploy.js
deploy/TestnetServices/SynchronizationBotLensQuickswap.deploy.js
deploy/TestnetServices/SynchronizationBotLensUniV3.deploy.js
deploy/core/ActivityRewardDistributor.deploy.js deploy/core/BatchManager.deploy.js
deploy/core/BucketImplementation.deploy.js deploy/core/DebtTokenImplementation.deploy.js
deploy/core/DexAdapter.deploy.js deploy/core/EPMXToken.deploy.js
deploy/core/InterestRateStrategy.deploy.js deploy/core/KeeperRewardDistributor.deploy.js
deploy/core/LimitOrderManager.deploy.js deploy/core/LiquidityMiningRewardDistributor.deploy.js
deploy/core/PMXToken.deploy.js deploy/core/PMXTokenOracle.deploy.js
deploy/core/PTokenImplementation.deploy.js deploy/core/PositionManager.deploy.js
deploy/core/PriceOracle.deploy.js deploy/core/PrimexDNS.deploy.js deploy/core/PrimexProxyAdmin.deploy.js
deploy/core/PrimexTimelocks.deploy.js deploy/core/PrimexUpkeep.deploy.js deploy/core/Redeemer.deploy.js
deploy/core/Registry.deploy.js deploy/core/Reserve.deploy.js
deploy/core/SpotTradingRewardDistributor.deploy.js deploy/core/SwapManager.deploy.js
deploy/core/TraderBalanceVault.deploy.js deploy/core/Treasury.deploy.js
deploy/core/WhiteBlackList.deploy.js deploy/nft/FeeDecreaser.deploy.js
deploy/nft/InterestIncreaser.deploy.js deploy/nft/PMXBonusNFT.deploy.js
deploy/referralProgram/ReferralProgram.deploy.js deploy/referralProgram/WhiteBlackList.deploy.js
deploy/test/Bucket.deploy.js deploy/test/Dexes.deploy.js deploy/test/EPMXTokenAccess.js
deploy/test/PrimexAggregatorV3TestService.deploy.js deploy/test/TestTokens.deploy.js
deploy/TestnetServices/BalancerBotLens.deploy.js deploy/TestnetServices/CurveBotLens.deploy.js
deploy/TestnetServices/PriceFeedUpdaterTestService.deploy.js
deploy/TestnetServices/SynchronizationBotLens.deploy.js
deploy/TestnetServices/SynchronizationBotLensQuickswap.deploy.js
deploy/TestnetServices/SynchronizationBotLensUniV3.deploy.js
deploy/core/ActivityRewardDistributor.deploy.js deploy/core/BatchManager.deploy.js
deploy/core/BucketImplementation.deploy.js deploy/core/DebtTokenImplementation.deploy.js
deploy/core/DexAdapter.deploy.js deploy/core/EPMXToken.deploy.js
deploy/core/InterestRateStrategy.deploy.js deploy/core/KeeperRewardDistributor.deploy.js
deploy/core/LimitOrderManager.deploy.js deploy/core/LiquidityMiningRewardDistributor.deploy.js
deploy/core/PMXToken.deploy.js deploy/core/PMXTokenOracle.deploy.js
deploy/core/PTokenImplementation.deploy.js deploy/core/PositionManager.deploy.js
deploy/core/PriceOracle.deploy.js deploy/core/PrimexDNS.deploy.js deploy/core/PrimexProxyAdmin.deploy.js
deploy/core/PrimexTimelocks.deploy.js deploy/core/PrimexUpkeep.deploy.js deploy/core/Redeemer.deploy.js
deploy/core/Registry.deploy.js deploy/core/Reserve.deploy.js
deploy/core/SpotTradingRewardDistributor.deploy.js deploy/core/SwapManager.deploy.js
deploy/core/TraderBalanceVault.deploy.js deploy/core/Treasury.deploy.js
deploy/core/WhiteBlackList.deploy.js deploy/nft/FeeDecreaser.deploy.js
deploy/nft/InterestIncreaser.deploy.js deploy/nft/PMXBonusNFT.deploy.js
deploy/referralProgram/ReferralProgram.deploy.js deploy/referralProgram/WhiteBlackList.deploy.js
deploy/test/Bucket.deploy.js deploy/test/Dexes.deploy.js
deploy/test/PrimexAggregatorV3TestService.deploy.js deploy/test/TestTokens.deploy.js
test/integration/InterestRateStrategy.test.js test/integration/KeeperRewardDistributor.test.js
test/integration/LenderRewardDistributor.test.js test/integration/LimitPriceCOM.test.js
test/integration/LiquidityMiningRewardDistributor.test.js test/integration/OperationWithBalancer.test.js
test/integration/OperationWithCurve.test.js test/integration/PMXBonusNFT.test.js
test/integration/PriceOracle.test.js test/integration/PrimexPricingLibrary.test.js
test/integration/PrimexTimelocks.test.js test/integration/Redeemer.test.js
test/integration/Reserve.test.js test/integration/SpotTradingRewardDistributor.test.js
test/integration/SwapManager.test.js test/integration/TakeProfitStopLossCCM.test.js
```

```
test/integration/TraderBalanceVault.test.js test/integration/TraderRewardDistributor.test.js
test/integration/TrailingStopCCM.test.js test/integration/Treasury.test.js
test/integration/Upgradability.test.js test/integration/WhiteBlackList.test.js
test/unit/ActivityRewardDistributor.test.js test/unit/BonusExecutor.test.js test/unit/EPMXToken.test.js
test/unit/InterestRateStrategy.test.js test/unit/KeeperRewardDistributor.test.js
test/unit/LimitPriceCOM.test.js test/unit/LiquidityMiningRewardDistributor.test.js
test/unit/PMXBonusNFT.test.js test/unit/PMXTokenOracle.test.js test/unit/PositionManager.test.js
test/unit/PriceOracle.test.js test/unit/PrimexPricingLibrary.test.js test/unit/PrimexProxyAdmin.test.js
test/unit/PrimexRegistry.test.js test/unit/PrimexTimelocks.test.js test/unit/Redeemer.test.js
test/unit/ReferralProgram.test.js test/unit/Reserve.test.js
test/unit/SpotTradingRewardDistributor.test.js test/unit/SwapManager.test.js
test/unit/TakeProfitStopLossCCM.test.js test/unit/TraderBalanceVault.test.js
test/unit/TrailingStopCCM.test.js test/unit/Treasury.test.js test/unit/WhiteBlackList.test.js
test/utils/activityRewardDistributorMath.js test/utils/addressFromEvent.js test/utils/bnMath.js
test/utils/conditionParams.js test/utils/constants.js test/utils/defaultBarCalcParams.js
test/utils/dexOperations.js test/utils/encodePriceSqrt.js test/utils/eventValidation.js
test/utils/generateSignature.js test/utils/hardhatUtils.js test/utils/makeWalletFromPrivateKey.js
test/utils/math.js test/utils/setBadOraclePrice.js test/utils/waffleMocks.js
test/integration/InterestRateStrategy.test.js test/integration/KeeperRewardDistributor.test.js
test/integration/LenderRewardDistributor.test.js test/integration/LimitPriceCOM.test.js
test/integration/LiquidityMiningRewardDistributor.test.js test/integration/OperationWithBalancer.test.js
test/integration/OperationWithCurve.test.js test/integration/PMXBonusNFT.test.js
test/integration/PriceOracle.test.js test/integration/PrimexPricingLibrary.test.js
test/integration/PrimexTimelocks.test.js test/integration/Redeemer.test.js
test/integration/Reserve.test.js test/integration/SpotTradingRewardDistributor.test.js
test/integration/SwapManager.test.js test/integration/TakeProfitStopLossCCM.test.js
test/integration/TraderBalanceVault.test.js test/integration/TraderRewardDistributor.test.js
test/integration/TrailingStopCCM.test.js test/integration/Treasury.test.js
test/integration/Upgradability.test.js test/integration/WhiteBlackList.test.js
test/unit/ActivityRewardDistributor.test.js test/unit/BonusExecutor.test.js test/unit/EPMXToken.test.js
test/unit/InterestRateStrategy.test.js test/unit/KeeperRewardDistributor.test.js
test/unit/LimitPriceCOM.test.js test/unit/LiquidityMiningRewardDistributor.test.js
test/unit/PMXBonusNFT.test.js test/unit/PMXTokenOracle.test.js test/unit/PositionManager.test.js
test/unit/PriceOracle.test.js test/unit/PrimexPricingLibrary.test.js test/unit/PrimexProxyAdmin.test.js
test/unit/PrimexRegistry.test.js test/unit/PrimexTimelocks.test.js test/unit/Redeemer.test.js
test/unit/ReferralProgram.test.js test/unit/Reserve.test.js
test/unit/SpotTradingRewardDistributor.test.js test/unit/SwapManager.test.js
test/unit/TakeProfitStopLossCCM.test.js test/unit/TraderBalanceVault.test.js
test/unit/TrailingStopCCM.test.js test/unit/Treasury.test.js test/unit/WhiteBlackList.test.js
test/Balancer.test.js test/BestDexLens.test.js test/Bucket.test.js test/BucketsFactory.test.js
test/ChainLinkUpKeep.test.js test/CrossDexClosePosition.test.js test/CurveBotLens.test.js
test/DebtToken.test.js test/DepositAsset_isAssetNotFromSwapPair.test.js
test/DepositAsset_isPositionAsset.test.js test/DexAdapter.test.js test/DexAdapterSwapPath.test.js
test/LimitOrderManager.test.js test/LimitedMintTokens.test.js test/PMXToken.test.js test/PToken.test.js
test/PhaseSwitching.test.js test/PositionBatchClose.test.js test/PositionManager.test.js
test/PriceFeedUpdaterTestService.test.js test/PriceFeedsAccessControl.test.js test/PrimexDNS.test.js
test/PrimexLens.test.js test/PrimexUpkeep.test.js test/ReferralProgram.test.js test/SpotTrading.test.js
tasks/index.js tasks/support.js tasks/ChainLink/CounterUpKeep.deploy.js
tasks/ChainLink/KeeperRegistry.deploy.js tasks/ChainLink/index.js
tasks/ChainLink/registerUpkeepAndAddFunds.js tasks/ChainLink/setKeepers.js
tasks/FrontendSetup/PriceFeedsSetup.js tasks/FrontendSetup/Pricefeeds.deploy.js
tasks/FrontendSetup/addBuckets.js tasks/FrontendSetup/addLiquidityAB.js
tasks/FrontendSetup/addLiquidityABBalancer.js tasks/FrontendSetup/addLiquidityABCurve.js
tasks/FrontendSetup/addLiquidityABMeshswap.js tasks/FrontendSetup/addLiquidityABQuickswapV3.js
tasks/FrontendSetup/addLiquidityABUniswapV3.js tasks/FrontendSetup/chainLinkSetup.js
tasks/FrontendSetup/createLimitNetworkTestnets.js tasks/FrontendSetup/index.js
tasks/FrontendSetup/setupEarlyRewardsInBuckets.js tasks/FrontendSetup/setupPairsConfig.js
tasks/FrontendSetup/setupSpotTradingRewards.js tasks/FrontendSetup/tokenERC20AddLimit.js
tasks/FrontendSetup/tokenERC20Mint.js tasks/TestnetServices/index.js
tasks/closablePositions/closablePositionByLiquidationPrice.js
tasks/closablePositions/closablePositionBySL.js tasks/closablePositions/closablePositionByTP.js
tasks/closablePositions/closePosition.js tasks/closablePositions/index.js
tasks/closablePositions/taskSetup.js tasks/core/index.js tasks/deployScripts/deployCore.js
tasks/deployScripts/deployCoreAndTestnetServices.js tasks/deployScripts/deployFullDevnode1.js
tasks/deployScripts/deployObscuro.js tasks/deployScripts/index.js tasks/dexes/index.js tasks/nft/index.js
tasks/phaseSwitching/index.js tasks/referral/index.js tasks/utils/accountAddresses.js
tasks/utils/accounts.js tasks/utils/addRole.js tasks/utils/configureObscuroWalletExtension.js
tasks/utils/customErrorDecode.js tasks/utils/decodeFunctionData.js tasks/utils/encodeFunctionData.js
tasks/utils/index.js tasks/utils/miner.js tasks/utils/syncContractDataWithEthernal.js
tasks/utils/upgradeContract.js tasks/ChainLink/CounterUpKeep.deploy.js
tasks/ChainLink/KeeperRegistry.deploy.js tasks/FrontendSetup/Pricefeeds.deploy.js
tasks/TestnetServices/BalancerBotLens/BalancerBotLens.deploy.js
```

```
tasks/TestnetServices/BalancerBotLens/index.js tasks/TestnetServices/CurveBotLens/CurveBotLens.deploy.js
tasks/TestnetServices/CurveBotLens/index.js tasks/TestnetServices/ERC20Mock/erc20Mock.deploy.js
tasks/TestnetServices/ERC20Mock/index.js
tasks/TestnetServices/PriceFeedUpdaterTestService/PriceFeedUpdaterTestService.deploy.js
tasks/TestnetServices/PriceFeedUpdaterTestService/index.js
tasks/TestnetServices/PrimexAggregatorV3TestService/index.js
tasks/TestnetServices/PrimexAggregatorV3TestService/primexAggregatorV3TestService.deploy.js
tasks/TestnetServices/SynchronizationBotLens/SynchronizationBotLens.deploy.js
tasks/TestnetServices/SynchronizationBotLens/index.js
tasks/TestnetServices/SynchronizationBotLensQuickswap/SynchronizationBotLensQuickswapTestService.deploy.j
s tasks/TestnetServices/SynchronizationBotLensQuickswap/index.js
tasks/TestnetServices/SynchronizationBotLensUniV3/SynchronizationBotLensUniV3TestService.deploy.js
tasks/TestnetServices/SynchronizationBotLensUniV3/index.js
tasks/core/ActivityRewardDistributor/ActivityRewardDistributor.deploy.js
tasks/core/ActivityRewardDistributor/index.js tasks/core/ActivityRewardDistributor/setupBucket.js
tasks/core/BatchManager/batchManager.deploy.js tasks/core/BatchManager/index.js
tasks/core/Bucket/bucket.deploy.js tasks/core/Bucket/index.js tasks/core/DexAdapter/dexAdapter.deploy.js
tasks/core/DexAdapter/dexAdapter.setDexType.js tasks/core/DexAdapter/index.js
tasks/core/EPMXToken/AddPrimexAddressesToWhitelist.js tasks/core/EPMXToken/EPMXToken.deploy.js
tasks/core/EPMXToken/index.js tasks/core/InterestRateStrategy/InterestRateStrategy.deploy.js
tasks/core/InterestRateStrategy/index.js
tasks/core/KeeperRewardDistributor/KeeperRewardDistributor.deploy.js
tasks/core/KeeperRewardDistributor/index.js tasks/core/LimitOrderManager/index.js
tasks/core/LimitOrderManager/limitOrderManager.deploy.js
tasks/core/LiquidityMiningRewardDistributor/LiquidityMiningRewardDistributor.deploy.js
tasks/core/LiquidityMiningRewardDistributor/index.js tasks/core/PMXPriceFeed/PMXPriceFeed.deploy.js
tasks/core/PMXPriceFeed/index.js tasks/core/PMXToken/PMXToken.deploy.js tasks/core/PMXToken/index.js
tasks/core/PositionManager/index.js tasks/core/PositionManager/positionManager.deploy.js
tasks/core/PriceOracle/index.js tasks/core/PriceOracle/priceOracle.deploy.js
tasks/core/PriceOracle/rinkebyChainlinkPriceFeeds.js tasks/core/PriceOracle/updateFeedsFromConfig.js
tasks/core/PriceOracle/updatePriceFeed.js tasks/core/PriceOracle/updateVolatilityFeed.js
tasks/core/PrimexDNS/PrimexDNS.addDEX.js tasks/core/PrimexDNS/PrimexDNS.deploy.js
tasks/core/PrimexDNS/index.js tasks/core/PrimexDNS/setAavePoolAddress.js
tasks/core/PrimexProxyAdmin/PrimexProxyAdmin.deploy.js tasks/core/PrimexProxyAdmin/index.js
tasks/core/PrimexTimelock/PrimexTimelock.deploy.js tasks/core/PrimexTimelock/index.js
tasks/core/PrimexUpkeep/PrimexUpkeep.deploy.js tasks/core/PrimexUpkeep/index.js
tasks/core/ProtocolFeeCollector/ProtocolFeeCollector.deploy.js tasks/core/ProtocolFeeCollector/index.js
tasks/core/Redeemer/Redeemer.deploy.js tasks/core/Redeemer/index.js tasks/core/Registry/index.js
tasks/core/Registry/registry.deploy.js tasks/core/Reserve/Reserve.deploy.js tasks/core/Reserve/index.js
tasks/core/Reserve/setTransferRestrictionsByConfig.js
tasks/core/SpotTradingRewardDistributor/SpotTradingRewardDistributor.deploy.js
tasks/core/SpotTradingRewardDistributor/index.js tasks/core/SwapManager/index.js
tasks/core/SwapManager/swapManager.deploy.js tasks/core/TraderBalanceVault/index.js
tasks/core/TraderBalanceVault/traderBalanceVault.deploy.js tasks/core/Treasury/Treasury.deploy.js
tasks/core/Treasury/index.js tasks/core/Treasury/setTreasurySpendersByConfig.js
tasks/core/WhiteBlackList/index.js tasks/core/WhiteBlackList/whiteBlackList.deploy.js
tasks/core/conditionalManagers/index.js tasks/core/factories/index.js tasks/core/lens/index.js
tasks/core/libraries/index.js tasks/deployScripts/deployEnvironment/deployEnv.js
tasks/deployScripts/deployEnvironment/deployEnvArbitrum.js
tasks/deployScripts/deployEnvironment/deployEnvDevNode1.js
tasks/deployScripts/deployEnvironment/deployEnvDevnode2.js
tasks/deployScripts/deployEnvironment/deployEnvDevnode3.js
tasks/deployScripts/deployEnvironment/deployEnvGoerli.js
tasks/deployScripts/deployEnvironment/deployEnvMoonbaseAlpha.js
tasks/deployScripts/deployEnvironment/deployEnvMumbai.js
tasks/deployScripts/deployEnvironment/deployEnvObscuro.js
tasks/deployScripts/deployEnvironment/deployEnvPolygonZKtestnet.js
tasks/deployScripts/deployEnvironment/deployEnvZkSync2.js tasks/deployScripts/deployEnvironment/index.js
tasks/deployScripts/phaseSwitching/index.js tasks/deployScripts/phaseSwitching/phase-1-initial-deploy.js
tasks/deployScripts/phaseSwitching/phase-2-execution-spot-trading-rewards.js
tasks/deployScripts/phaseSwitching/phase-2-proposal-spot-trading-rewards.js
tasks/deployScripts/phaseSwitching/phase-3-execution-early-lenders-rewards.js
tasks/deployScripts/phaseSwitching/phase-3-proposal-early-lenders-rewards.js
tasks/deployScripts/phaseSwitching/phase-4-execution-early-traders-rewards.js
tasks/deployScripts/phaseSwitching/phase-4-proposal-early-traders-rewards.js
tasks/deployScripts/phaseSwitching/phase-5-execution-enable-nft.js
tasks/deployScripts/phaseSwitching/phase-5-proposal-enable-nft.js
tasks/deployScripts/phaseSwitching/phase-6-execution-migrate-epmx-to-pmx.js
tasks/deployScripts/phaseSwitching/phase-6-proposal-migrate-epmx-to-pmx.js
tasks/dexes/Aave/Aave.deploy.js tasks/dexes/Aave/addLiquidity.js tasks/dexes/Aave/index.js
tasks/dexes/Balancer/Balancer.deploy.js tasks/dexes/Balancer/CreateWeightedPool.js
tasks/dexes/Balancer/addLiquidity.js tasks/dexes/Balancer/index.js tasks/dexes/Balancer/utils.js
```

```
tasks/dexes/Curve/Curve.deploy.js tasks/dexes/Curve/addLiquidity.js tasks/dexes/Curve/createPool.js
tasks/dexes/Curve/index.js tasks/dexes/Curve/swapExactTokensForTokens.js
tasks/dexes/Meshswap/Meshswap.deploy.js tasks/dexes/Meshswap/createPoolAndAddLiquidity.js
tasks/dexes/Meshswap/index.js tasks/dexes/QuickswapV3/CreatePool.js
tasks/dexes/QuickswapV3/QuickswapV3.deploy.js tasks/dexes/QuickswapV3/SwapExactInputSingle.js
tasks/dexes/QuickswapV3/addLiquidity.js tasks/dexes/QuickswapV3/index.js tasks/dexes/QuickswapV3/utils.js
tasks/dexes/Router/addLiquidity.js tasks/dexes/Router/addLiquidityETH.js tasks/dexes/Router/index.js
tasks/dexes/Router/removeLiquidity.js tasks/dexes/Router/removeLiquidityETH.js
tasks/dexes/Router/swapExactETHForTokens.js tasks/dexes/Router/swapExactTokensForETH.js
tasks/dexes/Router/swapExactTokensForTokens.js tasks/dexes/UniswapV2/UniswapV2.deploy.js
tasks/dexes/UniswapV2/index.js tasks/dexes/UniswapV3/CreatePool.js
tasks/dexes/UniswapV3/SwapExactInputSingle.js tasks/dexes/UniswapV3/UniswapMulticall.deploy.js
tasks/dexes/UniswapV3/UniswapV3.deploy.js tasks/dexes/UniswapV3/addLiquidity.js
tasks/dexes/UniswapV3/index.js tasks/nft/FeeDecreaser/FeeDecreaser.deploy.js
tasks/nft/FeeDecreaser/index.js tasks/nft/InterestIncreaser/InterestIncreaser.deploy.js
tasks/nft/InterestIncreaser/index.js tasks/nft/PMXBonusNFT/PMXBonusNFT.deploy.js
tasks/nft/PMXBonusNFT/index.js tasks/phaseSwitching/phase-2/getPhaseArguments.js
tasks/phaseSwitching/phase-2/setSpotTradingRewardDistributor.js tasks/phaseSwitching/phase-
2/setupSpotTradingRewardDistributorInPM.js tasks/phaseSwitching/phase-
2/topUpSpotTradingRewardDistributor.js tasks/phaseSwitching/phase-3-4/getPhaseArguments.js
tasks/phaseSwitching/phase-3-4/setupBucketsWithRewardsConfiguration.js tasks/phaseSwitching/phase-3-
4/setupEarlyRewardsInBuckets.js tasks/phaseSwitching/phase-3-4/topUpActivityRewardDistributor.js
tasks/phaseSwitching/phase-5/activateBonusNFTs.js tasks/phaseSwitching/phase-5/getPhaseArguments.js
tasks/phaseSwitching/phase-6/deployRewardDistributors.js tasks/phaseSwitching/phase-
6/getPhaseArguments.js tasks/phaseSwitching/phase-6/migrateEpmxToPmx.js tasks/phaseSwitching/phase-
6/updateRewardConfigurationsInBuckets.js tasks/phaseSwitching/phase-6/updateRewardDistributors.js
tasks/referral/ReferralProgram/index.js tasks/referral/ReferralProgram/referralProgram.deploy.js
tasks/referral/WhiteBlackList/index.js tasks/referral/WhiteBlackList/whiteBlackListReferral.deploy.js
tasks/core/conditionalManagers/LimitPriceCOM/LimitPriceCOM.deploy.js
tasks/core/conditionalManagers/LimitPriceCOM/index.js
tasks/core/conditionalManagers/TakeProfitStopLossCCM/TakeProfitStopLossCCM.deploy.js
tasks/core/conditionalManagers/TakeProfitStopLossCCM/index.js
tasks/core/conditionalManagers/TrailingStopCCM/TrailingStopCCM.deploy.js
tasks/core/conditionalManagers/TrailingStopCCM/index.js
tasks/core/factories/BucketsFactory/BucketImplementation.deploy.js
tasks/core/factories/BucketsFactory/BucketsFactory.deploy.js tasks/core/factories/BucketsFactory/index.js
tasks/core/factories/DebtTokensFactory/DebtTokenImplementation.deploy.js
tasks/core/factories/DebtTokensFactory/DebtTokensFactory.deploy.js
tasks/core/factories/DebtTokensFactory/index.js
tasks/core/factories/PTokensFactory/PTokenImplementation.deploy.js
tasks/core/factories/PTokensFactory/PTokensFactory.deploy.js tasks/core/factories/PTokensFactory/index.js
tasks/core/lens/BestDexLens/bestDexLens.deploy.js tasks/core/lens/BestDexLens/index.js
tasks/core/lens/PrimexLens/index.js tasks/core/lens/PrimexLens/primexLens.deploy.js
tasks/core/libraries/Errors/Errors.deploy.js tasks/core/libraries/Errors/index.js
tasks/core/libraries/LimitOrderLibrary/LimitOrderLibrary.deploy.js
tasks/core/libraries/LimitOrderLibrary/index.js
tasks/core/libraries/PositionLibrary/PositionLibrary.deploy.js
tasks/core/libraries/PositionLibrary/index.js
tasks/core/libraries/PrimexPricingLibrary/PrimexPricingLibrary.deploy.js
tasks/core/libraries/PrimexPricingLibrary/index.js
tasks/core/libraries/TokenTransfersLibrary/TokenTransfersLibrary.deploy.js
tasks/core/libraries/TokenTransfersLibrary/index.js config/configUtils.js
$ yarn solhint -c .solhint.json contracts/*.sol contracts/**/*.sol contracts/**/**/*.sol
$ /Users/poanlin/Downloads/primex_finance-primex_contracts-276d36f7002-
github/src/node_modules/.bin/solhint -c .solhint.json contracts/BatchManager.sol contracts/Constants.sol
contracts/DexAdapter.sol contracts/EPMXToken.sol contracts/InterestRateStrategy.sol
contracts/PMXPriceFeed.sol contracts/PMXToken.sol contracts/PrimexProxyAdmin.sol
contracts/PrimexRegistry.sol contracts/PrimexTimelock.sol contracts/PrimexUpkeep.sol
contracts/Redeemer.sol contracts/SwapManager.sol contracts/UniswapInterfaceMulticall.sol
contracts/ActivityRewardDistributor/ActivityRewardDistributor.sol
contracts/ActivityRewardDistributor/ActivityRewardDistributorStorage.sol
contracts/ActivityRewardDistributor/IActivityRewardDistributor.sol
contracts/ActivityRewardDistributor/IActivityRewardDistributorStorage.sol
contracts/BonusExecutor/BonusExecutor.sol contracts/BonusExecutor/FeeDecreaser.sol
contracts/BonusExecutor/FeeExecutor.sol contracts/BonusExecutor/FeeExecutorStorage.sol
contracts/BonusExecutor/IBonusExecutor.sol contracts/BonusExecutor/IFeeExecutor.sol
contracts/BonusExecutor/IFeeExecutorStorage.sol contracts/BonusExecutor/InterestIncreaser.sol
contracts/Bucket/Bucket.sol contracts/Bucket/BucketStorage.sol contracts/Bucket/BucketsFactory.sol
contracts/Bucket/IBucket.sol contracts/Bucket/IBucketStorage.sol contracts/Bucket/IBucketsFactory.sol
contracts/DebtToken/DebtToken.sol contracts/DebtToken/DebtTokenFactory.sol
contracts/DebtToken/DebtTokenStorage.sol contracts/DebtToken/IDebtToken.sol
contracts/DebtToken/IDebtTokenStorage.sol contracts/DebtToken/IDebtTokensFactory.sol
```

```
contracts/KeeperRewardDistributor/IKeeperRewardDistributor.sol
contracts/KeeperRewardDistributor/IKeeperRewardDistributorStorage.sol
contracts/KeeperRewardDistributor/KeeperRewardDistributor.sol
contracts/KeeperRewardDistributor/KeeperRewardDistributorStorage.sol
contracts/LimitOrderManager/ILimitOrderManager.sol
contracts/LimitOrderManager/ILimitOrderManagerStorage.sol
contracts/LimitOrderManager/LimitOrderManager.sol
contracts/LimitOrderManager/LimitOrderManagerStorage.sol
contracts/LiquidityMiningRewardDistributor/ILiquidityMiningRewardDistributor.sol
contracts/LiquidityMiningRewardDistributor/ILiquidityMiningRewardDistributorStorage.sol
contracts/LiquidityMiningRewardDistributor/LiquidityMiningRewardDistributor.sol
contracts/LiquidityMiningRewardDistributor/LiquidityMiningRewardDistributorStorage.sol
contracts/PMXBonusNFT/IPMXBonusNFT.sol contracts/PMXBonusNFT/IPMXBonusNFTStorage.sol
contracts/PMXBonusNFT/PMXBonusNFT.sol contracts/PMXBonusNFT/PMXBonusNFTStorage.sol
contracts/PToken/IPToken.sol contracts/PToken/IPTokenStorage.sol contracts/PToken/IPTokensFactory.sol
contracts/PToken/PToken.sol contracts/PToken/PTokenStorage.sol contracts/PToken/PTokensFactory.sol
contracts/PositionManager/IBasePositionManager.sol contracts/PositionManager/IPositionManager.sol
contracts/PositionManager/IPositionManagerStorage.sol contracts/PositionManager/PositionManager.sol
contracts/PositionManager/PositionManagerStorage.sol contracts/PriceOracle/IPriceOracle.sol
contracts/PriceOracle/IPriceOracleStorage.sol contracts/PriceOracle/PriceOracle.sol
contracts/PriceOracle/PriceOracleStorage.sol contracts/PrimexDNS/IPrimexDNS.sol
contracts/PrimexDNS/IPrimexDNSStorage.sol contracts/PrimexDNS/PrimexDNS.sol
contracts/PrimexDNS/PrimexDNSStorage.sol contracts/ReferralProgram/IReferralProgram.sol
contracts/ReferralProgram/IReferralProgramStorage.sol contracts/ReferralProgram/ReferralProgram.sol
contracts/ReferralProgram/ReferralProgramStorage.sol contracts/Reserve/IReserve.sol
contracts/Reserve/IReserveStorage.sol contracts/Reserve/Reserve.sol contracts/Reserve/ReserveStorage.sol
contracts/SpotTradingRewardDistributor/ISpotTradingRewardDistributor.sol
contracts/SpotTradingRewardDistributor/ISpotTradingRewardDistributorStorage.sol
contracts/SpotTradingRewardDistributor/SpotTradingRewardDistributor.sol
contracts/SpotTradingRewardDistributor/SpotTradingRewardDistributorStorage.sol
contracts/TestnetServices/BalancerBotLens.sol contracts/TestnetServices/CurveBotLens.sol
contracts/TestnetServices/PriceFeedUpdaterTestService.sol
contracts/TestnetServices/PrimexAggregatorV3TestService.sol
contracts/TestnetServices/SynchronizationBotLens.sol
contracts/TestnetServices/SynchronizationBotLensQuickswapTestService.sol
contracts/TestnetServices/SynchronizationBotLensUniV3TestService.sol
contracts/TraderBalanceVault/ITraderBalanceVault.sol
contracts/TraderBalanceVault/ITraderBalanceVaultStorage.sol
contracts/TraderBalanceVault/TraderBalanceVault.sol
contracts/TraderBalanceVault/TraderBalanceVaultStorage.sol contracts/Treasury/ITreasury.sol
contracts/Treasury/ITreasuryStorage.sol contracts/Treasury/Treasury.sol
contracts/Treasury/TreasuryStorage.sol contracts/conditionalManagers/LimitPriceCOM.sol
contracts/conditionalManagers/TakeProfitStopLossCCM.sol contracts/conditionalManagers/TrailingStopCCM.sol
contracts/interfaces/IAsset.sol contracts/interfaces/IBalancer.sol contracts/interfaces/IBatchManager.sol
contracts/interfaces/IBestDexLens.sol contracts/interfaces/IConditionalClosingManager.sol
contracts/interfaces/IConditionalOpeningManager.sol contracts/interfaces/IDexAdapter.sol
contracts/interfaces/IEPMXToken.sol contracts/interfaces/IERC20Mock.sol
contracts/interfaces/IInterestRateStrategy.sol contracts/interfaces/ILimitPriceCOM.sol
contracts/interfaces/IPMXPriceFeed.sol contracts/interfaces/IPausable.sol
contracts/interfaces/IPriceFeedUpdaterTestService.sol contracts/interfaces/IPrimexLens.sol
contracts/interfaces/IPrimexPricingLibraryMock.sol contracts/interfaces/IPrimexRegistry.sol
contracts/interfaces/IPrimexUpkeep.sol contracts/interfaces/IRedeemer.sol
contracts/interfaces/ISwapManager.sol contracts/interfaces/ITakeProfitStopLossCCM.sol
contracts/interfaces/ITrailingStopCCM.sol contracts/interfaces/IWhitelist.sol
contracts/lens/BestDexLens.sol contracts/lens/PrimexLens.sol contracts/libraries/Errors.sol
contracts/libraries/LimitOrderLibrary.sol contracts/libraries/PositionLibrary.sol
contracts/libraries/PrimexPricingLibrary.sol contracts/libraries/TokenTransfersLibrary.sol
contracts/mocks/AttackerBucket.sol contracts/mocks/BucketMock.sol contracts/mocks/ERC20Mock.sol
contracts/mocks/MaliciousDexMock.sol contracts/mocks/MockNearestSearch.sol
contracts/mocks/PrimexPricingLibraryMock.sol contracts/mocks/TokenTransfersLibraryMock.sol
contracts/TestnetServices/interfaces/IBalancerBotLens.sol
contracts/TestnetServices/interfaces/ICurveBotLens.sol
contracts/TestnetServices/interfaces/IPrimexAggregatorV3TestService.sol
contracts/TestnetServices/interfaces/ISynchronizationBotLens.sol
contracts/TestnetServices/interfaces/ISynchronizationBotLensQuickswapTestService.sol
contracts/TestnetServices/interfaces/ISynchronizationBotLensUniV3TestService.sol
contracts/WhiteBlackList/WhiteBlackList/IWhiteBlackList.sol
contracts/WhiteBlackList/WhiteBlackList/WhiteBlackList.sol
contracts/WhiteBlackList/WhiteBlackListBase/WhiteBlackListBase.sol
contracts/WhiteBlackList/WhiteBlackListReferral/IWhiteBlackListReferral.sol
contracts/WhiteBlackList/WhiteBlackListReferral/WhiteBlackListReferral.sol
contracts/WhiteBlackList/WhiteBlackListReferral/WhiteBlackListReferralStorage.sol
```

```
contracts/interfaces/routers/ICurveCalc.sol contracts/interfaces/routers/ICurvePool.sol
contracts/interfaces/routers/ICurveRegistry.sol contracts/interfaces/routers/ICurveRouter.sol
contracts/libraries/utils/BytesLib.sol contracts/libraries/utils/V3Path.sol
contracts/libraries/utils/WadRayMath.sol contracts/mocks/mocksInterfaces/IAttackerBucket.sol
contracts/mocks/mocksInterfaces/IBucketMock.sol contracts/mocks/mocksInterfaces/IMaliciousDexMock.sol
contracts/mocks/mocksInterfaces/IMockNearestSearch.sol contracts/mocks/upgradeMocks/BucketV2.sol
contracts/mocks/upgradeMocks/DebtTokenV2.sol contracts/mocks/upgradeMocks/FeeDecreaserV2.sol
contracts/mocks/upgradeMocks/InterestIncreaserV2.sol
contracts/mocks/upgradeMocks/KeeperRewardDistributorV2.sol
contracts/mocks/upgradeMocks/LimitOrderManagerV2.sol
contracts/mocks/upgradeMocks/LiquidityMiningRewardDistributorV2.sol
contracts/mocks/upgradeMocks/PMXBonusNFTV2.sol contracts/mocks/upgradeMocks/PTokenV2.sol
contracts/mocks/upgradeMocks/PositionManagerV2.sol contracts/mocks/upgradeMocks/PriceOracleV2.sol
contracts/mocks/upgradeMocks/PrimexDNSV2.sol contracts/mocks/upgradeMocks/ReferralProgramV2.sol
contracts/mocks/upgradeMocks/ReserveV2.sol
contracts/mocks/upgradeMocks/SpotTradingRewardDistributorV2.sol
contracts/mocks/upgradeMocks/TraderBalanceVaultV2.sol contracts/mocks/upgradeMocks/TreasuryV2.sol
contracts/mocks/upgradeMocks/WhiteBlackListReferralV2.sol
contracts/mocks/upgradeMocksInterfaces/IBucketV2.sol
contracts/mocks/upgradeMocksInterfaces/IDebtTokenV2.sol
contracts/mocks/upgradeMocksInterfaces/IFeeDecreaserV2.sol
contracts/mocks/upgradeMocksInterfaces/IInterestIncreaserV2.sol
contracts/mocks/upgradeMocksInterfaces/IKeeperRewardDistributorV2.sol
contracts/mocks/upgradeMocksInterfaces/ILimitOrderManagerV2.sol
contracts/mocks/upgradeMocksInterfaces/ILiquidityMiningRewardDistributorV2.sol
contracts/mocks/upgradeMocksInterfaces/IPMXBonusNFTV2.sol
contracts/mocks/upgradeMocksInterfaces/IPTokenV2.sol
contracts/mocks/upgradeMocksInterfaces/IPositionManagerV2.sol
contracts/mocks/upgradeMocksInterfaces/IPriceOracleV2.sol
contracts/mocks/upgradeMocksInterfaces/IPrimexDNSV2.sol
contracts/mocks/upgradeMocksInterfaces/IReferralProgramV2.sol
contracts/mocks/upgradeMocksInterfaces/IReserveV2.sol
contracts/mocks/upgradeMocksInterfaces/ISpotTradingRewardDistributorV2.sol
contracts/mocks/upgradeMocksInterfaces/ITraderBalanceVaultV2.sol
contracts/mocks/upgradeMocksInterfaces/ITreasuryV2.sol
contracts/mocks/upgradeMocksInterfaces/IWhiteBlackListReferralV2.sol

$ /Users/poanlin/Downloads/primex_finance-primex_contracts-276d36f7002-
github/src/node_modules/.bin/hardhat test
`optimizer` setting is deprecated, optimizer is always enabled
No need to generate any newer typings.
 ·-----------------------------------------|--------------|----------------·
 |  Contract Name                          ·  Size (KiB)  ·  Change (KiB)  |
 ··········································|··············|·················
 |  ActivityRewardDistributor              ·      10.932  ·                |
 ··········································|··············|·················
 |  Address                                ·       0.084  ·                |
 ··········································|··············|·················
 |  AddressUpgradeable                     ·       0.084  ·                |
 ··········································|··············|·················
 |  AttackerBucket                         ·       1.409  ·                |
 ··········································|··············|·················
 |  BalancerBotLens                        ·       5.071  ·                |
 ··········································|··············|·················
 |  BatchManager                           ·      13.716  ·                |
 ··········································|··············|·················
 |  BeaconProxy                            ·       0.916  ·                |
 ··········································|··············|·················
 |  BestDexLens                            ·      18.472  ·                |
 ··········································|··············|·················
 |  Bucket                                 ·      25.393  ·                |
 ··········································|··············|·················
 |  BucketMock                             ·       6.820  ·                |
 ··········································|··············|·················
 |  BucketsFactory                         ·      10.307  ·                |
 ··········································|··············|·················
 |  BucketV2                               ·      25.438  ·                |
 ··········································|··············|·················
 |  BytesLib                               ·       0.084  ·                |
 ··········································|··············|·················
 |  console                                ·       0.084  ·                |
 ··········································|··············|·················
```

CurveBotLens	3.830	
DataTypes	0.084	
DebtToken	8.543	
DebtTokensFactory	4.849	
DebtTokenV2	8.607	
DexAdapter	13.712	
ECDSA	0.084	
ECDSAUpgradeable	0.084	
EPMXToken	4.907	
ERC1967Proxy	0.736	
ERC20	2.420	
ERC20Mock	4.052	
ERC20Upgradeable	2.420	
ERC721Upgradeable	4.875	
Errors	0.084	
FeeDecreaser	12.043	
FeeDecreaserV2	12.117	
InterestIncreaser	12.043	
InterestIncreaserV2	12.122	
InterestRateStrategy	1.351	
KeeperRewardDistributor	6.917	
KeeperRewardDistributorV2	7.090	
LimitOrderLibrary	14.908	
LimitOrderManager	25.277	
LimitOrderManagerV2	25.369	
LimitPriceCOM	6.965	
LiquidityMiningRewardDistributor	11.190	
LiquidityMiningRewardDistributorStorage	1.045	
LiquidityMiningRewardDistributorV2	11.286	
MaliciousDexMock	0.655	
Math	0.084	
MathUpgradeable	0.084	
MockNearestSearch	7.775	
PMXBonusNFT	14.431	
PMXBonusNFTV2	14.481	
PMXPriceFeed	1.721	

PMXToken	2.556	
PositionLibrary	21.662	
PositionManager	29.979	
PositionManagerV2	30.084	
PriceFeedUpdaterTestService	7.251	
PriceOracle	5.563	
PriceOracleV2	5.720	
PrimexAggregatorV3TestService	3.273	
PrimexDNS	8.492	
PrimexDNSV2	8.563	
PrimexLens	26.051	
PrimexPricingLibrary	16.761	
PrimexPricingLibraryMock	3.726	
PrimexProxyAdmin	2.459	
PrimexRegistry	2.217	
PrimexTimelock	9.452	
PrimexUpkeep	15.163	
ProxyAdmin	1.976	
PToken	13.680	
PTokensFactory	4.849	
PTokenV2	13.726	
Redeemer	2.820	
ReferralProgram	6.059	
ReferralProgramV2	6.113	
Reserve	6.646	
ReserveV2	6.698	
SignedMath	0.084	
SpotTradingRewardDistributor	6.680	
SpotTradingRewardDistributorV2	6.868	
StorageSlot	0.084	
Strings	0.084	
StringsUpgradeable	0.084	
SwapManager	7.334	
SynchronizationBotLens	6.944	
SynchronizationBotLensQuickswapTestService	3.647	
SynchronizationBotLensUniV3TestService	3.724	

```
|  TakeProfitStopLossCCM                     ·       5.819  ·                |
·--------------------------------------------|-------------|-----------------·
|  TimelockController                        ·       8.357  ·                |
·--------------------------------------------|-------------|-----------------·
|  TokenTransfersLibrary                     ·       1.405  ·                |
·--------------------------------------------|-------------|-----------------·
|  TokenTransfersLibraryMock                 ·       0.475  ·                |
·--------------------------------------------|-------------|-----------------·
|  TraderBalanceVault                        ·       7.282  ·                |
·--------------------------------------------|-------------|-----------------·
|  TraderBalanceVaultV2                      ·       7.424  ·                |
·--------------------------------------------|-------------|-----------------·
|  TrailingStopCCM                           ·       3.509  ·                |
·--------------------------------------------|-------------|-----------------·
|  TransparentUpgradeableProxy               ·       2.257  ·                |
·--------------------------------------------|-------------|-----------------·
|  Treasury                                  ·       5.800  ·                |
·--------------------------------------------|-------------|-----------------·
|  TreasuryV2                                ·       5.947  ·                |
·--------------------------------------------|-------------|-----------------·
|  UniswapInterfaceMulticall                 ·       1.813  ·                |
·--------------------------------------------|-------------|-----------------·
|  UpgradeableBeacon                         ·       0.952  ·                |
·--------------------------------------------|-------------|-----------------·
|  V3Path                                    ·       0.084  ·                |
·--------------------------------------------|-------------|-----------------·
|  WadRayMath                                ·       0.084  ·                |
·--------------------------------------------|-------------|-----------------·
|  WhiteBlackList                            ·       3.407  ·                |
·--------------------------------------------|-------------|-----------------·
|  WhiteBlackListReferral                    ·       5.301  ·                |
·--------------------------------------------|-------------|-----------------·
|  WhiteBlackListReferralV2                  ·       5.438  ·                |
·--------------------------------------------|-------------|-----------------·

Warning: 7 contracts exceed the size limit for mainnet deployment.
<s> [webpack.Progress] 0%

<s> [webpack.Progress] 1% setup before run
<s> [webpack.Progress] 1% setup before run NodeEnvironmentPlugin
<s> [webpack.Progress] 1% setup before run
<s> [webpack.Progress] 2% setup run
<s> [webpack.Progress] 2% setup run
<s> [webpack.Progress] 4% setup normal module factory
<s> [webpack.Progress] 4% setup normal module factory
<s> [webpack.Progress] 5% setup context module factory
<s> [webpack.Progress] 5% setup context module factory
<s> [webpack.Progress] 6% setup before compile
<s> [webpack.Progress] 6% setup before compile ProgressPlugin
<s> [webpack.Progress] 6% setup before compile
<s> [webpack.Progress] 7% setup compile
<s> [webpack.Progress] 7% setup compile ExternalsPlugin
<s> [webpack.Progress] 7% setup compile
<s> [webpack.Progress] 8% setup compilation
<s> [webpack.Progress] 8% setup compilation ArrayPushCallbackChunkFormatPlugin
<s> [webpack.Progress] 8% setup compilation JsonpChunkLoadingPlugin
<s> [webpack.Progress] 8% setup compilation StartupChunkDependenciesPlugin
<s> [webpack.Progress] 8% setup compilation ImportScriptsChunkLoadingPlugin
<s> [webpack.Progress] 8% setup compilation FetchCompileWasmPlugin
<s> [webpack.Progress] 8% setup compilation FetchCompileAsyncWasmPlugin
<s> [webpack.Progress] 8% setup compilation WorkerPlugin
<s> [webpack.Progress] 8% setup compilation SplitChunksPlugin
<s> [webpack.Progress] 8% setup compilation ResolverCachePlugin
<s> [webpack.Progress] 8% setup compilation HtmlWebpackPlugin
<s> [webpack.Progress] 8% setup compilation
<s> [webpack.Progress] 9% setup compilation
<s> [webpack.Progress] 9% setup compilation ProgressPlugin
<s> [webpack.Progress] 9% setup compilation vue-loader-plugin
<s> [webpack.Progress] 9% setup compilation DefinePlugin
<s> [webpack.Progress] 9% setup compilation ChunkPrefetchPreloadPlugin
<s> [webpack.Progress] 9% setup compilation JavascriptModulesPlugin
<s> [webpack.Progress] 9% setup compilation JsonModulesPlugin
```

```
<s> [webpack.Progress] 9% setup compilation AssetModulesPlugin
<s> [webpack.Progress] 9% setup compilation EntryPlugin
<s> [webpack.Progress] 9% setup compilation RuntimePlugin
<s> [webpack.Progress] 9% setup compilation InferAsyncModulesPlugin
<s> [webpack.Progress] 9% setup compilation DataUriPlugin
<s> [webpack.Progress] 9% setup compilation FileUriPlugin
<s> [webpack.Progress] 9% setup compilation CompatibilityPlugin
<s> [webpack.Progress] 9% setup compilation HarmonyModulesPlugin
<s> [webpack.Progress] 9% setup compilation AMDPlugin
<s> [webpack.Progress] 9% setup compilation RequireJsStuffPlugin
<s> [webpack.Progress] 9% setup compilation CommonJsPlugin
<s> [webpack.Progress] 9% setup compilation LoaderPlugin
<s> [webpack.Progress] 9% setup compilation LoaderPlugin
<s> [webpack.Progress] 9% setup compilation NodeStuffPlugin
<s> [webpack.Progress] 9% setup compilation APIPlugin
<s> [webpack.Progress] 9% setup compilation ExportsInfoApiPlugin
<s> [webpack.Progress] 9% setup compilation WebpackIsIncludedPlugin
<s> [webpack.Progress] 9% setup compilation ConstPlugin
<s> [webpack.Progress] 9% setup compilation UseStrictPlugin
<s> [webpack.Progress] 9% setup compilation RequireIncludePlugin
<s> [webpack.Progress] 9% setup compilation RequireEnsurePlugin
<s> [webpack.Progress] 9% setup compilation RequireContextPlugin
<s> [webpack.Progress] 9% setup compilation ImportPlugin
<s> [webpack.Progress] 9% setup compilation RequireContextPlugin
<s> [webpack.Progress] 9% setup compilation SystemPlugin
<s> [webpack.Progress] 9% setup compilation ImportMetaPlugin
<s> [webpack.Progress] 9% setup compilation URLPlugin
<s> [webpack.Progress] 9% setup compilation DefaultStatsFactoryPlugin
<s> [webpack.Progress] 9% setup compilation DefaultStatsPresetPlugin
<s> [webpack.Progress] 9% setup compilation DefaultStatsPrinterPlugin
<s> [webpack.Progress] 9% setup compilation JavascriptMetaInfoPlugin
<s> [webpack.Progress] 9% setup compilation EnsureChunkConditionsPlugin
<s> [webpack.Progress] 9% setup compilation RemoveEmptyChunksPlugin
<s> [webpack.Progress] 9% setup compilation MergeDuplicateChunksPlugin
<s> [webpack.Progress] 9% setup compilation FlagIncludedChunksPlugin
<s> [webpack.Progress] 9% setup compilation SideEffectsFlagPlugin
<s> [webpack.Progress] 9% setup compilation FlagDependencyExportsPlugin
<s> [webpack.Progress] 9% setup compilation FlagDependencyUsagePlugin
<s> [webpack.Progress] 9% setup compilation InnerGraphPlugin
<s> [webpack.Progress] 9% setup compilation MangleExportsPlugin
<s> [webpack.Progress] 9% setup compilation ModuleConcatenationPlugin
<s> [webpack.Progress] 9% setup compilation NoEmitOnErrorsPlugin
<s> [webpack.Progress] 9% setup compilation RealContentHashPlugin
<s> [webpack.Progress] 9% setup compilation WasmFinalizeExportsPlugin
<s> [webpack.Progress] 9% setup compilation DeterministicModuleIdsPlugin
<s> [webpack.Progress] 9% setup compilation DeterministicChunkIdsPlugin
<s> [webpack.Progress] 9% setup compilation DefinePlugin
<s> [webpack.Progress] 9% setup compilation TerserPlugin
<s> [webpack.Progress] 9% setup compilation TemplatedPathPlugin
<s> [webpack.Progress] 9% setup compilation RecordIdsPlugin
<s> [webpack.Progress] 9% setup compilation WarnCaseSensitiveModulesPlugin
<s> [webpack.Progress] 9% setup compilation
<s> [webpack.Progress] 10% building
<s> [webpack.Progress] 10% building 0/1 entries 0/0 dependencies 0/0 modules
<s> [webpack.Progress] 10% building import loader ./node_modules/vue-loader/lib/index.js
<s> [webpack.Progress] 10% building 0/1 entries 5/6 dependencies 0/5 modules
<s> [webpack.Progress] 10% building import loader ./node_modules/vue-loader/lib/loaders/pitcher.js
<s> [webpack.Progress] 10% building import loader ./node_modules/vue-loader/lib/loaders/templateLoader.js
<s> [webpack.Progress] 10% building 0/1 entries 14/15 dependencies 1/12 modules
<s> [webpack.Progress] 10% building import loader ./node_modules/vue-style-loader/index.js
<s> [webpack.Progress] 10% building import loader ./node_modules/css-loader/dist/cjs.js
<s> [webpack.Progress] 10% building 0/1 entries 26/26 dependencies 9/23 modules
<s> [webpack.Progress] 10% building import loader ./node_modules/vue-
loader/lib/loaders/stylePostLoader.js
<s> [webpack.Progress] 10% building 0/1 entries 38/41 dependencies 9/31 modules
<s> [webpack.Progress] 65% building 1/1 entries 66/66 dependencies 53/53 modules
<s> [webpack.Progress] 65% building
<s> [webpack.Progress] 69% building finish
<s> [webpack.Progress] 69% building finish
<s> [webpack.Progress] 70% sealing finish module graph
<s> [webpack.Progress] 70% sealing finish module graph ResolverCachePlugin
<s> [webpack.Progress] 70% sealing finish module graph InferAsyncModulesPlugin
```

```
<s> [webpack.Progress] 70% sealing finish module graph FlagDependencyExportsPlugin
<s> [webpack.Progress] 70% sealing finish module graph InnerGraphPlugin
<s> [webpack.Progress] 70% sealing finish module graph WasmFinalizeExportsPlugin
<s> [webpack.Progress] 70% sealing finish module graph
<s> [webpack.Progress] 70% sealing plugins
<s> [webpack.Progress] 70% sealing plugins WarnCaseSensitiveModulesPlugin
<s> [webpack.Progress] 70% sealing plugins
<s> [webpack.Progress] 71% sealing dependencies optimization
<s> [webpack.Progress] 71% sealing dependencies optimization SideEffectsFlagPlugin
<s> [webpack.Progress] 71% sealing dependencies optimization FlagDependencyUsagePlugin
<s> [webpack.Progress] 71% sealing dependencies optimization
<s> [webpack.Progress] 71% sealing after dependencies optimization
<s> [webpack.Progress] 71% sealing after dependencies optimization
<s> [webpack.Progress] 72% sealing chunk graph
<s> [webpack.Progress] 72% sealing chunk graph
<s> [webpack.Progress] 73% sealing after chunk graph
<s> [webpack.Progress] 73% sealing after chunk graph
<s> [webpack.Progress] 73% sealing optimizing
<s> [webpack.Progress] 73% sealing optimizing
<s> [webpack.Progress] 74% sealing module optimization
<s> [webpack.Progress] 74% sealing module optimization
<s> [webpack.Progress] 75% sealing after module optimization
<s> [webpack.Progress] 75% sealing after module optimization
<s> [webpack.Progress] 75% sealing chunk optimization
<s> [webpack.Progress] 75% sealing chunk optimization EnsureChunkConditionsPlugin
<s> [webpack.Progress] 75% sealing chunk optimization RemoveEmptyChunksPlugin
<s> [webpack.Progress] 75% sealing chunk optimization MergeDuplicateChunksPlugin
<s> [webpack.Progress] 75% sealing chunk optimization SplitChunksPlugin
<s> [webpack.Progress] 75% sealing chunk optimization RemoveEmptyChunksPlugin
<s> [webpack.Progress] 75% sealing chunk optimization
<s> [webpack.Progress] 76% sealing after chunk optimization
<s> [webpack.Progress] 76% sealing after chunk optimization
<s> [webpack.Progress] 77% sealing module and chunk tree optimization
<s> [webpack.Progress] 77% sealing module and chunk tree optimization
PersistentChildCompilerSingletonPlugin
<s> [webpack.Progress] 77% sealing module and chunk tree optimization
<s> [webpack.Progress] 77% sealing after module and chunk tree optimization
<s> [webpack.Progress] 77% sealing after module and chunk tree optimization
<s> [webpack.Progress] 78% sealing chunk modules optimization
<s> [webpack.Progress] 78% sealing chunk modules optimization ModuleConcatenationPlugin
<s> [webpack.Progress] 78% sealing chunk modules optimization
<s> [webpack.Progress] 78% sealing after chunk modules optimization
<s> [webpack.Progress] 78% sealing after chunk modules optimization
<s> [webpack.Progress] 79% sealing module reviving
<s> [webpack.Progress] 79% sealing module reviving RecordIdsPlugin
<s> [webpack.Progress] 79% sealing module reviving
<s> [webpack.Progress] 80% sealing before module ids
<s> [webpack.Progress] 80% sealing before module ids
<s> [webpack.Progress] 80% sealing module ids
<s> [webpack.Progress] 80% sealing module ids DeterministicModuleIdsPlugin
<s> [webpack.Progress] 80% sealing module ids
<s> [webpack.Progress] 81% sealing module id optimization
<s> [webpack.Progress] 81% sealing module id optimization
<s> [webpack.Progress] 82% sealing module id optimization
<s> [webpack.Progress] 82% sealing module id optimization
<s> [webpack.Progress] 82% sealing chunk reviving
<s> [webpack.Progress] 82% sealing chunk reviving RecordIdsPlugin
<s> [webpack.Progress] 82% sealing chunk reviving
<s> [webpack.Progress] 83% sealing before chunk ids
<s> [webpack.Progress] 83% sealing before chunk ids
<s> [webpack.Progress] 84% sealing chunk ids
<s> [webpack.Progress] 84% sealing chunk ids DeterministicChunkIdsPlugin
<s> [webpack.Progress] 84% sealing chunk ids
<s> [webpack.Progress] 84% sealing chunk id optimization
<s> [webpack.Progress] 84% sealing chunk id optimization FlagIncludedChunksPlugin
<s> [webpack.Progress] 84% sealing chunk id optimization
<s> [webpack.Progress] 85% sealing after chunk id optimization
<s> [webpack.Progress] 85% sealing after chunk id optimization
<s> [webpack.Progress] 86% sealing record modules
<s> [webpack.Progress] 86% sealing record modules RecordIdsPlugin
<s> [webpack.Progress] 86% sealing record modules
<s> [webpack.Progress] 86% sealing record chunks
```

```
<s> [webpack.Progress] 86% sealing record chunks RecordIdsPlugin
<s> [webpack.Progress] 86% sealing record chunks
<s> [webpack.Progress] 87% sealing module hashing
<s> [webpack.Progress] 87% sealing module hashing
<s> [webpack.Progress] 87% sealing code generation
<s> [webpack.Progress] 87% sealing code generation
<s> [webpack.Progress] 88% sealing runtime requirements
<s> [webpack.Progress] 88% sealing runtime requirements
<s> [webpack.Progress] 89% sealing hashing
<s> [webpack.Progress] 89% sealing hashing
<s> [webpack.Progress] 89% sealing after hashing
<s> [webpack.Progress] 89% sealing after hashing
<s> [webpack.Progress] 90% sealing record hash
<s> [webpack.Progress] 90% sealing record hash
<s> [webpack.Progress] 91% sealing module assets processing
<s> [webpack.Progress] 91% sealing module assets processing
<s> [webpack.Progress] 91% sealing chunk assets processing
<s> [webpack.Progress] 91% sealing chunk assets processing
<s> [webpack.Progress] 92% sealing asset processing
<s> [webpack.Progress] 92% sealing asset processing PersistentChildCompilerSingletonPlugin
<s> [webpack.Progress] 92% sealing asset processing TerserPlugin
<s> [webpack.Progress] 92% sealing asset processing HtmlWebpackPlugin
<s> [webpack.Progress] 92% sealing asset processing RealContentHashPlugin
<s> [webpack.Progress] 92% sealing asset processing
<s> [webpack.Progress] 93% sealing after asset optimization
<s> [webpack.Progress] 93% sealing after asset optimization
<s> [webpack.Progress] 93% sealing recording
<s> [webpack.Progress] 93% sealing recording
<s> [webpack.Progress] 94% sealing after seal
<s> [webpack.Progress] 94% sealing after seal
<s> [webpack.Progress] 95% emitting emit
<s> [webpack.Progress] 95% emitting emit
<s> [webpack.Progress] 98% emitting after emit
<s> [webpack.Progress] 98% emitting after emit SizeLimitsPlugin
<s> [webpack.Progress] 98% emitting after emit
<s> [webpack.Progress] 99% done plugins
<s> [webpack.Progress] 99% done plugins
<s> [webpack.Progress] 99%

<s> [webpack.Progress] 99% cache store build dependencies
<s> [webpack.Progress] 99% cache store build dependencies
<s> [webpack.Progress] 99% cache begin idle
<s> [webpack.Progress] 99% cache begin idle
<s> [webpack.Progress] 100%

<s> [webpack.Progress] 99% cache shutdown
<s> [webpack.Progress] 99% cache shutdown
<s> [webpack.Progress] 100%



  Balancer
Mints to test accounts completed.
=== Access to transfer EPMX is added ===
    ✔ Creating position through Balancer should work (442ms)
    ✔ should be reverted when the pool id is incorrect (350ms)
    ✔ getAmountOut return correct value  (228ms)

  BestDexLens
    ✔ should be revert if the length data differs (40ms)
    getBestMultipleDexes
      ✔ should getBestMultipleDexes when the amount to sell and shares are equal (950ms)
      ✔ should getBestMultipleDexes (852ms)
      ✔ should getBestMultipleDexes when the isAmountToBuy is true (691ms)
      ✔ should getBestMultipleDexes with equal shares (797ms)
      ✔ Should revert getBestMultipleDexes when zero assetToBuy address
      ✔ Should revert getBestMultipleDexes when zero assetToSell address
      ✔ Should revert when getBestMultipleDexes when assetToBuy == assetToSell
      ✔ Should revert getBestMultipleDexes when shares is 0
      ✔ Should revert getBestMultipleDexes when the shares is greater than the amount to sell
      ✔ should getBestMultipleDexes with the fourth dex (3059ms)
      ✔ should have more amount out than single dex (2090ms)
```

```
      ✔ should have more amount out than single dex when sushiswap has low price for testTokenB (1875ms)
      ✔ should have more amount out than single dex when uniswapv2 has low price for testTokenB (2601ms)
      ✔ should have more amount out than single dex when uniswapv3 has low price for testTokenB (2928ms)
      ✔ should getBestMultipleDexes for amounts less than gas fees (2346ms)
      ✔ should swap all amount on one dex when it has best price (1969ms)
    getBestDexByOrder
      ✔ should revert when positionManager address not supported (146ms)
      ✔ should revert when limitOrderManager address not supported (141ms)
    getBestDexForOpenablePosition
      ✔ Should be reverted when one of the shares equal to zero (521ms)
      ✔ Should be reverted when one of the addresses params is equal to zero (54ms)
      ✔ Should be reverted when depositedAmount is equal to zero
      ✔ getBestDexForOpenablePosition return the correct values when deposit in the third asset (905ms)
      ✔ getBestDexForOpenablePosition return the correct values when deposit in the borrowed asset
(314ms)
      ✔ getBestDexForOpenablePosition return the correct values when deposit in the position asset
(318ms)
      ✔ getBestDexForOpenablePosition return the correct values when multiple dexes (1169ms)
      ✔ getPositionProfit return correct values (1183ms)
      ✔ getCurrentPriceAndProfitByPosition should be reverted when shares is equal zero
      ✔ getCurrentPriceAndProfitByPosition should be reverted when the address of position manager does
not match the its interface
      ✔ getCurrentPriceAndProfitByPosition return correct values (403ms)
      ✔ Should be revert when shares is zero
      ✔ getCurrentPriceAndProfitByPosition return correct values when multiple dexes (1002ms)
      ✔ getArrayCurrentPriceAndProfitByPosition return correct values when multiple dexes (3271ms)
      ✔ getArrayCurrentPriceAndProfitByPosition return correct values (1445ms)
      ✔ getArrayCurrentPriceAndProfitByPosition should be reverted when shares is equal zero (50ms)

  Bucket

Accounts
========
┌─────────┬──────────────────────────────────┬──────────────────────────────────────────┬──────────────
──────────┐
│ (index) │               name               │                 account                  │
balance          │
├─────────┼──────────────────────────────────┼──────────────────────────────────────────┼──────────────
──────────┤
│    0    │            'deployer'            │ '0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266' │
'9999.758524004895093496' │
│    1    │          'epmxDeployer'          │ '0x70997970C51812dc3A010C7d01b50e0d17dc79C8' │
'9999.998285555454949189' │
│    2    │             'holder'             │ '0x70997970C51812dc3A010C7d01b50e0d17dc79C8' │
'9999.998285555454949189' │
│    3    │           'recipient'            │ '0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC' │
'10000.0'         │
│    4    │             'caller'             │ '0x90F79bf6EB2c4f870365E785982E1f101E93b906' │
'10000.0'         │
│    5    │             'lender'             │ '0x15d34AAf54267DB7D7c367839AAf71A00a2C6A65' │
'10000.0'         │
│    6    │            'lender2'             │ '0x9965507D1a55bcC2695C58ba16FB37d819B0A4dc' │
'10000.0'         │
│    7    │             'trader'             │ '0x976EA74026E726554dB657fA54763abd0C3a0aa9' │
'10000.0'         │
│    8    │            'trader2'             │ '0x14dC79964da2C08b23698B3D3cc7Ca32193d9955' │
'10000.0'         │
│    9    │          'liquidator'            │ '0x23618e81E3f5cdF7f54C3d65f7FBc0aBf5B21E8f' │
'10000.0'         │
│   10    │              'user'              │ '0xa0Ee7A142d267C1f36714E4a8F75612F20a79720' │
'10000.0'         │
│   11    │             'user2'              │ '0xBcd4042DE499D14e55001CcbB24a551F3b954096' │
'10000.0'         │
│   12    │             'user3'              │ '0x71bE63f3384f5fb98995898A86B02Fb2426c5788' │
'10000.0'         │
│   13    │            'aclAdmin'            │ '0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266' │
'9999.758524004895093496' │
│   14    │         'emergencyAdmin'         │ '0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266' │
'9999.758524004895093496' │
│   15    │           'poolAdmin'            │ '0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266' │
'9999.758524004895093496' │
│   16    │ 'addressesProviderRegistryOwner' │ '0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266' │
```

```
'9999.758524004895093496' │
│   17    │       'treasuryProxyAdmin'       │ '0x70997970C51812dc3A010C7d01b50e0d17dc79C8' │
'9999.998285555454949189' │
│   18    │      'incentivesProxyAdmin'      │ '0x70997970C51812dc3A010C7d01b50e0d17dc79C8' │
'9999.998285555454949189' │
│   19    │   'incentivesEmissionManager'    │ '0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266' │
'9999.758524004895093496' │
│   20    │     'incentivesRewardsVault'     │ '0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266' │
'9999.758524004895093496' │
└─────────┴──────────────────────────────────┴──────────────────────────────────────────────┴──────────
──────────────┘
deploying "PoolAddressesProviderRegistry" (tx:
0x7ff43e65ccd3428907a9db8a001304ff852d528aaec19e6d1595262c74db7061)...: deployed at
0xeF31027350Be2c7439C1b0BE022d49421488b72C with 799740 gas
[Deployment] Transferred ownership of PoolAddressesProviderRegistry to:
0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
deploying "SupplyLogic" (tx: 0x3484dd81e95d9593bb941e65074190fcc57c8102989de29241d2e03ecaa88b50)...:
deployed at 0xaC47e91215fb80462139756f43438402998E4A3a with 3291796 gas
deploying "BorrowLogic" (tx: 0x3f2411259ee70f9cfc02a08c54894e43351ad7f1379f5e59520c11c691a1de6f)...:
deployed at 0x9BcC604D4381C5b0Ad12Ff3Bf32bEdE063416BC7 with 4932060 gas
deploying "LiquidationLogic" (tx: 0x53d583e36a7f920eb7404a4991289f6bf36fa2a5e8b22b6c1ae29c2c06704476)...:
deployed at 0x63fea6E447F120B8Faf85B53cdaD8348e645D80E with 3444037 gas
deploying "EModeLogic" (tx: 0xe17980834d40ca9bed6daf3998c3394700aac7e3dc31c828c21550bbffbfbf43)...:
deployed at 0xdFdE6B33f13de2CA1A75A6F7169f50541B14f75b with 1174900 gas
deploying "BridgeLogic" (tx: 0x79da8c9eb88cff017f8cfd71d26d1ace50b96d9ae54bc6de32a01598d7bd83e9)...:
deployed at 0xaC9fCBA56E42d5960f813B9D0387F3D3bC003338 with 1833276 gas
deploying "ConfiguratorLogic" (tx:
0x70435006e2a80a8f39b3e085bec61a80f6320135882615812444d711d4f419be)...: deployed at
0x38A70c040CA5F5439ad52d0e821063b0EC0B52b6 with 1942543 gas
deploying "FlashLoanLogic" (tx: 0x78a911c212caef01902c7375f49805c662fd9f9fcbe4304d7c480c7c2891a17f)...:
deployed at 0x54B8d8E2455946f2A5B8982283f2359812e815ce with 2429673 gas
deploying "PoolLogic" (tx: 0xbd70a1c7ac833053da662d3661d3d7e66f7baf9ceaf5c44dded2128cebbecd89)...:
deployed at 0xf090f16dEc8b6D24082Edd25B1C8D26f2bC86128 with 2138665 gas
deploying "Treasury-Controller" (tx:
0xabb0e61a86cb59c35a19ae21092bc24b508d04fc0bd79909c60bd8b47e3a3dcd)...: deployed at
0x56D13Eb21a625EdA8438F55DF2C31dC3632034f5 with 701912 gas
deploying "Treasury-Implementation" (tx:
0x1fbd95fb7a05be4b8479390500e5561c88c7ec5802f4529d87ec07d7095eb07a)...: deployed at
0xE8addD62feD354203d079926a8e563BC1A7FE81e with 2116786 gas
[WARNING] Using deployed Testnet tokens instead of ReserveAssets from configuration file
{
  USDC: '0xaB7B4c595d3cE8C85e16DA86630f2fc223B05057',
  WBTC: '0xB06c856C8eaBd1d8321b687E188204C1018BC4E5',
  WETH: '0xddE78e6202518FF4936b5302cC2891ec180E8bFf'
}
deploying "PoolAddressesProvider-Test" (tx:
0xb8f0e1f9173845c3c2ed5e2aca6efcbc4042612431ce67fcc0408f448b70601c)...: deployed at
0xe70f935c32dA4dB13e7876795f1e175465e6458e with 2235259 gas
Added LendingPoolAddressesProvider with address "0xe70f935c32dA4dB13e7876795f1e175465e6458e" to registry
located at 0xeF31027350Be2c7439C1b0BE022d49421488b72C
deploying "PoolDataProvider-Test" (tx:
0x516f8ab2ee56137bee7c5e2e5118d118986aeedecd204995d9a8ef3a5b5b9826)...: deployed at
0x3904b8f5b0F49cD206b7d5AABeE5D1F37eE15D8d with 2695418 gas
[WARNING] Using deployed Testnet tokens instead of ReserveAssets from configuration file
deploying "USDC-TestnetPriceAggregator-Test" (tx:
0x6c9da21ecab94144b1569c8148a14b8b262c3817721a328af559807a6e96ae00)...: deployed at
0x56fC17a65ccFEC6B7ad0aDe9BD9416CB365B9BE8 with 114488 gas
deploying "WBTC-TestnetPriceAggregator-Test" (tx:
0xaca207638841f424bafd012cecc0fd972388eea1b01df4e9720c635ac41e79f1)...: deployed at
0x2625760C4A8e8101801D3a48eE64B2bEA42f1E96 with 114512 gas
deploying "WETH-TestnetPriceAggregator-Test" (tx:
0x5b3a7d8a7345eb55846772474dd40ac4f9234b79aebebac09069fdc3c7726113)...: deployed at
0xFE5f411481565fbF70D8D33D992C78196E014b90 with 114500 gas
deploying "Pool-Implementation" (tx:
0x93f5c0b556a5935fe7f9e7bc8c7abd86117848ab611c194510a3d4d2abc15e5e)...: deployed at
0xD6b040736e948621c5b6E0a494473c47a6113eA8 with 4714200 gas
Initialized Pool Implementation
[INFO] Skipped L2 Pool due current network 'hardhat' is not supported
deploying "PoolConfigurator-Implementation" (tx:
0x1df94754200ce9d6e2f1fcef7caba2b39f3b4fb598550eeb1016e547b0a8b1e6)...: deployed at
0xAdE429ba898c34722e722415D722A70a297cE3a2 with 5249184 gas
Initialized PoolConfigurator Implementation
```

```
deploying "ACLManager-Test" (tx: 0x1495c13aee5010fb74f167ac1c0029510e14acb5e95d685a9418dcc8356c09cb)...:
deployed at 0x87006e75a5B6bE9D1bbF61AC8Cd84f05D9140589 with 1155885 gas
== Market Admins ==
- ACL Admin 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
- Pool Admin 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
- Emergency Admin 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
[WARNING] Using deployed Testnet tokens instead of ReserveAssets from configuration file
[WARNING] Using deployed Mock Price Aggregators instead of ChainlinkAggregator from configuration file
deploying "AaveOracle-Test" (tx: 0x13c81ef31f24c86fa5db9fbfb859ed2f7a2a494edec8ef36314a7cb48e301be5)...:
deployed at 0x359570B3a0437805D0a71457D61AD26a28cAC9A2 with 884624 gas
[Deployment] Added PriceOracle 0x359570B3a0437805D0a71457D61AD26a28cAC9A2 to PoolAddressesProvider
[Deployment] Attached Pool implementation and deployed proxy contract:
- Tx hash: 0x538f79489f830106ed3c08c27bda5c9431ece3ac61ae6b877d6c1aeb2807428c
- Deployed Proxy: 0x51CfABdB442281A31eB298C44De4da9c1E7e9de6
[Deployment] Attached PoolConfigurator implementation and deployed proxy
- Tx hash: 0x3e0077b3a464273f17e59951d82f851135108964277237f2a9a0786a3fa029e4
- Deployed Proxy: 0x9C7E17872bed0dfA19697d642B782eEc63a87d9e
deploying "EmissionManager" (tx: 0x95baf694020da0c9ea4270b67b2f9cc8130125d2aa32892e5820ad845ca0d59d)...:
deployed at 0x9385556B571ab92bf6dC9a0DbD75429Dd4d56F91 with 1171030 gas
deploying "IncentivesV2-Implementation" (tx:
0x5bf31664ac4d40913c6e323d50b1957617360a2849540733506ce3fa938b8fc3)...: deployed at
0x162700d1613DfEC978032A909DE02643bC55df1A with 4061533 gas
[Deployment] Attached Rewards implementation and deployed proxy contract:
- Tx hash: 0x2e1301528e84f22f2915882fb62cfc8181db2e04ea43b74e423683f3add0f3aa
deploying "PullRewardsTransferStrategy" (tx:
0x4a92a48edf153bb483be6ae73a9c550fcb4f7452225b853f969b282060df423c)...: deployed at
0x976C214741b4657bd99DFD38a5c0E3ac5C99D903 with 416568 gas
[WARNING] Missing StkAave address. Skipping StakedTokenTransferStrategy deployment.
deploying "AToken-Test" (tx: 0xa08426a6d18cac6284d0ea08045c00561a2e4559d72460775c1f6566964790f5)...:
deployed at 0x942ED2fa862887Dc698682cc6a86355324F0f01e with 3074695 gas
deploying "DelegationAwareAToken-Test" (tx:
0x055cdbd826a891423b52c991560d5ab719e7c4f4bbccb2f27b4d979c37e83711)...: deployed at
0xcC4c41415fc68B2fBf70102742A83cDe435e0Ca7 with 3213366 gas
deploying "StableDebtToken-Test" (tx:
0x07f2696d4a71dd34f74d9dd8d343c78487233161cbdeff89ebd18e8691267eb2)...: deployed at
0xc7cDb7A2E5dDa1B7A0E792Fe1ef08ED20A6F56D4 with 2417561 gas
deploying "VariableDebtToken-Test" (tx:
0x3ac5bf606a5b99c528cff75e77900f02017b09b079dc792c39ac9f782cae390c)...: deployed at
0xe1708FA6bb2844D5384613ef0846F9Bc1e8eC55E with 2137877 gas
deploying "ReserveStrategy-rateStrategyVolatileOne" (tx:
0x5f5215129876e44c5a97c18fdd7c626eeef7bcb08fbf4b0a3e243cb40b0f75ac)...: deployed at
0x8e264821AFa98DD104eEcfcfa7FD9f8D8B320adA with 722916 gas
deploying "ReserveStrategy-rateStrategyStableOne" (tx:
0x62d7b2643cd6dfe67a605c36e89321894bfdace07c90df486d6a957a475f8de8)...: deployed at
0x871ACbEabBaf8Bed65c22ba7132beCFaBf8c27B5 with 723120 gas
deploying "ReserveStrategy-rateStrategyStableTwo" (tx:
0x94110e91be3474f40abc13ed2e68658245118ecdefdf30387c11e6d76f2a61f6)...: deployed at
0x6A59CC73e334b018C9922793d96Df84B538E6fD5 with 723120 gas
[WARNING] Using latest deployed Treasury proxy instead of ReserveFactorTreasuryAddress from configuration
file
[WARNING] Using deployed Testnet tokens instead of ReserveAssets from configuration file
Strategy address for asset WBTC: 0x8e264821AFa98DD104eEcfcfa7FD9f8D8B320adA
Strategy address for asset USDC: 0x871ACbEabBaf8Bed65c22ba7132beCFaBf8c27B5
Strategy address for asset WETH: 0x8e264821AFa98DD104eEcfcfa7FD9f8D8B320adA
- Reserves initialization in 1 txs
  - Reserve ready for: WBTC, USDC, WETH
    - Tx hash: 0x9fb339cb66e1f7f33b313239d7ad4d668645ee22037fdd50612211869a02b95b
[Deployment] Initialized all reserves
- Configure reserves in 1 txs
  - Init for: WBTC, USDC, WETH
    - Tx hash: 0x446bc9cee0e71907d96de8bfda038b1ca8f6381aaf456d116f79eb7895c9e45e
[Deployment] Configured all reserves
deploying "MockFlashLoanReceiver" (tx:
0x48d6612bf8429a207bed3f5fa7ce2a5481ed788f40a5fb92d3ac122d71eb07be)...: deployed at
0x71a0b8A2245A9770A4D887cE1E4eCc6C1d4FF28c with 650091 gas
    Initialization
      ✔ Should deploy bucket with liquidity mining is off and check initial params (86ms)
      ✔ Should deploy bucket with liquidity mining is on and check initial params (103ms)
      ✔ Should deploy bucket with initial bar calculation params (86ms)
      ✔ Should deploy bucket with initial maxTotalDeposit value (72ms)
      ✔ Should revert when liquidityMiningAmount isn't 0 and liquidityMiningRewardDistributor address not
supported (56ms)
```

```
      ✔ Should revert when liquidityMiningAmount isn't 0 and liquidityMiningDeadline is smaller current
timestamp (162ms)
      ✔ Should revert when liquidityMiningAmount isn't 0 and amountPerUser is 0 (48ms)
      ✔ Should revert when registry address not supported (135ms)
      ✔ Should revert when pToken address not supported (143ms)
      ✔ Should revert when dns address not supported (183ms)
      ✔ Should revert when debtToken address not supported (156ms)
      ✔ Should revert when positionManager address not supported (167ms)
      ✔ Should revert if Reserve address does not support IReserve (154ms)
      ✔ Should revert if InterestRateStrategy address does not support IInterestRateStrategy (179ms)
      ✔ Should revert when asset address is zero (69ms)
      ✔ Should revert when decimals of borrowed asset exceeds the max value (174ms)
      ✔ Should set withdrawalFeeRate during deploy (75ms)
      ✔ Should set the correct values of estimated Bar and Lar during deploy bucket with liquidity mining
(79ms)
    Set functions
      ✔ Should revert if not BIG_TIMELOCK_ADMIN call setReserveRate
      ✔ Should be: new reserve Rate set successfully
      ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call setFeeBuffer
      ✔ Should be: new fee buffer set successfully
      ✔ Should set maxTotal deposit successfully and emit event
      ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call setMaxTotalDeposit
      ✔ Should revert if not BIG_TIMELOCK_ADMIN call setWithdrawalFee
      ✔ Should revert if new withdrawalFeeRate mor or equal WAD/10 (10 percent) (39ms)
      ✔ Should set new withdrawalFeeRate (40ms)
      ✔ Should emit FeeBufferChanged when fee buffer is changed (40ms)
      ✔ Should revert if not BIG_TIMELOCK_ADMIN call setInterestRateStrategy
      ✔ Should set a new InterestRateStrategy address if it supports IInterestRateStrategy (237ms)
      ✔ Should revert if new InterestRateStrategy address does not support IInterestRateStrategy (190ms)
    View functions
      ✔ Should returns the correct status when the bucket is inactive (62ms)
      ✔ Should returns the correct status when the bucket is active (45ms)
      ✔ Should returns the correct status when the bucket is deprecated (63ms)
      ✔ Should returns the correct status when the current timestamp > delisting deadline (67ms)
      ✔ Should returns the correct status when the current timestamp > admin deadline (75ms)
    receiveDeposit
      ✔ Should revert receiveDeposit if DEPOSIT_EXCEEDS_MAX_TOTAL_DEPOSIT (44ms)
      ✔ Should receiveDeposit if deposit does not exceed maxTotalDeposit (84ms)
    withdrawAfterDelisting
      ✔ Should revert if not BIG_TIMELOCK_ADMIN call withdrawAfterDelisting
      ✔ Should revert withdrawAfterDelisting when the bucket status is not time after delisting
      ✔ Should withdrawAfterDelisting to treasury (169ms)
    Integration tests LiquidityMining in bucket and LiquidityMiningRewardDistributor
      ✔ Should revert openPosition while bucket is not launched (327ms)
      ✔ Should revert receiveDeposit if it's called not by bucket in system
      ✔ Should openPosition when bucket is launched (625ms)
      ✔ claimReward should transfer pmx on balance in TraderBalanceVault (345ms)
      ✔ withdrawPmxByAdmin should transfer pmx (301ms)
      deposit
        ✔ Should revert when the msg.sender is on the blacklist (143ms)
        ✔ Should revert when liquidityMiningDeadline is passed (44ms)
        ✔ Should revert when user deposit is more than amountPerUser (614ms)
        ✔ Should transfer underlying asset and mint ptokens (1386ms)
        ✔ Shouldn't revert and should launch bucket when tranfer asset in bucket without deposit (92ms)
        ✔ Should update only msg.sender info in LiquidityMiningRewardDistributor(pTokenReceiver parameter
is unusable argument in liquidity mining phase) (199ms)
        ✔ Should update trader info in LiquidityMiningRewardDistributor (3078ms)
      withdraw
        ✔ Should revert withdraw when the msg.sender is on the blacklist (140ms)
        ✔ Should correct update LM amount if withdraw before launch bucket (986ms)
        ✔ Withdraw should be success if _amount is magic number(MaxUint256) (100ms)
        ✔ Should emit RatesUpdated when withdraw from a bucket (128ms)
        ✔ Should revert withdraw if bucket is launched but stabilizationPeriodDuration isn't passed and
user withdraws its mining liquidity amount (263ms)
        ✔ Shouldn't reset to zero all user points if bucket is launched, stabilizationPeriodDuration
hasn't passed and user withdraws amount not participated in liquidity mining event (491ms)
        ✔ Shouldn't reset to zero all user points if bucket is launch and stabilizationPeriodDuration is
passed (229ms)
        ✔ Should withdraw from bucket and top up withdrawer balance and treasury balance (181ms)
        ✔ Should emit TopUpTreasury event (162ms)
      depositFromBucket and receiveDeposit integrations tests
        ✔ Should revert depositFromBucket when the msg.sender is on the blacklist (153ms)
```

```
        ✔ Should revert depositFromBucket in bucket from which this function is called (97ms)
        ✔ Should revert depositFromBucket when bucket is launched (92ms)
        ✔ Should revert depositFromBucket when bucket isn't launched and deadline isn't passed (100ms)
        ✔ Should revert depositFromBucket when bucketTo1 is bucket with other asset and swapManager
doesn't have VAULT_ACCESS_ROLE (80ms)
        ✔ depositFromBucket works after partial withdrawal from bucket (247ms)
        ✔ depositFromBucket should burn user's ptoken in this bucket and mint ptokens on user address in
receiverBucket (410ms)
        ✔ depositFromBucket should transfer underlying asset from this bucket to receiverBucket (161ms)
        ✔ depositFromBucket should swap and transfer underlying asset from this bucket to receiverBucket
if assets of buckets are different (350ms)
        ✔ depositFromBucket should add extra reward in receiverBucket (196ms)
        ✔ if receiverBucket is launched do locked deposit and immediately claim extra reward to balance
in traderBalanceVault (384ms)
    maxAssetLeverage
      ✔ Should not return maxAssetLeverage when asset doesn't exist
      ✔ Should return the correct max leverage when the asset isn't equal to the deposit asset and the
position asset (296ms)
      ✔ Should return the correct max leverage when the asset is equal to the deposit asset or position
asset (295ms)
    allowedAssetList
      ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call addAsset
      ✔ Should revert if not EMERGENCY_ADMIN call removeAsset
      ✔ Should allow to add asset only when volatility of asset relative to borrowed asset > 0
      ✔ Should add token to allowed for deals token list in constructor
      ✔ Should add few tokens to allowed for deals token list (157ms)
      ✔ removeAsset should correct update state (183ms)
      ✔ removeAsset should revert when asset isn't exist
      ✔ removeAsset should create event
      ✔ addAsset should create event
      ✔ Should not allow to add duplicate token to allowed for deals token list
      ✔ Should not allow to add token with incorrect decimals  (94ms)
    Fee collection
      ✔ Should mint correct amount of token to the reserve for Utilization Ratio 5% (569ms)
      ✔ Should mint correct amount of token to the reserve for Utilization Ratio 30% (568ms)
      ✔ Should mint correct amount of token to the reserve for Utilization Ratio 70% (554ms)
      ✔ Should return correct balances of ptoken which sum is less or equal to availableLiquidity
(2033ms)
      ✔ Should return correct balances of ptoken which sum is less than availableLiquidity when updating
indexes and rates isn't called (849ms)
    deposit
      ✔ Should revert deposit if bucket not active in dns (53ms)
      ✔ Should revert deposit if bucket not added in dns (244ms)
      ✔ Should revert deposit if bucket not in primex protocol (253ms)
      ✔ Should revert deposit if user has not approved tokens
      ✔ Should emit RatesUpdated when deposit into a launched bucket (111ms)
      ✔ Should revert when deposit is more than maxTotalDeposit (55ms)
      ✔ Should deposit when deposit amount is less than maxTotalDeposit (113ms)
    paybackPermanentLoss - unit testing
      ✔ Should revert when the msg.sender is on the blacklist (145ms)
      ✔ Should pay exact permanent loss value if input amount is greater than permanent loss value (89ms)
      ✔ Should subtract amount from permanentLoss (91ms)
      ✔ Should return correct permanentLossScaled (64ms)
    paybackPermanentLoss - integration testing
      ✔ Should revert if a param 'uint256 amount' > pToken.balanceOf(msg.sender) (1085ms)
      ✔ Should emit Burn event if a param 'uint256 amount' <= pToken.balanceOf(msg.sender) (1470ms)
      ✔ Should increase debt over time (1144ms)
    Integration tests deposit liquidity to Aave
      ✔ Should deposit to Aave when isInvestEnabled = true and emit DepositToAave event (136ms)
      ✔ Should deposit to Aave user tokens together with directly sent tokens (142ms)
      ✔ Should withdraw from Aave during liquidity mining and emit withdrawFromAave event (314ms)
      ✔ Should withdraw from Aave during liquidity mining using _amount is magic number(MaxUint256)
(254ms)
      ✔ Should withdraw all liquidity from Aave when liquidity mining is failed (297ms)
      ✔ Should withdraw all liquidity from Aave when liquidity mining is failed and user
depositFromBucket (983ms)
      ✔ Should withdraw all liquidity from Aave when bucket is launched (280ms)
      ✔ Should transfer earned interest to treasury when bucket is launched and emit TopUpTreasury event
(570ms)
      ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call withdrawByAdminFromAave
      ✔ Should withdrawByAdminFromAave when tokens left in Aave pool (237ms)
```

```
  BucketsFactory
    ✔ constructor
    ✔ should createBucket if right access (172ms)
    ✔ should revert if not MEDIUM_TIMELOCK_ADMIN call createBucket (56ms)
    ✔ should revert createBucket if PositionManager address does not support IPositionManager (133ms)
    ✔ should revert createBucket if Reserve address does not support IReserve (136ms)
    ✔ emits BucketCreated if a bucket is created (167ms)
    ✔ should revert when trying to access a non-existent bucket (167ms)
    ✔ should return appropriate length of allBuckets[] if several buckets were created (489ms)
    ✔ should create a Bucket contract code (165ms)
Warning: Potentially unsafe deployment of contracts/mocks/upgradeMocks/BucketV2.sol:BucketV2

    You are using the `unsafeAllow.external-library-linking` flag to include external libraries.
    Make sure you have manually checked that the linked libraries are upgrade safe.

    ✔ should upgrade Bucket implementation (482ms)
    setPTokensFactory
      ✔ Should set pTokens factory (45ms)
      ✔ Should revert when new pTokensFactory address not supported (131ms)
      ✔ Should revert if not BIG_TIMELOCK_ADMIN call setPTokensFactory
    setDebtTokensFactory
      ✔ Should set debtTokens factory (44ms)
      ✔ Should revert when new debtTokensFactory address not supported (125ms)
      ✔ Should revert if not BIG_TIMELOCK_ADMIN call setDebtTokensFactory (39ms)

  ChainLinkUpKeep
deploying "LinkToken" (tx: 0x1755646c28ed1d36fb1f54b9b6c5a081c7c9404b3651bc1593ce2ad411935c41)...:
deployed at 0x2B07F89c9F574a890F5B8b7FddAfbBaE40f6Fde2 with 756180 gas
deploying "PrimexAggregatorV3TestService linkEthFeed price feed" (tx:
0x2855a98c835adc62fd425101f2a3303bafffb76a5d2598877df8b3ec86080941)...: deployed at
0xCaC60200c1Cb424f2C1e438c7Ee1B98d487f0254 with 870238 gas
deploying "PrimexAggregatorV3TestService fastGasFeed price feed" (tx:
0x015b5f536f8d42c9d73a29e65e343407d2b23e5823d10db648f1e2386d50dbb6)...: deployed at
0xABc84968376556B5e5B3C3bda750D091a06De536 with 870238 gas
deploying "KeeperRegistry" (tx: 0x3475a412ccc7599af0c51eb6b976877d2142d98247c7914e6cfd8ee2afb30255)...:
deployed at 0xFf8FA9381caf61cB3368a6ec0b3F5C788028D0Cd with 3507221 gas
deploying "Counter" (tx: 0xca7fc602ea193c0bb48e305be730a56aa79e1bd2229d18008c5ca1c694d89387)...: deployed
at 0xE55cc27460B55c8aC7E73043F38b537758C9E51e with 228036 gas
    ✔ keeper's flow (227ms)

  CrossDexClosePosition
    closePosition
      ✔ Shouldn't close position and throw revert if called by the NON-owner (424ms)
      ✔ Should revert close position if the dex is frozen in PrimexDNS, but this dex was NOT the dex of
opening a position (111ms)
      ✔ Should close position and transfer testTokenB from 'PositionManager' to 'Pair' (293ms)
      ✔ Should close position and transfer testTokenA from 'Pair' (342ms)
      ✔ Should close position and delete trader position from traderPositions list (227ms)
      ✔ Should close position and fully repay traders debt (261ms)
      ✔ Should close position and fully repay traders debt after 1 block past (264ms)
      ✔ Should close position and fully repay traders debt after 10 blocks past (268ms)
      ✔ Should close position 1 block past and transfer increased full amount (principal + fees) of
testTokenA to 'Bucket' (264ms)
      ✔ Should close position 1 block past and rest of trader deposit to traderBalanceVault when deal is
loss (361ms)
      ✔ Should close position 1 block past and transfer trader profit from PositionManager to
DepositVault when deal is profit (552ms)
      ✔ Should close position 1 block past and repay to bucket when deal is profit (860ms)
      ✔ Should close position 1 block after and add amount to available balance in TraderBalanceVault
(371ms)
      ✔ Should close position and throw event (713ms)
    liquidatePosition
      ✔ Should liquidate risky position on dex1 if the position is risky (789ms)
      ✔ Should liquidate risky position on dex2 if the position is risky (706ms)

  CurveBotLens
    removeAndSetLiquidity
      ✔ should have less lp tokens and more assets after reducing liquidity (410ms)
      ✔ should have more lp tokens and less assets after increasing liquidity (469ms)
      ✔ should add liquidity without initial liquidity (427ms)
      ✔ should recalculate amounts if cannot remove necessary liquidity (909ms)
```

```
  DebtToken
    initialization
      ✔ Should initialize with correct values. (64ms)
    Transfers and allowances
      ✔ Should revert when the increaseAllowance func is called
      ✔ Should revert when the decreaseAllowance func is called
    SetBucket
      ✔ Should revert when bucket already set
      ✔ Should revert when a param 'IBucket _bucket' does not support IBucket (193ms)
    setFeeDecreaser
      ✔ Should revert if not BIG_TIMELOCK_ADMIN call setFeeDecreaser
      ✔ Should revert when the address does not support IBonusExecutor
      ✔ Should set the FeeDecreaser (53ms)
      ✔ Should set zero address
    setTraderRewardDistributor
      ✔ Should revert if not BIG_TIMELOCK_ADMIN call setTraderRewardDistributor
      ✔ Should revert when the address does not support IBonusExecutor
      ✔ Should set the TraderRewardDitributor (135ms)
      ✔ Should set zero address
    Not supported
      ✔ Should revert when transfer
      ✔ Should revert when approve
      ✔ Should revert when transferFrom
      ✔ Should revert when increaseAllowance
      ✔ Should revert when decreaseAllowance
    Access
      ✔ Should revert mint when caller is not bucket
      ✔ Should revert burn when caller is not bucket
    Mint & Burn
      ✔ Should Mint
      ✔ Should revert mint when a param 'address _user' is zero
      ✔ Should revert mint when amount is 0
      ✔ Should revert mint when invalid mint amount
      ✔ Should Burn (40ms)
      ✔ Should revert Burn when a param 'address _user' is zero
      ✔ Should revert Burn when amount is 0
      ✔ Should revert Burn when invalid mint amount

  DepositAsset_isAssetNotFromSwapPair
    PositionManager
      openPosition
        ✔ Should revert open position by order when token (testTokenX) not allowed (60ms)
        ✔ Should revert open position when the amount of tokens received is smaller amountOutMin (606ms)
        ✔ Should revert when depositInThirdAssetRoutes sum of shares is 0 (430ms)
        ✔ Should be revert when the dex price is less than the oracle price by
DefaultOracleTolerableLimit + 5%(oracle tokenA-tokenB) (477ms)
        ✔ Should be revert when the dex price is less than the oracle price by oracleTolerableLimit + 5%
(oracle tokenA-tokenB) (486ms)
        ✔ Should be revert when the dex price is less than the oracle price by
DefaultOracleTolerableLimit + 5%(oracle tokenX-tokenB) (522ms)
        ✔ Should be revert when the dex price is less than the oracle price by oracleTolerableLimit + 5%
(oracle tokenX-tokenB) (513ms)
        ✔ Should revert createPosition when isProtocolFeeInPmx=true and trader doesn't have enough
protocolFee assets (pmx) on traderBalanceVault (494ms)
        ✔ Should open position when the amount of tokens received is equal or more amountOutMin (1675ms)
        ✔ Should be openPositionByOrder by oracle price if dex price is more than the oracle price by
DefaultOracleTolerableLimit + 5%(oracle tokenX-tokenA) (600ms)
        ✔ Should be openPositionByOrder by oracle price if dex price is more than the oracle price by
oracleTolerableLimit + 5%(oracle tokenX-tokenA) (583ms)
        ✔ Should create 'Position' and transfer testTokenA from 'Bucket' to 'Pair' (512ms)
        ✔ Should createPosition when isProtocolFeeInPmx=false (571ms)
        ✔ Should create 'Position' and transfer testTokenX when isProtocolFeeInPmx=true (928ms)
        ✔ Should create position and increase traders count, and add traderPositions (822ms)
        ✔ Should open position and throw event (657ms)
        ✔ Should open position when position size >= minPositionSize (612ms)
        ✔ Should revert when position size < minPositionSize (226ms)
      openPosition with deposit
        ✔ Should be revert when the dex price is less than the oracle price by
DefaultOracleTolerableLimit + 5%(oracle tokenA-tokenB) (505ms)
        ✔ Should be revert when the dex price is less than the oracle price by oracleTolerableLimit + 5%
(oracle tokenA-tokenB) (468ms)
        ✔ Should revert when deposit Amount insufficient for deal (662ms)
```

```
        ✔ Should be openPositionByOrder by oracle price if dex price is more than the oracle price by
DefaultOracleTolerableLimit + 5%(oracle tokenX-tokenA) (583ms)
        ✔ Should be openPositionByOrder by oracle price if dex price is more than the oracle price by
oracleTolerableLimit + 5%(oracle tokenX-tokenA) (582ms)
        ✔ Should be revert when the dex price is less than the oracle price by
DefaultOracleTolerableLimit + 5%(oracle tokenX-tokenB) (332ms)
        ✔ Should be revert when the dex price is less than the oracle price by oracleTolerableLimit + 5%
(oracle tokenX-tokenB) (344ms)
        ✔ Should revert open position when the amount of tokens received is smaller amountOutMin (582ms)
        ✔ Should revert createPosition when isProtocolFeeInPmx=true and trader doesn't have enough
protocolFee assets (pmx) on traderBalanceVault (145ms)
        ✔ Should open position when the amount of tokens received is equal or more amountOutMin (1399ms)
        ✔ Should create 'Position' and transfer testTokenX (605ms)
        ✔ Should createPosition when isProtocolFeeInPmx=true (880ms)
      closePosition
        ✔ Shouldn't close position and throw revert if called by the NON-owner (917ms)
        ✔ Should be revert when the dex price is less than the oracle price by oracleTolerableLimit + 5%
(238ms)
        ✔ Should close position and transfer testTokenB from 'PositionManager' to 'Pair' (305ms)
        ✔ Should close position and transfer testTokenA rest of trader deposit from 'Pair' (736ms)
        ✔ Should close position and delete trader position from traderPositions list (267ms)
        ✔ Should close position and fully repay traders debt (290ms)
        ✔ Should close position and fully repay traders debt after 1 block past (329ms)
        ✔ Should close position and fully repay traders debt after 10 blocks past (324ms)
        ✔ Should close position 1 block past and transfer increased full amount (principal + fees) of
testTokenA to 'Bucket' (369ms)
        ✔ Should close position 1 block past and transfer trader depositAfterDeal from PositionManager to
TraderBalanceVault when deal is loss (438ms)
        ✔ Should close position 1 block past and transfer trader profit from PositionManager to
TraderBalanceVault when deal is profit (681ms)
        ✔ Should close position 1 block past and repay to bucket when deal is profit (584ms)
        ✔ Should close position 1 block after and add amount to available balance in TraderBalanceVault
(502ms)
        ✔ Should close position and throw event (891ms)
      liquidatePosition
        ✔ Should be revert when the dex price is less than the oracle price by oracleTolerableLimit + 5%
(545ms)
        ✔ Shouldn't liquidate position until it not risky (732ms)
        ✔ Should liquidate risky position and transfer testTokenB from 'PositionManager' to 'Pair' when
health position is equal to ~ 0.91wad (1074ms)
        ✔ Should liquidate risky position and transfer testTokenB from 'PositionManager' to 'Pair' when
health position is equal to ~ 0.99wad (963ms)
        ✔ Should liquidate risky position and transfer testTokenA from 'Pair' (1047ms)
        ✔ Should liquidate risky position and delete trader position from traderPositions list (737ms)
        ✔ Should liquidate risky position and fully repay traders debt (880ms)
        ✔ Should liquidate risky position and fully repay traders debt after 3 blocks past (791ms)
        ✔ Should liquidate risky position and fully delete trader's deposit from 'TraderBalanceVault'
(854ms)
        ✔ Should liquidate risky position and throw event (713ms)
        ✔ Should liquidate risky position and transfer rest of trader deposit to treasury (1070ms)
        ✔ Should liquidate risky position 1 block past and transfer positionDebt (principal + fees) of
testTokenA to 'Bucket' (760ms)
      increaseDeposit
        ✔ Should set amountOut from dex even if oracle amountOut is less than dex amount out (403ms)
    LimitOrderManager
      openPositionByOrder
        ✔ Should be revert when the dex price is less than the oracle price by
DefaultOracleTolerableLimit + 5%(oracle tokenA-tokenB) (552ms)
        ✔ Should be revert when the dex price is less than the oracle price by oracleTolerableLimit + 5%
(oracle tokenA-tokenB) (678ms)
        ✔ Should revert when firstAssetRoutes summ of shares is 0 (473ms)
        ✔ Should revert when depositInThirdAssetRoutes summ of shares is 0 (362ms)
        ✔ Should be revert when the dex price is less than the oracle price by
DefaultOracleTolerableLimit + 5%(oracle tokenX-tokenB) (761ms)
        ✔ Should be revert when the dex price is less than the oracle price by oracleTolerableLimit + 5%
(oracle tokenX-tokenB) (498ms)
        ✔ Should be openPositionByOrder by oracle price if dex price is more than the oracle price by
DefaultOracleTolerableLimit + 5%(oracle tokenX-tokenA) (818ms)
        ✔ Should be openPositionByOrder by oracle price if dex price is more than the oracle price by
oracleTolerableLimit + 5%(oracle tokenX-tokenA) (798ms)
        ✔ Should create position by order when stopLoss=0, takeProfit=0 (966ms)
        ✔ Should create position by order and transfer testTokenA from 'Bucket' to 'Pair' (831ms)
```

```
        ✔ Should create position by order when dex amountOut < oracle amountOut, increase traders count,
add traderPositions and then delete the order (1077ms)
        ✔ Should create position by order when dex amountOut > oracle amountOut, increase traders count,
add traderPositions and then delete the order (967ms)
        ✔ Should open position by order and do not lock trader deposit amount from dex in
traderBalanceVault (791ms)
        ✔ Should open position when isProtocolFeeInPmx=true (860ms)
        ✔ Should open position by order and do not lock trader deposit amount traderBalanceVault (701ms)
      canBeFilled
        ✔ Should revert when depositAsset is third asset and firstAssetRoutes length is empty (1057ms)
        ✔ Should revert when depositAsset is third asset and depositInThirdAssetRoutes length is empty
(364ms)
        ✔ Should return true when limitPrice is more than current price on dex and trader has enough pmx
on traderBalanceVault (974ms)
        ✔ Should return true when limitPrice is current price on dex  (1432ms)
        ✔ Should return false when limitPrice is less than current price on dex  (753ms)
        ✔ Should return false when limitPrice > current price(10) but deadline < block.timestamp

  DepositAsset_isPositionAsset
    PositionManager
      openPosition
        ✔ Should revert open position when not allowed token (testTokenX) (137ms)
        ✔ Should revert when firstAssetRoutes is empty list (430ms)
        ✔ Should revert open position when the amount of tokens received is smaller amountOutMin (433ms)
        ✔ Should open position when the amount of tokens received is equal or more amountOutMin (1056ms)
        ✔ Should revert when the dex price is less than the oracle price by oracleTolerableLimit +5%
(335ms)
        ✔ Should create 'Position' and transfer testTokenA from 'Bucket' to 'Pair' (472ms)
        ✔ Should create 'Position' and transfer testTokenB from traderBalanceVault to positionManager
(513ms)
        ✔ Should create 'Position' with isProtocolFeeInPmx=true (529ms)
        ✔ Should create position and increase traders count, and add traderPositions (629ms)
        ✔ Should open position and throw event (578ms)
        ✔ Should open position on multiple dexes (771ms)
        ✔ Should open position when position size >= minPositionSize (471ms)
        ✔ Should revert when position size < minPositionSize (159ms)
      openPosition with deposit
        ✔ Should revert open position when the amount of tokens received is smaller amountOutMin (411ms)
        ✔ Should revert when deposit Amount on dex insufficient for deal (378ms)
        ✔ Should be revert when the dex price is less than the oracle price by oracleTolerableLimit +5%
(320ms)
        ✔ Should revert openPosition with isProtocolFeeInPmx=true and makeDeposit = false if trader
doesn't have enough pmx in traderBalanceVault (130ms)
        ✔ Should open position when the amount of tokens received is equal or more amountOutMin (1003ms)
        ✔ Should not lock tokens in traderBalanceVault as a collateral for deal (491ms)
        ✔ Should create 'Position' with isProtocolFeeInPmx=true (568ms)
      closePosition
        ✔ Shouldn't close position and throw revert if called by the NON-owner (172ms)
        ✔ Should be revert when the dex price is less than the oracle price by oracleTolerableLimit + 5%
(201ms)
        ✔ Should close position and transfer testTokenB from 'PositionManager' to 'Pair' (355ms)
        ✔ Should close position and transfer testTokenA from 'Pair' (369ms)
        ✔ Should close position and delete trader position from traderPositions list (257ms)
        ✔ Should close position and fully repay traders debt (290ms)
        ✔ Should close position and fully repay traders debt after n block past (305ms)
        ✔ Should close position 1 block past and transfer increased full amount (principal + fees) of
testTokenA to 'Bucket' (295ms)
        ✔ Should close position 1 block past and rest of trader deposit to traderBalanceVault when deal
is loss (441ms)
        ✔ Should close position 1 block past and transfer trader profit from PositionManager to
TraderBalanceVault when deal is profit (901ms)
        ✔ Should close position 1 block past and repay to bucket when deal is profit (489ms)
        ✔ Should close position 1 block after and add amount to available balance in TraderBalanceVault
(378ms)
        ✔ Should close position and throw event (402ms)
        ✔ Should close position on multiple dexes (631ms)
      liquidatePosition
        ✔ Should revert when the dex price is less than the oracle price by oracleTolerableLimit +5%
(518ms)
        ✔ Shouldn't liquidate position until it not risky (328ms)
        ✔ Should liquidate risky position and transfer testTokenB from 'PositionManager' to 'Pair' when
health position is equal to ~ 0.91wad (870ms)
```

```
        ✔ Should liquidate risky position and transfer testTokenB from 'PositionManager' to 'Pair' when
health position is equal to ~ 0.99wad (827ms)
        ✔ Should liquidate risky position and transfer testTokenA from 'Pair' (799ms)
        ✔ Should liquidate risky position and delete trader position from traderPositions list (661ms)
        ✔ Should liquidate risky position and fully repay traders debt (677ms)
        ✔ Should liquidate risky position and throw event (1246ms)
        ✔ Should liquidate risky position and transfer to Treasury rest of trader deposit (827ms)
        ✔ Should liquidate risky position 1 block past and transfer positionDebt (principal + fees) of
testTokenA to 'Bucket' (1100ms)
        ✔ Should liquidate risky position on multiple dexes (1633ms)
    LimitOrderManager
      openPositionByOrder
        ✔ Should revert when the dex price is less than the oracle price by oracleTolerableLimit +5%
(364ms)
        ✔ Should create position by order when stopLoss=0, takeProfit=0 (785ms)
        ✔ Should create position by order and transfer testTokenA from 'Bucket' to 'Pair' (605ms)
        ✔ Should create position by order, increase traders count, add traderPositions and then deleted
the order (1017ms)
        ✔ Should open position by order and top-up Treasury balance by fee amount in eth (586ms)
        ✔ Should create 'Position' with isProtocolFeeInPmx=true (673ms)
        ✔ Should create position by order on multiple dexes, increase traders count, add traderPositions
and then deleted the order (822ms)
      canBeFilled
        ✔ Should revert when depositAsset is positionAsset and depositInThirdAssetRoutes length isn't 0
(374ms)
        ✔ Should revert when depositAsset is positionAsset and firstAssetRoutes length is empty (341ms)
        ✔ Should return true when limitPrice is more than current price on dex and there's enough pmx in
trader traderBalanceVault (917ms)
        ✔ Should return true when limitPrice is current price on dex  (750ms)
        ✔ Should return false when limitPrice is less than current price on dex  (584ms)
        ✔ Should return false when limitPrice > current price(10) but deadline < block.timestamp

  DexAdapter
    ✔ Storage
    Constructor
      ✔ Should deploy dexAdapter
      ✔ Should revert when a param 'address registry' is not supported (140ms)
    Set function
      ✔ Should revert setDexType() when a param 'address _dexRouter' is zero
      ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call setDexType
      ✔ Should setDexType
      ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call setQuoter
      ✔ Should revert setQuoter() when the dexType of the dexRouter is none
      ✔ Should revert setQuoter() when the quoter is zero
      ✔ Should setQuoter() when the dexType of the dexRouter is not none and the quoter is not zero
    SwapExactTokensForTokens
      ✔ Should swapExactTokensForTokens (71ms)
      ✔ Should revert when deadline passed for _swapWithCurve (47ms)
      ✔ Should revert swapExactTokensForTokens() when a param 'address _params.to' is zero
      ✔ Should revert swapExactTokensForTokens() when a param 'address _params.dexRouter' is zero
      ✔ Should revert swapExactTokensForTokens() when amount in is zero
      ✔ Should revert swapExactTokensForTokens() when dex unknown
    GetAmountsOut
      ✔ Should getAmountsOut
      ✔ Should revert getAmountsOut() when a param 'address _params.dexRouter' is zero
      ✔ Should revert getAmountsOut() when 0 amount in
      ✔ Should revert getAmountsOut() when unknown dex type
      ✔ Should revert getAmountsOut() when Uniswap v3 is added to the dexType but uniV3Quoter is not set
(96ms)
    getAmountIn
      ✔ Should revert getAmountsIn() when a param 'address _params.dexRouter' is zero
      ✔ Should revert getAmountsIn() when 0 amount in
      ✔ Should revert getAmountsIn() when Uniswap v3 is added to the dexType but uniV3Quoter is not set
(67ms)
      ✔ getAmountsIn on curve testTokenA_curve to TestTokenX (11611ms)
      ✔ getAmountsIn on curve TestTokenC to testTokenA_curve (13055ms)
      ✔ getAmountsIn on curve TestTokenX to TestTokenC (6655ms)
      ✔ getAmountsIn on curve TestTokenX to testTokenA_curve (11620ms)
      ✔ getAmountsIn on curve testTokenA_curve to TestTokenC (9073ms)
      ✔ getAmountsIn on curve TestTokenC to TestTokenX (11932ms)
    Should getGas
      ✔ Should getGas (54ms)
```

```
    ✔ Should revert getGas when dexRouter is zero
    ✔ Should revert getGas when dexRouter unknown type

  DexAdapter swap by path
    Uniswap V2
      ✔ Should swapExactTokensForTokens by path (194ms)
      ✔ Should getAmountsOut by path (136ms)
      ✔ Should getAmountsIn by path (137ms)
    Uniswap V3
      ✔ Should swapExactTokensForTokens by path (274ms)
      ✔ Should getAmountsOut by path (348ms)
      ✔ Should getAmountsIn by path (356ms)
    Curve
      ✔ Should swapExactTokensForTokens by path (917ms)
      ✔ Should getAmountsOut by path (632ms)
      ✔ Should getAmountsIn by path (30804ms)
    Balancer
      ✔ Should swapExactTokensForTokens by path (286ms)
      ✔ Should getAmountsOut by path (273ms)
      ✔ Should getAmountsIn by path (282ms)
    Meshswap
      ✔ Should swapExactTokensForTokens by path (329ms)
      ✔ Should getAmountsOut by path (186ms)
      ✔ Should getAmountsIn by path (181ms)
    Quickswap V3
      ✔ Should swapExactTokensForTokens by path (669ms)
      ✔ Should getAmountsOut by path (465ms)
      ✔ Should getAmountsIn by path (313ms)

  LimitOrderManager
Warning: All subsequent Upgrades warnings will be silenced.

    Make sure you have manually checked all uses of unsafe flags.

    initialize
      ✔ Should initialize with correct values (38ms)
      ✔ Should revert when initialized with wrong primexDNS address (198ms)
      ✔ Should revert when initialized with wrong registry address (206ms)
      ✔ Should revert when initialized with wrong positionManager address (87ms)
      ✔ Should revert when initialized with wrong traderBalanceVault address (88ms)
      ✔ Should revert when initialized with wrong swapManager address (90ms)
    Limit Order
      CreateLimitOrder
        ✔ Should revert when the limitOrderManager is paused
        ✔ Should revert when the msg.sender is on the blacklist
        ✔ Should revert when create limit order with not allowed token (testTokenX) (49ms)
        ✔ Should revert when the fee amount is insufficient (132ms)
        ✔ Should revert when the fee amount is insufficient and make deposit is true (122ms)
        ✔ Should revert when create limit order when leverage >= maxLeverage of the bucket (111ms)
        ✔ Should create 'LimitOrder' and transfer testTokenA from trader to 'TraderBalanceVault' (309ms)
        ✔ Should create 'LimitOrder' and return the change (255ms)
        ✔ Should create 'LimitOrder' and transfer the fee amount from trader to 'TraderBalanceVault' and
lock it (244ms)
        ✔ Should create 'LimitOrder' when makeDeposit is false and lock the fee amount on
'TraderBalanceVault' (253ms)
        ✔ Should create 'LimitOrder' when makeDeposit is false, isProtocolFeeInPmx is true and lock the
fee amount on 'TraderBalanceVault' (259ms)
        ✔ Should create 'LimitOrder' when isProtocolFeeInPmx is true and transfer the fee amount from
trader to 'TraderBalanceVault' and lock it (271ms)
        ✔ Should create 'LimitOrder' with the correct variables (326ms)
        ✔ Should create 'LimitOrder' with the correct variables when isProtocolFeeInPmx is true (263ms)
        ✔ Should open limit order and throw event (302ms)
        ✔ Should create limit order with sl price > liquidationPrice (sl amount >= liquidation amount)
(257ms)
        ✔ Should revert when created with no open conditions (192ms)
        ✔ Should revert when created with shouldOpenPosition=false and not empty close conditions (160ms)
        ✔ Should revert when openingManagerAddresses has duplicates (207ms)
        ✔ Should revert when openingManagerAddresses is not COM (202ms)
        ✔ Should revert when closingManagerAddresses is not CCM (190ms)
      CreateLimitOrder with minPositionSize
        ✔ Should revert when position size < minPositionSize (129ms)
        ✔ Should create limit order when position size >= minPositionSize (244ms)
```

```
      ✔ Should return false in canBeFilled when position size < minPositionSize (346ms)
      ✔ Should revert openPositionByOrder when position size < minPositionSize (396ms)
    cancelLimitOrder
      ✔ Should revert when the msg.sender is on the blacklist
      ✔ Should revert when order does not exist
      ✔ Should revert when caller is not the trader
      ✔ Should cancel the order and throw correct event (333ms)
      ✔ Should cancel the order and unlock trader deposit in traderBalanceVault (328ms)
      cancelLimitOrdersByAdmin
        ✔ Shouldn't revert cancelLimitOrderByAdmin when order does not exist
        ✔ Should revert if not SMALL_TIMELOCK_ADMIN call cancelLimitOrdersByAdmin
        ✔ Should revert cancelLimitOrdersByAdmin when the order is spot (187ms)
        ✔ Should revert cancelLimitOrdersByAdmin when current timestamp < adminDeadline
        ✔ Should cancelLimitOrdersByAdmin when now is the time after delisting (468ms)
    openPositionByOrder
      ✔ Should revert when the msg.sender is on the blacklist
      ✔ Should revert when the limitOrderManager is paused
      ✔ Should revert when the order does not exist
      ✔ Should revert when the bucket is not active (182ms)
      ✔ Should revert openPositionByOrder when positionAsset isn't allowed (54ms)
      ✔ Should revert openPositionByOrder when firstAssetRoutes is empty list (262ms)
      ✔ Should revert openPositionByOrder when depositInThirdAssetRoutes is not empty list and
depositedAsset is borrowedAsset (299ms)
      ✔ Should revert when the order price isn't reached (823ms)
      ✔ Should revert when conditionIndex index is out of bounds (50ms)
      ✔ Should revert when keeper address is zero (102ms)
      ✔ Should create position by order with stopLoss=0 takeProfit=0 (1347ms)
      ✔ Should canBeFilled return false when positionAsset isn't allowed (215ms)
      ✔ Should canBeFilled return false when bucket is frozen (213ms)
      ✔ Should create position by order and transfer testTokenA from 'Bucket' to 'Pair' (777ms)
      ✔ Should revert openPositionByOrder if POSITION_SIZE_EXCEEDED (431ms)
      ✔ Should create position by order, increase traders count, add traderPositions and then deleted
the order (840ms)
      ✔ Should open position by order and throw event 'OpenPosition' (846ms)
      ✔ Should open position by order and throw event 'CloseLimitOrder' (640ms)
      ✔ Should open position by order and lock trader deposit in traderBalanceVault and receive fee to
treasury (615ms)
      ✔ Should open position by order with isProtocolFeeInPmx (879ms)
      ✔ Should open position by order with isProtocolFeeInPmx when the pmx token has been changed
(1230ms)
    updateOrder
      ✔ Should revert updateOrder when the msg.sender is on the blacklist (38ms)
      ✔ Should revert updateOrder when the bucket is not active (49ms)
      ✔ Should revert when update shouldOpenPosition in margin limit order (47ms)
      ✔ Should revert when caller is not trader
      ✔ Should revert when leverage < 1 (74ms)
      ✔ Should revert when leverage > maxLeverage (159ms)
      ✔ Should revert when convert margin order to spot order (74ms)
      ✔ Should revert when convert spot order to margin order (288ms)
      ✔ Should revert when depositedAmount * leverage < minPositionSize (137ms)
      ✔ Should increase leverage from wallet and transfer fee amount from trader to the
TraderBalanceVault and lock it (702ms)
      ✔ Should increase leverage from traderBalanceVault and increase locked fee amount (265ms)
      ✔ Should decrease leverage and unlock the excess fee (158ms)
      ✔ Should decrease depositedAmount and unlock the excess fee (157ms)
      ✔ Should increase depositedAmount from wallet and increase locked fee amount (972ms)
      ✔ Should increase depositedAmount from trader balance vault and increase locked fee amount
(231ms)
      ✔ Should change depositedAsset (198ms)
      ✔ Should change the protocolFeeAsset, unlock the old fee asset and lock new one (197ms)
      ✔ Should change the protocolFeeAsset from EPMX to PMX, unlock the EPMX fee asset and lock PMX
(1174ms)
      ✔ Should emit event after order update (194ms)
  updateOrderConditions
    ✔ Should revert when caller is on the blacklist
    ✔ Should revert when caller is not trader
    ✔ Should revert updateOrderConditions when the bucket is not active (59ms)
    ✔ Should revert when updated with no open conditions (63ms)
    ✔ Should change prices (120ms)
    ✔ Should be able to set TP/SL prices to 0 (93ms)
    ✔ Should emit event after updateOrderConditions (95ms)
  getBestDexByOrder for limit orders
```

```
      ✔ When first dex is best to swap borrowedAmount return correct dexes name (622ms)
      ✔ When second dex is best to swap borrowedAmount return correct dexes name (641ms)
      ✔ When multiple shares return correct dexes (736ms)
    Order mappings
      ✔ Should revert if order does not exist
      ✔ Should have correct order count for different traders (295ms)
      ✔ Should have correct order count for different buckets (247ms)
      ✔ Should have correct order indexes (96ms)
      ✔ Should have correct order indexes for trader (139ms)
      ✔ Should have correct order indexes for buckets (195ms)
    pause & unpause
      ✔ Should revert if not EMERGENCY_ADMIN call pause
      ✔ Should revert if not SMALL_TIMELOCK_ADMIN call unpause

  LimitedMintTokens
Mints to test accounts completed.
    ✔ Should revert when setMintTimeLimit call not owner
    isTimeLimitedMinting is false
      ✔ Any amount of tokens should be minting to any account (60ms)
      ✔ Should be minting regardless of the time of the last minting
    isTimeLimitedMinting is true
      ✔ Should minting a constant amount of tokens on sender account regardless of the injected amount
and injected account (58ms)
      ✔ Should be revert if day has not passed since last minting
      ✔ Should be re-minting tokens if day has passed since last minting (39ms)

  PMXToken
    deploy with zero recipient address
      ✔ Should contain tokens total supply equal to initialSupply after the contract deploy
      ✔ Should mint total supply to deployer (159ms)
    deploy with non-zero recipient address
      ✔ Should mint total supply to recipient

  Ptoken
    initialization
      ✔ Should initialize with correct values. (917ms)
    setBucket
      ✔ Should setBucket (72ms)
      ✔ Should revert when bucket already set
      ✔ Should revert when a param 'IBucket _bucket' does not support IBucket (218ms)
    setInterestIncreaser
      ✔ Should revert if not BIG_TIMELOCK_ADMIN call setInterestIncreaser
      ✔ Should revert when the address does not support IBonusExecutor
      ✔ Should set the InterestIncreaser (70ms)
      ✔ Should set zero address
    setLenderRewardDistributor
      ✔ Should revert if not BIG_TIMELOCK_ADMIN call setLenderRewardDistributor
      ✔ Should revert when the address does not support IBonusExecutor (39ms)
      ✔ Should set the LenderRewardDistributor (294ms)
      ✔ Should set zero address (42ms)
    MintToReserve
      ✔ Should only allow the bucket to mint tokens
      ✔ Should return when the amountScaled is equal to zero
      ✔ Should mintToReserve
    Mint & Burn
      ✔ Should Mint
      ✔ Should revert Mint when a param 'address _user' is zero
      ✔ Should revert Mint when amount is 0
      ✔ Should revert Mint when invalid mint amount
      ✔ Should Burn (61ms)
      ✔ Should revert Burn when a param 'address _user' is zero
      ✔ Should revert Burn when invalid mint amount (45ms)
      ✔ Should revert transfer() when invalid transfer amount (43ms)
      ✔ Should revert transferFrom() when invalid transfer amount (57ms)
    Fixed-term deposits
      ✔ Should revert lockDeposit when _duration is 0
      ✔ Should revert lockDeposit when _amount is 0
      ✔ Should revert lockDeposit when the msg.sender is on the blacklist (41ms)
      ✔ Should revert lockDeposit when the bucket is not active
      ✔ Can't lockDeposit for other user
      ✔ lockDeposit should correct update state (73ms)
      ✔ Should revert unlockDeposit when msg.sender is on the blacklist (68ms)
```

```
      ✔ Should revert when unlockDeposit not owned user (160ms)
      ✔ Should lockDeposit and throw event (39ms)
      ✔ Should unlockDeposit and throw event (68ms)
      ✔ Should unlockDeposit  immediately when the bucket is delisted (64ms)
      ✔ Should revert depositProlongation when the msg.sender is on the blacklist (65ms)
      ✔ Should revert depositProlongation when the bucket is not active (58ms)
      ✔ Should DepositProlongation and throw event (58ms)
      ✔ Should revert lockDeposit when amount to lock is more available balance (85ms)
      ✔ Should revert lockDeposit when amount to lock + amountInLiquidityMining is more than available
balance (165ms)
      ✔ Should revert unlockDeposit when block.timestamp is lower or equal than deadline (86ms)
      ✔ unlockDeposit should correct update state (226ms)
      ✔ Should depositProlongation correct update state when block.timestamp is more than deadline (65ms)
      ✔ Should depositProlongation correct update state when block.timestamp is lower than deadline
(66ms)
    Deposit
      ✔ Should only allow the admin to mint tokens
      ✔ Should create locked deposit if duration isn't 0 (355ms)
      ✔ Should mint PTokens for given address, equal of providing TestTokenA amount (152ms)
      ✔ Should transfer testTokenA to Bucket contract (145ms)
      ✔ Should emit event 'Transfer' with correct parametrs (124ms)
      ✔ Should emit event 'Mint' with correct parametrs (118ms)
    balanceOf
      ✔ Should return balance without change
      ✔ Shouldn't increment balance over past 2 blocks (time past) because nobody borrowed yet
      ✔ Should increment balance over past 3 blocks, after deposit and borrowed (1176ms)
    availableBalanceOf
      ✔ Should availableBalanceOf be equal to balanceOf (877ms)
      ✔ Should update availableBalanceOf after lockDeposit (255ms)
      ✔ Should update availableBalanceOf after unlockDeposit (334ms)
    Transfers and allowances
      ✔ Should return total supply equals balanceOf lender
      ✔ Should return correct balances of two users after pToken transfer (366ms)
      ✔ Should revert transfer() when a param 'address _recipient' is zero
      ✔ Should update allowance after approve was changed (107ms)
      ✔ Should return correct balances of two users after pToken transferFrom and decrease allowance
(129ms)
      ✔ Should revert transferFrom() when a param 'address _sender' is zero
      ✔ Shold revert transferFrom() when a param 'address _recipient' is zero
      ✔ Should transfer full user balance to another user (303ms)
      ✔ Should transfer full user balance to another user using transferFrom (323ms)
      ✔ Should revert transfer if amount is more available balance (67ms)
      ✔ Should revert transferFrom if amount is more available balance (82ms)
      ✔ Should revert transfer if amount + amountInLiquidityMining is more than available balance (644ms)
      ✔ Should revert transferFrom if amount + amountInLiquidityMining is more available balance (894ms)
    Total supply
      ✔ Should return total supply equals balanceOf lender (50ms)
      ✔ Should return totalSupply equals to all user's balances (75ms)
    burn
      ✔ Should only allow the bucket to burn tokens
      ✔ Should burn PTokens and transfer correct amount of underlying asset to user (139ms)
      ✔ Should transfer testTokenA to user (126ms)
      ✔ Should emit event 'Transfer' with correct parametrs (112ms)
      ✔ Should emit event 'Burn' with correct parametrs (105ms)
      ✔ Should burn PTokens and transfer correct amount of underlying asset to user after 2 blocks
(136ms)
      ✔ Should return zero balance of PTokens, when user burn MaxUint of PTokens 2 blocks after deposit
(111ms)
      ✔ Can burn only available PToken amount (282ms)

  PhaseSwitching
    Should switch to phase 1 - deploy
      ✔ Should not add reward for spot trading (939ms)
      ✔ Should not add rewards for early lenders (134ms)
      ✔ Should not add rewards for early traders (616ms)
      ✔ Should not enable NFT bonuses (179ms)
    Should switch to phase 2 - spot trading rewards
Early rewards have been set!
SpotTradingRewardDistributor: setRewardPerPeriod scheduled in 1201s
=== finished ===
Early rewards have been set!
SpotTradingRewardDistributor: added 2000 to the available PMX balance.
```

```
SpotTradingRewardDistributor: available PMX balance is 2000.0.
SpotTradingRewardDistributor: setRewardPerPeriod executed
=== finished ===
      ✔ Should set correct params in spotTradingRewardDistributor
      ✔ Should add reward for spot trading (801ms)
    Should switch to phase 3 – rewards for early lenders
ActivityRewardDistributor: setupBucket scheduled in 1201s
Early rewards have been set!
=== finished ===
Early rewards have been set!
ActivityRewardDistributor: setupBucket executed
=== finished ===
      ✔ Should set correct params in ActivityRewardDistributor (117ms)
      ✔ Should add rewards for early lenders (155ms)
    Should switch to phase 4 – rewards for early traders
ActivityRewardDistributor: setupBucket scheduled in 1201s
Early rewards have been set!
=== finished ===
ActivityRewardDistributor: setupBucket executed
Early rewards have been set!
=== finished ===
      ✔ Should set correct params in ActivityRewardDistributor (122ms)
      ✔ Should add rewards for early traders (609ms)
    Should switch to phase 5 – enable NFT bonuses
PToken, DebtToken: setup NFT executed scheduled in 1797s
=== finished ===
PToken, DebtToken: setup NFT executed
=== finished ===
      ✔ Should enable NFT bonuses (191ms)
    Should switch to phase 6 – update from ePMX to PMX
Reward Distributors deployed
Update ePMX to PMX scheduled in 1797s
Update Reward Distributors scheduled in 1797s
=== finished ===
Update ePMX to PMX executed
Update Reward Distributors executed
=== finished ===
      ✔ Should switch from ePMX to PMX (289ms)

  PositionManager batch functions
    constructor
      ✔ Should deploy dexAdapter and set the correct PM and PriceOracle (64ms)
      ✔ Should revert when a param 'address registry' is not supported
    closeBatchPositions
      ✔ Should revert when the array of id positions is empty
      ✔ Should revert when the passed bucket address is not correct (639ms)
      ✔ Should revert when msg.sender is on the black list
      ✔ Should revert when the passed position asset doesn't match the asset of the positions (52ms)
      ✔ Should revert when the passed deposited asset doesn't match bucket's borrowed asset (58ms)
      ✔ Should revert when ids and conditionIndexes arrays have different length for TP/SL
      ✔ Should revert when the dex price is less than the oracle price by oracleTolerableLimit + 5%
(530ms)
      ✔ Should revert when the passed CloseReason is not supported (80ms)
      ✔ Shouldn't liquidate position until it is not risky (129ms)
      ✔ Should liquidate position if it's not risky but positionAsset is removed from allowedAsset of
this bucket (620ms)
      ✔ Should liquidate risky positions and transfer testTokenB from 'PositionManager' to dex (998ms)
      ✔ Should skip id if position does not exist (1272ms)
      ✔ Should liquidate 2 out of 3 positions when the last one is not risky (2187ms)
      ✔ Should liquidate 3 out of 4 positions when the second to last one is not risky (3319ms)
      ✔ Should liquidate risky positions and delete from traderPositions list (1202ms)
      ✔ Should liquidate risky position and fully repay trader's debt after n blocks (843ms)
      ✔ Should liquidate risky positions and fully delete trader's deposit from 'TraderBalanceVault'
(1426ms)
      ✔ Should liquidate risky position and burn the trader's debt tokens (842ms)
      ✔ Should liquidate risky position 1 block past and transfer testTokenA to 'Bucket' and the rest of
deposit transfer to Treasury (883ms)
      ✔ Should close positions by SL and return the rest of deposit to trader (847ms)
      ✔ Should close 2 out of 3 positions by SL when the last one can't be closed (1348ms)
      ✔ Should close 2 out of 3 positions by liqudation when the last has no debt (1464ms)
      ✔ Should be able to close spot positions by TP/SL (1290ms)
      ✔ Should revert spot batch close if it doesn't pass oracle check (861ms)
```

```
    ✔ Should revert if at least one position is spot and bucket is not AddressZero (473ms)
    ✔ Should revert batch close if at least one can't be closed by TP (1213ms)
    ✔ Should revert close by SL if first position has wrong close manager (783ms)
    ✔ Should revert close by TP if first position has wrong close manager (818ms)
    Batch close events
      ✔ Should liquidate risky positions and throw event (416ms)
      ✔ Should close positions by SL and throw event (423ms)
      ✔ Should close positions by TP and throw event (769ms)

PositionManager
  ✔ Should initialize with correct values
  initialize
    ✔ Should deploy (215ms)
    ✔ Should revert deploy when registry address not supported (242ms)
    ✔ Should revert deploy when dns address not supported (233ms)
    ✔ Should revert deploy when traderBalanceVault address not supported (232ms)
    ✔ Should revert deploy when priceOracle address not supported (239ms)
    ✔ Should revert deploy when keeperRewardDistributor address not supported (243ms)
    ✔ Should revert deploy when spotTradingRewardDistributor address not supported (240ms)
  sets
    setMaxPositionSize
      ✔ Should set maxPositionSize and emit event
      ✔ Should revert when addres one of tokens is zero address
      ✔ Should revert when token addreses are the same
  ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call setMaxPositionSize
setDefaultOracleTolerableLimit
  ✔ Should set DefaultOracleTolerableLimit
  ✔ Should revert when the precent of the price difference is more WAD
  ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call setDefaultOracleTolerableLimit
setSecurityBuffer
  ✔ Should set securityBuffer
  ✔ Should revert when newSecurityBucket is more WAD
  ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call setSecurityBuffer
setMaintenanceBuffer
  ✔ Should set maintenanceBuffer
  ✔ Should revert when newMaintenanceBuffer is more WAD
  ✔ Should revert when newMaintenanceBuffer is zero
  ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call setMaintenanceBuffer
setOracleTolerableLimit
  ✔ Should set OracleTolerableLimit
  ✔ Should revert when the precent of the price difference is more WAD
  ✔ Should revert when the percent is equal to zero
  ✔ Should revert when one of the assets is equal to zero
  ✔ Should revert when the asset addresses are identical
  ✔ Should revert if not BIG_TIMELOCK_ADMIN call setOracleTolerableLimit
getOracleTolerableLimit
  ✔ Should return the correct OracleTolerableLimit
  ✔ Should return the default value
    minPositionSize
      ✔ Should set minPositionSize
      ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call setMinPositionSize
      ✔ Should allow to set when minPositionToken is zero and minPositionSize is zero
  openPosition
    ✔ Should revert open position when the msg.sender is on the blacklist
    ✔ Should revert open position when the positionManager is paused
    - Should be reverted when the ancillary data is incorrect
    ✔ Should revert open position when token (testTokenX) not allowed (45ms)
    ✔ Should revert open position when the bucket is inactive (51ms)
    ✔ Should be revert when the dex price is less than the oracle price by DefaultOracleTolerableLimit
+ 5% (296ms)
    ✔ Should be revert when the dex price is less than the oracle price by OracleTolerableLimit + 5%
(304ms)
    ✔ Should revert openPosition when firstAssetRoutes is empty list (177ms)
    ✔ Should revert openPosition when depositInThirdAssetRoutes is not empty list and depositedAsset is
borrowedAsset (196ms)
    ✔ Should revert open position when the amount of tokens received is smaller amountOutMin (378ms)
    ✔ Should revert when firstAssetRoutes summ of shares is 0 (199ms)
    ✔ Should revert when closingManagerAddresses has duplicates (383ms)
    ✔ Should revert when closingManagerAddresses does not have CCM role (373ms)
    ✔ Should revert when trader balance in traderBalanceVault is smaller then depositAmount + feeAmount
(324ms)
    ✔ Should revert when trader balance in traderBalanceVault is smaller then feeAmount (339ms)
```

```
      ✔ Should revert openPosition is POSITION_SIZE_EXCEEDED (286ms)
      ✔ Should create position and the treasury receive fee amount in the native token (393ms)
      ✔ Should revert position creation if deposited asset is the native token (758ms)
      ✔ Should revert when isProtocolFeeInPmx is true & when trader balance in PMX in traderBalanceVault
is smaller then feeAmountInPmx (393ms)
      ✔ Should create position and reserve receive fee amount in PMX on trader balance vault when
isProtocolFeeInPmx is true (473ms)
      ✔ Should open position when the amount of tokens received is equal or more amountOutMin (1498ms)
      ✔ Should create 'Position' and transfer testTokenA from 'Bucket' to 'Pair' (504ms)
      ✔ Should create position and increase traders count, and add traderPositions (892ms)
      ✔ Should create three position with different time intervals between and add correct borrowIndex in
position. Success close positions (2558ms)
      ✔ Should open position and throw event (626ms)
      ✔ Should open position with stopLoss price > liquidation price (1001ms)
    positions
      ✔ should revert if position does not exist
      ✔ Should have correct position count for different traders (302ms)
      ✔ Should have correct position count for different buckets (557ms)
      - Should have correct position indexes
      - Should have correct position indexes for trader
      - Should have correct position indexes for buckets
    openPosition with deposit
      ✔ Should revert open position when the amount of tokens received is smaller amountOutMin (300ms)
      ✔ Should revert when the dex price is less than the oracle price by DefaultOracleTolerableLimit +5%
(309ms)
      ✔ Should revert when the dex price is less than the oracle price by oracleTolerableLimit +5%
(319ms)
      ✔ Should create position and the treasury receive fee amount in native token (399ms)
      ✔ Should revert when isProtocolFeeInPmx is true & when trader balance in PMX in traderBalanceVault
is smaller then feeAmountInPmx (456ms)
      ✔ Should create position and transfer fee amount in PMX to the treasury when isProtocolFeeInPmx is
true (440ms)
      ✔ Should create position and transfer fee amount in PMX from the Vault to the treasury when
isProtocolFeeInPmx is true (491ms)
      ✔ Should open position when the amount of tokens received is equal or more amountOutMin (924ms)
      ✔ Should transfer tokens to traderBalanceVault, as collateral for deal (382ms)
      ✔ Should revert when deposit Amount insufficient for deal (298ms)
    openPosition with minPositionSize
      ✔ Should revert when position size < minPositionSize (224ms)
      ✔ Should open position when position size >= minPositionSize (482ms)
    closePositionByCondition
      ✔ Should revert if the msg.sender is on the blacklist
      ✔ Should revert if conditional manager is AddressZero (419ms)
    closePosition
      ✔ Should revert when summ of shares is 0 (105ms)
      ✔ Shouldn't close position and throw revert if called by the NON-owner (149ms)
      ✔ Shouldn't close position and throw revert if the msg.sender is on the blacklist
      ✔ Should revert if SHARESONDEX_LENGTH_IS_0 (143ms)
      ✔ Should revert if deposit receiver is zero address
      ✔ Should close position and transfer testTokenB from 'PositionManager' to 'Pair' (286ms)
      ✔ Should be revert when the dex price is less than the oracle price by DefaultOracleTolerableLimit
+ 5% (180ms)
      ✔ Should not revert if called by a trustedAddress when the dex price is less than the oracle price
by DefaultOracleTolerableLimit + 5% (263ms)
      ✔ Should be revert when the dex price is less than the oracle price by oracleTolerableLimit + 5%
(498ms)
      ✔ Should close position and transfer testTokenA from 'Pair' (346ms)
      ✔ Should close position and delete trader position from traderPositions list (240ms)
      ✔ Should close position and fully repay traders debt (271ms)
      ✔ Should close position and fully repay traders debt after n block past (275ms)
      ✔ Should close position 1 block past and transfer increased full amount (principal + fees) of
testTokenA to 'Bucket' (277ms)
      ✔ Should close position 1 block past and transfer trader profit from PositionManager to
TraderBalanceVault when deal is profit (565ms)
      ✔ Should close position 1 block past and repay to bucket when deal is profit (460ms)
      ✔ Should close position 1 block past and unlock trader's tokens in deposit Vault (374ms)
      ✔ Should close position and throw event (363ms)
      ✔ Should close position with amountOutMin less than amountOut (352ms)
      ✔ Should close position with amountOutMin equal to amountOut (341ms)
      ✔ Should revert when amountOutMin greater than amountOut (243ms)
      ✔ Should close position with amountOutMin less or equal to amountOut on another dex (504ms)
    liquidatePosition by SL/TP
```

```
      ✔ Should revert if SHARESONDEX_LENGTH_IS_0 (272ms)
      ✔ Should be revert when the dex price is less than the oracle price by DefaultOracleTolerableLimit
+ 5% (390ms)
      ✔ Should be revert when the dex price is less than the oracle price by oracleTolerableLimit + 5%
(381ms)
      ✔ Should revert when _sharesOnDex summ of shares is 0 (101ms)
      ✔ Should revert when conditionIndex is out of bounds (153ms)
      ✔ Should close position by limit and transfer testTokenB from 'PositionManager' to 'Pair' (527ms)
      ✔ Should close position by limit and transfer testTokenA from 'Pair' (596ms)
      ✔ Should liquidate when position can be closed and correctly updated balances in the vault (613ms)
      ✔ Should liquidate when position can be closed and transfer profit from Bucket to
TraderBalanceVault (1262ms)
      ✔ Should liquidate when position can be closed – liquidator is trader and correctly updated
balances in the vault (456ms)
      ✔ Should liquidate risky position and throw event (1200ms)
    liquidatePosition
      ✔ Should revert when the dex price is less than the oracle price by oracleTolerableLimit + 5%
(983ms)
      ✔ Shouldn't liquidate position until it not risky (334ms)
      ✔ Should liquidate position if it's not risky but positionAsset is removed from allowedAsset of
this bucket (641ms)
      ✔ Should liquidate risky position and transfer testTokenB from 'PositionManager' to 'Pair' when
health position is equal to ~ 0.91wad (812ms)
      ✔ Should liquidate risky position and transfer testTokenB from 'PositionManager' to 'Pair' when
health position is equal to ~ 0.99wad (813ms)
      ✔ Should liquidate risky position and transfer testTokenB from 'PositionManager' to 'Pair' when
health position is equal to 1 wad (513ms)
      ✔ Should liquidate risky position and transfer testTokenA from 'Pair' (1150ms)
      ✔ Should liquidate risky position and delete trader position from traderPositions list (655ms)
      ✔ Should liquidate risky position and fully repay traders debt after n blocks past (723ms)
      ✔ Should liquidate risky position and fully delete trader's deposit from 'TraderBalanceVault'
(637ms)
      ✔ Should liquidate risky position and throw event (664ms)
      ✔ Should liquidate risky position and transfer rest of trader deposit to treasury (676ms)
      ✔ Should liquidate risky position 1 block past and transfer positionDebt (principal + fees) of
testTokenA to 'Bucket' (661ms)
    getBestDexByPosition
      ✔ When first dex is best to swap borrowedAmount return correct dexes name (371ms)
      ✔ When first dex is best to swap borrowedAmount but frozen return correct dexes name (217ms)
      ✔ When second dex is best to swap borrowedAmount return correct dexes name (528ms)
      ✔ When multiple shares return correct dexes (586ms)
      ✔ Should be revert if all dexes frozen (118ms)
      Add pair testTokenA–testTokenX on first dex
        ✔ When the first dex has this swap pair, and the second does not, should returns correct dexes
name (1189ms)
    isDelistedPosition
      ✔ should revert if position does not exist (358ms)
      ✔ should return 'false' when the bucket of the position is not delisted (383ms)
      ✔ should return 'true' when the bucket of the position is not active (382ms)
      ✔ should liquidate the position and throw the correct event (677ms)
    isPositionRisky
      ✔ should revert if position does not exist
      ✔ should return 'true' when position is risky at oracle price (170ms)
      ✔ should return 'false' when position is not risky at oracle price (171ms)
    updatePosition
      ✔ Should revert when caller is on the blacklist
      ✔ Should revert when caller is not trader
      ✔ Should change both stopLossPrice and takeProfitPrice (73ms)
      ✔ Should emit event after conditions update (46ms)
      ✔ Should be able to change stopLossPrice takeProfitPrice to 0
      ✔ Should revert increaseDeposit when caller is on the blacklist
      ✔ Should revert when increaseDeposit caller is not trader
      ✔ Should emit event after increaseDeposit (141ms)
      ✔ Should increaseDeposit in borrowAsset from wallet (157ms)
      ✔ Should increaseDeposit in borrowAsset from vault (194ms)
      ✔ Should increaseDeposit in other asset from wallet (342ms)
      ✔ Should revert when depositedAmountInBorrowed is less than amountOutMin (227ms)
      ✔ Should increaseDeposit in other asset from vault (451ms)
      ✔ Should be able to cover all debt with increaseDeposit (736ms)
      ✔ Should correctly count debt tokens after deposit increase (392ms)
      ✔ Should update available balance in traderBalanceVault when increase deposit over debt (234ms)
      ✔ Should increase deposit when previous position was closed (1007ms)
```

```
    ✔ Should revert depositDecrease when caller is on the blacklist
    ✔ Should revert when depositDecrease caller is not trader
    ✔ Should revert when decreaseDeposit amount is more than deposit
    ✔ Should revert when after decreaseDeposit position becomes risky (135ms)
    ✔ Should emit event after depositDecrease (169ms)
    ✔ Should decreaseDeposit (197ms)
    ✔ Should correctly count debt tokens after deposit decrease (403ms)
    ✔ Should decrease deposit when previous position was closed (1235ms)
  partiallyClosePosition
    ✔ Should revert when caller is on the blacklist
    ✔ Should revert when called not by trader
    ✔ Should revert when called with wrong position id
    ✔ Should partially close position (293ms)
    ✔ Should correctly count debt tokens after partially close position (710ms)
    ✔ Should correctly count debt tokens after multiple partial closing (4401ms)
    ✔ Should partially close position and emit event (859ms)
    ✔ Should revert when amount >= positionAmount
    ✔ Should partially close position without debt and a bucket is not removed (465ms)
    ✔ Should revert when position is too small after partially close position (273ms)
    ✔ Should revert when closing amount is too small (780ms)
  pause & unpause
    ✔ Should revert if not EMERGENCY_ADMIN call pause
    ✔ Should revert if not SMALL_TIMELOCK_ADMIN call unpause

  PriceFeedUpdaterTestService
    ✔ should checkArrayPriceFeed and return correct price feeds statuses (1662ms)
    constructor
      ✔ Should deploy (45ms)
      ✔ Should revert deploy when updater is zero address
      ✔ Should revert deploy when dexAdapter address not supported (142ms)
      ✔ Should revert deploy when router is zero address
    addRouter
      ✔ Should add new router
      ✔ Should revert if msg.sender hasn't BIG_TIMELOCK_ADMIN (45ms)
      ✔ Should revert when router address is 0 address
    deleteRouter
      ✔ Should delete router from array
      ✔ Should revert if msg.sender hasn't BIG_TIMELOCK_ADMIN (43ms)
      ✔ Should revert when router address is 0 address
    setDivider
      ✔ Should setDivider update divider
      ✔ Should revert if msg.sender hasn't BIG_TIMELOCK_ADMIN (47ms)
      ✔ Should revert when newDivider is 0
    updateArrayPriceFeed
      ✔ should update price feeds and emit events (44ms)
      ✔ Should revert if msg.sender hasn't BIG_TIMELOCK_ADMIN (46ms)
      ✔ Should revert two arguments lengths is not equal
    updatePriceFeed
      ✔ should update price feed and emit event
      ✔ Should revert if msg.sender hasn't BIG_TIMELOCK_ADMIN (42ms)
      ✔ Should revert when new set answer is 0
      ✔ Should revert when price feed address is zero address
    checkPriceFeed
      ✔ should checkPriceFeed and return correct price feed status (545ms)
      ✔ Should revert when one of addresses is 0
      ✔ Should revert when denominator is 0 (212ms)

  PriceFeedsAccessControl
    setAnswer
      ✔ Should revert if call account not DEFAULT_UPDATER_ROLE (50ms)
      ✔ Should update value if call account with DEFAULT_UPDATER_ROLE
    setDecimals
      ✔ Should revert if call account not DEFAULT_UPDATER_ROLE (45ms)
      ✔ Should update value if call account with DEFAULT_UPDATER_ROLE

  PrimexDNS
    ✔ getDnsBucketAddress revert if bucket not added
    ✔ getDnsDex revert if dex not added
    ✔ dnsBucket return correct bucketData
    ✔ dnsDex return correct dexData
    ✔ should revert if not MEDIUM_TIMELOCK_ADMIN call addBucket
    ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call addDEX
```

```
    ✔ addDEX should revert if address values are null address
  Initialization
    ✔ Should initialize primexDNS (190ms)
    ✔ Should revert when registry address not supported (115ms)
    ✔ Should revert when PMXToken address not supported (114ms)
    ✔ Should revert when treasury address not supported (233ms)
  setPMX
    ✔ should change pmx token address if by BIG_TIMELOCK_ADMIN role
    ✔ should revert if not BIG_TIMELOCK_ADMIN call setPMX
    ✔ should revert if new address null address
    ✔ should revert if new address is not supported
  setDexAdapter
    ✔ setDexAdapter if called BIG_TIMELOCK_ADMIN
    ✔ should revert if not BIG_TIMELOCK_ADMIN call setDexAdapter
    ✔ should revert if new address null address
    ✔ should revert if new DexAdapter address is not supported (134ms)
  setConditionalManager
    ✔ Should setConditionalManager if called by BIG_TIMELOCK_ADMIN
    ✔ Should revert if the new ConditionalManager address is not supported or is zero address (149ms)
    ✔ Should revert if nit BIG_TIMELOCK_ADMIN call setConditionalManager
  setAavePool
    ✔ setAavePool if called by BIG_TIMELOCK_ADMIN
    ✔ Should revert if not BIG_TIMELOCK_ADMIN call setAavePool
  setProtocolRate
    ✔ change protocolRate if called by BIG_TIMELOCK_ADMIN
    ✔ Should revert if not BIG_TIMELOCK_ADMIN call setProtocolRate
    ✔ should revert if new value is 0
  setProtocolRateInPmx
    ✔ change protocolRateInPmx if called by BIG_TIMELOCK_ADMIN
    ✔ Should revert if not BIG_TIMELOCK_ADMIN call setProtocolRate
    ✔ should revert if new value is 0
  addBucket
    ✔ dnsBucket return correct bucketData
    ✔ addBucket with liquidityMining shoulld transfer pmx to LiquidityMiningRewardDistributor (269ms)
    ✔ getDnsBucketAddress returns the correct bucket address
    ✔ function addBucket creates an event with bucket parameters
    ✔ addBucket should revert if exist bucket with this name
    ✔ Should revert 'addBucket()' when bucket address not supported (133ms)
    ✔ should revert if not EMERGENCY_ADMIN call freezeBucket
    deprecateBucket
      ✔ Should revert if not BIG_TIMELOCK_ADMIN call deprecateBucket
      ✔ Should revert when the bucket is not active
      ✔ Should deprecateBucket
    freezeBucket
      ✔ dnsBucket return correct bucketData
      ✔ getDnsBucketAddress revert if bucket not active
      ✔ freezeBucket should revert if bucket already frozen
      ✔ Should revert if not SMALL_TIMELOCK_ADMIN call activateBucket
      activateBucket
        ✔ dnsBucket return correct bucketData
        ✔ getDnsBucketAddress returns the correct bucket address
        ✔ activateBucket should revert if bucket already activated
  addDEX
    ✔ getDnsDex return correct routerAddress
    ✔ dnsDex return correct dexData
    ✔ function addDEX creates an event with dex parameters
    ✔ addDEX should revert if exist dex with this name
    ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call freezeDEX
    freezeDEX
      ✔ getDnsDex revert if dex not active
      ✔ dnsDex return correct dexData
      ✔ freezeDEX should revert if dex already frozen
      ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call activateDEX
      activateDEX
        ✔ getDnsDex correct routerAddress and adapterAddress
        ✔ dnsDex return correct dexData
        ✔ activateBucket should revert if bucket already activated

PrimexLens
  ✔ getBucket return correct values (771ms)
  ✔ getBucketsArray does not return a bucket if it is deprecated and 'showDeprecated' param is false
and there is no user's deposit in a bucket (1354ms)
```

```
    ✔ getBucketsArray returns correct values if the bucket is deprecated and 'showDeprecated' param is
false but there is user's deposit in a bucket (968ms)
    ✔ getBucketsArray returns correct values if the bucket is deprecated and 'showDeprecated' param is
true (1470ms)
    ✔ getBucketsArray does not return empty buckets (2217ms)
    ✔ getOpenPositionData return correct values (1171ms)
    ✔ getOpenPositionData return correct values for spot position (548ms)
    ✔ getLiquidationPrice return correct values (269ms)
    constructor
      ✔ Should deploy (44ms)
      ✔ Should revert deploy when takeProfitStopLossCCM address not supported
    getPositionStatus
      ✔ Should return correctly position status (167ms)
    getArrayOpenPositionDataByTrader
      ✔ should return empty data if position by trader does not exist, cursor greater or equal position
length
      ✔ should return all positions data by trader, cursor plus count greater than positions length
(1142ms)
      ✔ should return correct positions data by trader and new cursor when cursor plus count less than
positions length  (819ms)
      ✔ should revert if positionManager address is not correct
    getArrayOpenPositionDataByBucket
      ✔ should return empty data if position by bucket does not exist, cursor greater or equal position
length
      ✔ should return all positions data by bucket, cursor plus count greater than positions length
(1147ms)
      ✔ should return correct positions data by bucket and new cursor when cursor plus count less than
positions length  (772ms)
      ✔ should revert if positionManager address is not correct
    getOpenPositionsWithConditions
      ✔ should return empty data if position does not exist, cursor greater or equal position length
      ✔ should return all positions data, cursor plus count greater than position length (81ms)
      ✔ should return correct conditions data if a position is closed (368ms)
      ✔ should revert if positionManager address is not correct
    getLimitOrdersWithConditions
      ✔ should return empty data if order does not exist, cursor greater or equal orders length
      ✔ should return all orders data, cursor plus count greater or equal orders length (68ms)
      ✔ should return correct conditions data if a order is closed (162ms)
      ✔ should revert if Limit order Manager address is not correct
    getLiquidationPrice for openable positions
      ✔ should return correct values (631ms)
      ✔ should revert if first argument isn't positionManager
      ✔ should revert if positionAsset isn't allowed in bucket (104ms)
    getPositionMaxDecrease
      ✔ should return depositedAmount if max decrease > depositedAmount (220ms)
      ✔ should get position max decrease (861ms)

  PrimexUpkeep
    PrimexUpkeep functionality
      ✔ Should revert deploy when liquidations output size is 0
      ✔ checkUpkeep return correct new cursor when amount position to close is liquidationsOutputSize and
count positions are more liquidationsOutputSize  (794ms)
      ✔ checkUpkeep return correct new cursor when amount open position by order is openByOrderOutputSize
and count orders are more openByOrderOutputSize (996ms)
      ✔ PrimexUpkeep flow for positions (2866ms)
      ✔ PrimexUpkeep flow for orders (4575ms)
      ✔ PrimexUpkeep close positions by performUpkeepPositions (2914ms)
      ✔ PrimexUpkeep open positions by performUpkeepOrders (4149ms)
      constructor
        ✔ Should deploy (74ms)
        ✔ Should revert deploy when position manager address not supported (276ms)
        ✔ Should revert deploy when limit order manager address not supported (148ms)
        ✔ Should revert deploy when registry address not supported (423ms)
        ✔ Should revert deploy when bestDexLens address not supported (145ms)
        ✔ Should revert deploy when primexLens address not supported (137ms)
    Splitter functionality
      Deposit in a first asset
        ✔ Should close position by condition on multiple dexes – deposit in a first asset (3211ms)
        ✔ Should open position by order on multiple dexes – deposit in a first asset (3309ms)
      Deposit in a second asset
        ✔ Should close position by condition on multiple dexes – deposit in a second asset (5748ms)
        ✔ Should open position by order on multiple dexes – deposit in a second asset (7198ms)
```

```
      Deposit in a third asset
        ✔ Should open position by order on multiple dexes - deposit in a third asset (6231ms)
        ✔ Should close position by condition on multiple dexes - deposit in a third asset (3478ms)

  ReferralProgram
    ✔ addReferrals (104ms)
    ✔ notWhitelistedReferrer
    ✔ whitelistedReferral
    ✔ parentExist (51ms)
    ✔ Should revert when referrer address wrong (44ms)
    ✔ blacklistedReferral
    ✔ getReferrers (65ms)
    initialize
      ✔ Should deploy (200ms)
      ✔ Should revert deploy when registry address not supported (274ms)
      ✔ Should revert deploy when whiteBlackList address not supported (253ms)

  SpotTrading
    openPosition
      ✔ Should revert openPosition when firstAssetRoutes is empty list and it's spot (134ms)
      ✔ Should revert openPosition when depositInThirdAssetRoutes is not empty list and it's spot (122ms)
      ✔ Should revert openPosition when depositAsset is equal positionAsset
      ✔ Should revert openPosition when position asset doesn't have oracle price feed with the deposit
asset. (55ms)
      ✔ Should create 'Position' and swap trader deposit (339ms)
      ✔ Should create 'Position' and transfer depositAmount from trader (327ms)
      ✔ Should create position and increase traders count, and add traderPositions (295ms)
      ✔ Should open position and throw event (369ms)
      ✔ Should open position with stopLoss price < currentPrice (313ms)
      ✔ Should transfer tokens from traderBalanceVault when openPosition with makeDeposit is false
(354ms)
      ✔ Should revert openPosition with fee in PMX when user balance in traderBalanceVault doesn't have
enough pmx with makeDeposit is false (109ms)
      ✔ Should revert openPosition with fee in PMX when user balance in traderBalanceVault doesn't have
enough protocolFee assets (pmx) (323ms)
      ✔ Should openPosition with fee in PMX from vault (391ms)
    openPosition with minPositionSize
      ✔ Should revert when depositedAmount < minPositionSize (257ms)
      ✔ Should open position when position size >= minPositionSize (426ms)
    openPosition with maxPositionSize
      ✔ Should revert when position size > maxPositionSize (323ms)
      ✔ Should open position when position size <= maxPositionSize (387ms)
    closePosition
      ✔ Should revert if SHARESONDEX_LENGTH_IS_0 (73ms)
      ✔ Should close position and transfer testTokenD from 'PositionManager' to 'Pair' (231ms)
      ✔ Should close position and transfer testTokenC from 'Pair' to 'traderBalanceVault' (386ms)
      ✔ Should close position and delete trader position from traderPositions list (226ms)
      ✔ Should close position and update available balance of trader's tokens in trader balance Vault
(347ms)
      ✔ Should close position and throw event (285ms)
      ✔ Should revert close position when prices on dex and oracle are different (176ms)
      ✔ Should partially close spot position (328ms)
    liquidatePosition by SL/TP
      ✔ Should close position by stop loss and transfer testTokenD from 'PositionManager' to 'Pair'
(483ms)
      ✔ Should close position by stop loss and transfer testTokenC from 'Pair' to 'traderBalanceVault'
(388ms)
      ✔ Should liquidate position by stop loss and correctly updated balances in the vault (471ms)
      ✔ Should liquidate position by take profit and correctly updated balances in the vault (577ms)
      ✔ Should liquidate position by stop loss and throw event (758ms)
    getBestDexByPosition
      ✔ When first dex is best to swap borrowedAmount return correct dexes name (390ms)
      ✔ When second dex is best to swap borrowedAmount return correct dexes name (590ms)
    getBestDexForOpenablePosition
      ✔ When first dex is best to open spot position return correct dex name (452ms)
      ✔ When second dex is best to open spot position return correct dexes name (576ms)
    getCurrentPriceAndProfitByPosition
      ✔ Return correct profit and current price (482ms)
    canBeClosed
      ✔ isStopLossReached should return 'false' when stopLossPrice < oracle price (315ms)
      ✔ isStopLossReached should return 'true' when oracle price <= stopLossPrice (323ms)
      ✔ isTakeProfitReached should return 'false' when takeProfitPrice >= price on dex (439ms)
```

```
      ✔ isTakeProfitReached should return 'true' when takeProfitPrice <= price on dex (640ms)
    Limit Order
      CreateLimitOrder
        ✔ Should revert when depositedAsset is equal positionAsset
        ✔ Should revert when leverage is not 1
        ✔ Should revert when position asset doesn't have oracle price feed with the deposit asset. (43ms)
        ✔ Should create 'LimitOrder' and transfer testTokenC from trader to 'TraderBalanceVault' (233ms)
        ✔ Should create 'LimitOrder' with makeDeposit=false and lock testTokenC in 'TraderBalanceVault'
(252ms)
        ✔ Should create 'LimitOrder' with the correct variables (200ms)
        ✔ Should create 'LimitOrder' with isProtocolFeeInPmx=true with makeDeposit=true (242ms)
        ✔ Should create 'LimitOrder' with isProtocolFeeInPmx=true with makeDeposit=false (258ms)
        ✔ Should open spot limit order and throw event (204ms)
        ✔ Should open swap limit order and throw event (199ms)
        ✔ Should createLimitOrder with stopLossPrice*positionAmount < depositedAmount (160ms)
      CancelLimitOrder
        ✔ Should cancel spot limit order and throw event (72ms)
      CreateLimitOrder with minPositionSize
        ✔ Should revert when depositedAmount < minPositionSize (93ms)
        ✔ Should create limit order when position size >= minPositionSize (170ms)
        ✔ Should return false in canBeFilled when depositedAmount < minPositionSize (263ms)
        ✔ Should revert openPositionByOrder when position size < minPositionSize (824ms)
      canBeFilled
        ✔ Should revert when firstAssetRoutes length is empty (687ms)
        ✔ Should return true when limitPrice is less than current price on dex and trader has enough pmx
in traderBalanceVault (546ms)
        ✔ Should return true when limitPrice is current price on dex  (490ms)
        ✔ Should return false when limitPrice is more than current price on dex  (865ms)
        ✔ Should return false when limitPrice > current price(10) but deadline < block.timestamp
      openPositionByOrder - spot order
        ✔ Should revert openPositionByOrder when firstAssetRoutes is empty list (134ms)
        ✔ Should revert openPositionByOrder when depositInThirdAssetRoutes is not empty list (163ms)
        ✔ Should revert when the order price isn't reached (609ms)
        ✔ Should create position by order and transfer testTokenC from 'Bucket' to 'Pair' (453ms)
        ✔ Should create position by order and transfer testTokenD to 'PositionManager' (410ms)
        ✔ Should create position by order, increase traders count, add traderPositions and then deleted
the order (433ms)
        ✔ Should open position by order and throw event 'OpenPosition' (520ms)
        ✔ Should open position by order and throw event 'CloseLimitOrder' (750ms)
        ✔ Should open position by order and lock trader deposit in traderBalanceVault (429ms)
        ✔ Should open position by order when isProtocolFeeInPmx=true (659ms)
      openPositionByOrder - swap order
        ✔ Should revert when the order price isn't reached (476ms)
        ✔ Should not create position by order and transfer testTokenC from 'Bucket' to 'Pair' (302ms)
        ✔ Should not create position by order and transfer testTokenD to traderBalanceVault, update
trader balance in traderBalanceVault (341ms)
        ✔ Should not create position by order and transfer testTokenD to traderBalanceVault, update
trader balance in traderBalanceVault. protocolFeeInPmx=true (528ms)
        ✔ Should throw event 'CloseLimitOrder' (274ms)
      getBestDexByOrder
        ✔ When first dex is best to swap borrowedAmount return correct dexes name (543ms)
        ✔ When second dex is best to swap borrowedAmount return correct dexes name (548ms)
    updatePosition
      ✔ Revert when increaseDeposit for a position if borrowed amount = 0
      ✔ Revert when decreaseDeposit for spot position
    updateOrder
      ✔ Should update shouldOpenPosition (49ms)
      ✔ Should revert when update leverage (209ms)

  InterestRateStrategy_integration
    calculateInterestRates
      ✔ Should return correct BAR and LAR for Utilization Ratio = 0% (95ms)
      ✔ Should return correct BAR and LAR for Utilization Ratio = 5% (496ms)
      ✔ Should return correct BAR and LAR for Utilization Ratio = 20% (498ms)
      ✔ Should return correct BAR and LAR for Utilization Ratio = 40% (495ms)
      ✔ Should return correct BAR and LAR for Utilization Ratio = 50% (461ms)
      ✔ Should return correct BAR and LAR for UR > URoptimal and b1 < 0 (489ms)
      ✔ Should revert when BAR overflows (1019ms)
      ✔ Should return correct BAR and LAR for Utilization Ratio = 1.00 (491ms)
      ✔ Should return correct BAR and LAR for fractioned Utilization Ratio = 0.010% (468ms)
      ✔ Should return correct BAR and LAR for fractioned Utilization Ratio = 4.351% (462ms)
      ✔ Should return correct BAR and LAR for fractioned Utilization Ratio = 8.692% (511ms)
```

```
    ✔ Should return correct BAR and LAR for fractioned Utilization Ratio = 13.033% (484ms)
    ✔ Should return correct BAR and LAR for fractioned Utilization Ratio = 17.374% (848ms)
    ✔ Should return correct BAR and LAR for fractioned Utilization Ratio = 21.715% (449ms)
    ✔ Should return correct BAR and LAR for fractioned Utilization Ratio = 26.056% (453ms)
    ✔ Should return correct BAR and LAR for fractioned Utilization Ratio = 30.397% (484ms)
    ✔ Should return correct BAR and LAR for fractioned Utilization Ratio = 34.738% (467ms)
    ✔ Should return correct BAR and LAR for fractioned Utilization Ratio = 39.079% (464ms)
    ✔ Should return correct BAR and LAR for fractioned Utilization Ratio = 43.420% (560ms)
    ✔ Should return correct BAR and LAR for fractioned Utilization Ratio = 47.761% (516ms)
    ✔ BAR, LAR change depending on the accumulation of debt (946ms)

KeeperRewardDistributor_integration
  ✔ Should updateReward for keeper (103ms)
  ✔ Should updateReward by maxGasAmount when gasSpent > maxGasAmount (116ms)
  ✔ Should updateReward by oracleGasPrice when gasPrice > oracleGasPrice + % (151ms)
  ✔ Should close position by stop loss condition and update keeper balance (688ms)
  ✔ Should close position by trailing stop condition and update keeper balance (863ms)
  ✔ Should close risky position and update keeper balance (1379ms)
  ✔ Should batch close risky positions and update keeper balance (1927ms)
  ✔ Should open position by order and update keeper activity (757ms)
  ✔ Should not update keeper balance when position closed by trader (1303ms)
  ✔ Should claim reward, update keeper balance and emit events (862ms)

LenderRewardDistributor_integration
    ✔ Step 0. Should setup lender reward distributor (65ms)
    ✔ Step 1. Should update users activities after deposit and withdraw from bucket (370ms)
    ✔ Step 2. Should update users activities after transfer and transferFrom PToken (162ms)
    ✔ Step 3. ClaimReward (210ms)

  LimitPriceCOM_integration
    constructor
      ✔ Should initialize with correct values
      ✔ Should revert when initialized with wrong primexDNS address (106ms)
      ✔ Should revert when initialized with wrong priceOracle address (109ms)
      ✔ Should revert when initialized with wrong positionManager address (129ms)
    canBeFilled
      ✔ Should revert when depositInThirdAssetRoutes length isn't 0 (51ms)
      ✔ Should return true when limitPrice is less than current price on dex (126ms)
      ✔ Should return true when limitPrice is current price on dex  (143ms)
      ✔ Should return false when limitPrice is more than current price on dex  (106ms)
      ✔ Should return false when limitPrice > current price(10) but deadline < block.timestamp
      ✔ Should return false when calculated leverage is more then maxAssetLeverage (244ms)
      ✔ Should return false when size of opened position by this order will be more than maximum position
size by pair (113ms)

  LiquidityMiningRewardDistributor_integration
    ✔ Should set minReward = maxReward if liquidityMiningDeadline is reached and the bucket is launched
(103ms)
    ✔ Should not launch the bucket if availableLiquidity > liquidityMiningAmount (top-up bucket via
transfer) and liquidityMiningDeadline is not reached (86ms)
    ✔ Should not change lender info if the lender added liquidity to the bucket via transfer (151ms)
    withdrawPmxByAdmin
      bucket with liquidity mining
        ✔ Should revert if reinvestment period is not over (52ms)
        ✔ Should revert if reinvestment period is over but the bucket isLaunched (122ms)
        ✔ Should emit WithdrawPmxByAdmin when transfer to treasury is successful (45ms)
        ✔ Should update withdrawnRewards when transfer to treasury is successful (45ms)
        ✔ Should revert if amount to transfer to treasury is zero (51ms)
        ✔ Should revert if caller is not granted with MEDIUM_TIMELOCK_ADMIN
        ✔ Should transfer to treasury pmxAmount without withdrawnRewards (91ms)
      delisted bucket
        ✔ Should emit WithdrawPmxByAdmin when transfer to treasury is successful (173ms)

  Operation with the Balancer dex_integration
    For a pool with 3 tokens
      ✔ Should return correct getAmountsOut (299ms)
      ✔ Should open position and increase position count when tokenToBuy is testTokenB (353ms)
      ✔ Should close position and decrease position count when tokenToBuy is testTokenB (713ms)
      ✔ Should open a position and increase position count when tokenToBuy is testTokenX (450ms)
      ✔ Should close a position and decrease position count when tokenToBuy is testTokenX (781ms)
    For a pool with 4 tokens
      ✔ Should return correct getAmountsOut (314ms)
```

```
    ✔ Should open position and increase position count when tokenToBuy is testTokenB (351ms)
    ✔ Should close position and decrease position count when tokenToBuy is testTokenB (716ms)
    ✔ Should open a position and increase position count when tokenToBuy is testTokenX (454ms)
    ✔ Should close a position and decrease position count when tokenToBuy is testTokenX (697ms)

Operation with the Curve dex_integration
  For a pool with 3 tokens
    ✔ Should return correct getAmountsOut (652ms)
    ✔ Should open position and increase position count when tokenToBuy is testTokenB (551ms)
    ✔ Should open a position and increase position count when tokenToBuy is testTokenX (797ms)
    ✔ Should close a position and decrease position count when tokenToBuy is testTokenB (1518ms)
    ✔ Should close a position and decrease position count when tokenToBuy is testTokenX (1422ms)

PrimexNFT_integration
  InterestIncreaser
    ✔ Should block NFT and delete the activated bonus in the executor (162ms)
    ✔ Should activate nft both in the Nft and in the BonusExecutor contract (110ms)
    ✔ Should update the activated bonus via transfer pTokens (222ms)
    ✔ Should correct return accumulatedAmount when caller isn't _user (164ms)
    ✔ Should update the activated bonus via transferFrom (230ms)
    ✔ Should correctly update the activated bonus via transferFrom to itself (232ms)
    ✔ Should update the activated bonus via mint (309ms)
    ✔ Should update the activated bonus via burn (637ms)
    ✔ Should claim and transfer pTokens (485ms)
    FeeDecreaser
      ✔ Should activate nft both in the Nft and in the BonusExecutor contract (389ms)
      ✔ Should update the activated bonus via mint (775ms)
      ✔ Should update the activated bonus via burn (1005ms)
      ✔ Should  the activated bonus via burn (194ms)
      ✔ Should claim and transfer pTokens (556ms)

  PriceOracle_integration
    Price oracle Contract tests
      ✔ set price in chainLink and get with oracle (69ms)
      ✔ set fractional price in chainLink and get with oracle (55ms)
      ✔ Should set gas price feed and return gas price from it

  PrimexPricingLibrary_integration
    getOracleAmountsOut
      ✔ Should return the correct amount2. In PriceFeed, the direction of tokens is token x, token y. The
decimals of tokens x>y. Call getOracleAmountsOut(x,y,amount1)
      ✔ should return the correct amount2. In PriceFeed, the direction of tokens is token x, token y. The
decimals of tokens x>y. Call getOracleAmountsOut(y,x,amount1)
      ✔ should return the correct amount2. In PriceFeed, the direction of tokens is token y, token x. The
decimals of tokens x>y. Call getOracleAmountsOut(y,x,amount1)
      ✔ should return the correct amount2. In PriceFeed, the direction of tokens is token y, token x. The
decimals of tokens x>y. Call getOracleAmountsOut(x,y,amount1)
    getAmountOut
      ✔ getAmountOut TestTokenA to TestTokenB on uniswap (200ms)
      ✔ getAmountIn TestTokenA to TestTokenB on uniswap (194ms)
      ✔ getAmountOut TestTokenA to TestTokenB on sushiswap (194ms)
      ✔ getAmountIn TestTokenA to TestTokenB on sushiswap (194ms)
      ✔ getAmountOut TestTokenA to TestTokenX on uniswap (193ms)
      ✔ getAmountIn TestTokenA to TestTokenX on uniswap (335ms)
      ✔ getAmountOut TestTokenA to TestTokenX on sushiswap (227ms)
      ✔ getAmountIn TestTokenA to TestTokenX on sushiswap (335ms)
      ✔ getAmountOut TestTokenB to TestTokenA on uniswap (213ms)
      ✔ getAmountIn TestTokenB to TestTokenA on uniswap (195ms)
      ✔ getAmountOut TestTokenB to TestTokenA on sushiswap (442ms)
      ✔ getAmountIn TestTokenB to TestTokenA on sushiswap (233ms)
      ✔ getAmountOut TestTokenB to TestTokenX on uniswap (219ms)
      ✔ getAmountIn TestTokenB to TestTokenX on uniswap (237ms)
      ✔ getAmountOut TestTokenB to TestTokenX on sushiswap (239ms)
      ✔ getAmountIn TestTokenB to TestTokenX on sushiswap (232ms)
      ✔ getAmountOut TestTokenX to TestTokenA on uniswap (256ms)
      ✔ getAmountIn TestTokenX to TestTokenA on uniswap (210ms)
      ✔ getAmountOut TestTokenX to TestTokenA on sushiswap (196ms)
      ✔ getAmountIn TestTokenX to TestTokenA on sushiswap (197ms)
      ✔ getAmountOut TestTokenX to TestTokenB on uniswap (197ms)
      ✔ getAmountIn TestTokenX to TestTokenB on uniswap (239ms)
      ✔ getAmountOut TestTokenX to TestTokenB on sushiswap (241ms)
      ✔ getAmountIn TestTokenX to TestTokenB on sushiswap (213ms)
```

```
    ✔ should revert if first asset in path is incorrect (71ms)
    ✔ should revert if last asset in path is incorrect (66ms)
    ✔ should getAmountOut when path length > 2 (206ms)
    ✔ should getAmountOut when path has different dexes (348ms)
    ✔ should getAmountIn when path has different dexes (374ms)
  getDepositedAmountInBorrowed
    ✔ Should return correct amount (248ms)
  multiSwap
    ✔ Should return correct amount (269ms)
    ✔ should revert if first asset in path is incorrect (113ms)
    ✔ should revert if last asset in path is incorrect (151ms)
    ✔ should swap when path length > 2 (268ms)
    ✔ should swap when path has different dexes (487ms)
    ✔ should revert when dex returns a value but doesn't swap anything (322ms)
  getLiquidationPrice
    ✔ Should return correct price for position (163ms)
  getLiquidationPriceByOrder
    ✔ Should return correct price for order (65ms)
  Encode path
    ✔ Should encode path for uniswap (42ms)
    ✔ Should encode path for uniswapv3 (63ms)
    ✔ Should encode path for curve (67ms)
    ✔ Should encode path for balancer (779ms)
    ✔ Should revert encode path when dex is not supported
  Decode path
    ✔ Should decode path for uniswap (41ms)
    ✔ Should decode path for uniswapv3 (55ms)
    ✔ Should decode path for curve (45ms)
    ✔ Should decode path for balancer (729ms)

PrimexTimelocks_integration
  ✔ GUARDIAN_ADMIN can cancel operations in each timelock (107ms)
  ✔ scheduleBatch and executeBatch are working (109ms)
  BigTimelockAdmin
    ✔ BigTimelockAdmin has BIG_TIMELOCK_ADMIN
    ✔ BigTimelockAdmin can call admin method (53ms)
  MediumTimelockAdmin
    ✔ MediumTimelockAdmin has MEDIUM_TIMELOCK
    ✔ MediumTimelockAdmin can call method with MEDIUM_TIMELOCK_ADMIN (49ms)
  SmallTimelockAdmin
    ✔ SmallTimelockAdmin has SMALL_TIMELOCK_ADMIN
    ✔ SmallTimelockAdmin can call method with SMALL_TIMELOCK_ADMIN (58ms)

Redeemer_integration
  Redeem
    ✔ Should redeem and transfer early tokens to the Redeemer contract (52ms)
    ✔ Should redeem and transfer pmx tokens to the caller address (55ms)
    ✔ Should redeem and transfer pmx tokens to the caller address with x2 rate (55ms)
    ✔ Should redeem and transfer pmx tokens to the caller address with 0.5 rate (59ms)
  adminClaimEarlyTokens
    ✔ Should revert if not BIG_TIMELOCK_ADMIN call adminClaimEarlyTokens
    ✔ Should claim
  adminClaimRegularTokens
    ✔ Should revert if not BIG_TIMELOCK_ADMIN call adminClaimRegularTokens
    ✔ Should claim (39ms)

Reserve_integration
  paybackPermanentLoss() called with a Primex Bucket as a param
    ✔ Should revert if permanentLoss is 0 (1216ms)
    ✔ Should emit BurnAmountCalculated event with an arg 'burnAmount' (1646ms)
    ✔ Should emit Burn event when pTokens were burned (1611ms)
  paybackPermanentLoss() called with an Attacker Bucket as a param
    ✔ Should revert if attacker bucket address is not one of dnsBucket (38ms)
  transferToTreasury
    ✔ Should revert if Reserve balance is not sufficient (42ms)
    ✔ Should revert if Reserve balance is not sufficient by minPercentOfTotalSupplyToBeLeft (69ms)
    ✔ Should revert if Reserve balance is not sufficient by minAmountToBeLeft (55ms)
    ✔ Should emit TransferFromReserve event when transfer is successful (141ms)
    ✔ Should increase Treasury balance and decrease Bucket balance when transfer is successful (180ms)
    ✔ Should burn pTokens on Reserve balance when transfer is successful (201ms)

SpotTradingRewardDistributor_integration
```

```
    withdrawPmx
      ✔ Should transfer pmx to the recipient and reduce contract balance (217ms)
    updateTraderActivity
      ✔ Should openPositionByOrder and update trader activity (583ms)
      ✔ Should openPosition by user and trader and update their trader's activity appropriately (672ms)
    claimReward
      ✔ Should transfer rewards on the balance in traderBalanceVault and reduce contract balance (437ms)

  SwapManager_integration
    swap
      ✔ Should revert if the swapManager is paused
      ✔ Should revert if the msg.sender is on the blacklist
      ✔ Should revert if amountOutMin more than amountOut (137ms)
      ✔ Should swap from user wallet and return correct value (143ms)
      ✔ Should swap from user wallet and emit SpotSwap event (150ms)
      ✔ Should swap and increase balance of receiver (155ms)
      ✔ Should decrease user wallet balance by amount and increase treasury balance by feeAmountInEth
when isSwapFromWallet is true (166ms)
      ✔ Should return change to msg.sender when msg.value > feeAmountInEth isSwapFromWallet is true
(160ms)
      ✔ Should not take fee when user has NO_FEE_ROLE (755ms)
      ✔ Should receive swap to user balance in protocol (177ms)
      ✔ Should swap and send fee to the treasury (161ms)
      ✔ Should revert when user has not enough balance in protocol (48ms)
      ✔ Should swap from user balance in protocol (252ms)
      ✔ Should revert when isSwapFeeInPmx is true & when trader balance in PMX in traderBalanceVault is
smaller then feeAmountInPmx (53ms)
      ✔ Should swap and the treasury receive fee amount in PMX when isProtocolFeeInPmx is true (346ms)
      ✔ Should swap and should increase treasury balance by fee amount in PMX when isProtocolFeeInPmx and
isSwapFromWallet is true (278ms)
      ✔ Should swap and should not increase treasury balance if swapRateInPmx = 0 when fee in PMX (262ms)
      ✔ Should swap and should not increase treasury balance if swapRate = 0 when fee in ETH (225ms)

  TakeProfitStopLossCCM_integration
    constructor
      ✔ Should initialize with correct values
      ✔ Should revert when initialized with wrong primexDNS address (137ms)
      ✔ Should revert when initialized with wrong priceOracle address (146ms)
    canBeClosed
      ✔ should return 'false' when stopLossPrice and takeProfitPrice is zero (556ms)
      ✔ isStopLossReached should return 'false' when stopLossPrice < oracle price (535ms)
      ✔ isStopLossReached should return 'true' when oracle price <= stopLossPrice (757ms)
      ✔ isTakeProfitReached should return 'false' when takeProfitPrice >= oracle price (657ms)
      ✔ isTakeProfitReached should return 'true' when takeProfitPrice is lower market price (689ms)

  TraderBalanceVault_integration
    deposit()
      ✔ Should transfer 'testTokenA' tokens from msg.sender to TraderBalanceVault contract (66ms)
      ✔ Should transfer native currency from msg.sender to TraderBalanceVault contract
      ✔ Should transfer native currency from msg.sender to TraderBalanceVault contract via the receive
function (144ms)
    withdraw()
      ✔ Should transfer 'testTokenA' tokens from TraderBalanceVault contract to msg.sender (107ms)
      ✔ Should transfer native currency from TraderBalanceVault contract to msg.sender (53ms)

  TraderRewardDistributor_integration
    ✔ Step 0. Should setup trader reward distributor (82ms)
    ✔ Step 1. Should update users activities after increase and decrease debt(open and close position)
(2205ms)
    ✔ Step 2. ClaimReward (253ms)

  TrailingStopCCM_integration
    canBeClosed
      ✔ should return 'false' when highPrice < params.activationPrice (539ms)
      ✔ should revert when highPrice base timestamp is before position timestamp (497ms)
      ✔ should revert when highPrice quote timestamp is before position timestamp (574ms)
      ✔ should revert when there is no intersection in feed timestamps (602ms)
      ✔ should revert when lowPriceRoundNumber is more than latest round (650ms)
      ✔ should return true and be able to close position when trailing stop is reached (1014ms)

  Treasury_integration
    transferFromTreasury
```

```
    ✔ Should revert when amount > maxAmountDuringTimeframe during current timeFrame (269ms)
    ✔ Should revert when amount > maxAmountDuringTimeframe (94ms)
    ✔ Should set withdrawnDuringTimeframe == amount when amount <= maxAmountDuringTimeframe (146ms)
    ✔ Should transfer pmx to the recipient and reduce contract balance (72ms)
    ✔ Should transfer NATIVE_CURRENCY to the recipient and reduce contract balance (40ms)
    ✔ Should emit TransferFromTreasury event if transferFromTreasury is successful (53ms)
    ✔ Should set correct spendingInfo for spender after call setMaxSpendingLimit second time (63ms)

Upgradability_integration
  ✔ Should upgrade contract (7060ms)
  Beacon proxies
    ✔ should upgrade contract Bucket (296ms)
    ✔ should upgrade contract PToken (1294ms)
    ✔ should upgrade contract DebtToken (571ms)

WhiteBlackList_integration
  ✔ Should deploy (465ms)
  registerInWhitelist
    signer of a signature from a param 'bytes _sig' is not granted with a WHITELIST_ADMIN
      ✔ Should revert
    signer of a signature from a param 'bytes _sig' is granted with a WHITELIST_ADMIN
      ✔ Should register msg.sender in a whitelist (49ms)
      ✔ Should revert when msg.sender address is already in a whitelist (53ms)

ActivityRewardDistributor_unit
  initialize
    ✔ Should deploy (381ms)
    ✔ Should revert initialize when the registry is not supported (300ms)
    ✔ Should revert initialize when the PrimexDNS is not supported (283ms)
    ✔ Should revert initialize when the PMX address is not supported (319ms)
    ✔ Should revert initialize when the TraderBalanceVault is not supported (287ms)
    ✔ Should revert initialize when the WhiteBlackList is not supported (308ms)
  withdrawPmx
    ✔ Should revert if not BIG_TIMELOCK_ADMIN call withdrawPmx (61ms)
    ✔ Should revert while attempt to withdraw more than availablePMX
    ✔ Should successfully withdrawPmx (84ms)
  setupBucket & decreaseRewardPerDay
    ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call setupBucket
    ✔ Should revert if not EMERGENCY_ADMIN call decreaseRewardPerDay (55ms)
    ✔ Should transfer pmx from msg.sender in activityRewardDistributor (258ms)
    initial setup
      ✔ Should revert decreaseRewardPerDay if the passed value is greater than the current one (137ms)
      ✔ Should setup only totalReward (144ms)
      ✔ Should setup only reward per day when ptokens totalSupply is 0 (324ms)
      ✔ Should setup only reward per day when ptokens totalSupply isn't 0 (48ms)
      ✔ Should setup reward per day and totalReward (55ms)
      second setup
        ✔ Should setup reward per day and totalReward while totalReward is smaller than accumulated
bucket reward (117ms)
        ✔ Should setup reward per day and totalReward while totalReward is equal accumulated bucket
reward (153ms)
        ✔ decreaseRewardPerDay should setup reward per day 0  (97ms)
        ✔ decreaseRewardPerDay should setup a new value as the reward per day (120ms)
    updateUserActivity & updateUsersActivities
      ✔ updateUserActivity should revert if bucket is not in primex system
      ✔ updateUserActivity should revert if msg.sender is not ptoken of sended bucket
      ✔ updateUsersActivities should revert if bucket is not in primex system
      ✔ updateUsersActivities should revert if msg.sender is not ptoken of sended bucket
      ✔ updateUserActivity & updateUsersActivities should not set user's and bucket's paramms if
totalReward in bucket is 0 (133ms)
      ✔ updateUserActivity & updateUsersActivities should not set user's and bucket's paramms if
rewardPerDay in bucket is 0 (153ms)
      ✔ updateUserActivity should set bucket.isFinished is true and for next updateUserActivity not set
user's and bucket's paramms (149ms)
      ✔ updateUsersActivities should set bucket.isFinished = true and next updateUsersActivities should
not update user and bucket params (158ms)
      ✔ updateUserActivity should set user's and bucket's paramms (146ms)
      ✔ updateUsersActivities should set users' and bucket's paramms (181ms)
      ✔ updateUserActivity should set bucket.unusedTime if rewardPerToken is 0 (272ms)
    claimReward & getBucketAccumulatedReward
      ✔ getBucketAccumulatedReward should return 0 if bucket.lastUpdatedRewardTimestamp is 0
      ✔ claim should revert if the msg.sender is on the blacklist (215ms)
```

```
      ✔ reward is lender.fixedReward if pToken balance in this bucket is 0 (562ms)
      ✔ Should revert claimReward if activityRewardDistributor is paused
      ✔ claimReward should correct update state (483ms)
      ✔ getClaimableReward should return correct value (188ms)
      ✔ getClaimableReward and claimReward correct calculate array of buckets (1559ms)
    getBucketAccumulatedReward
      ✔ Should return 0 if lastUpdatedRewardTimestamp is 0
      ✔ Should return accumulatedReward (86ms)
      ✔ Should return bucket.totalReward if accumulatedReward is more then it (45ms)
      ✔ Should return accumulatedReward-bucket.unusedTime (222ms)
    pause & unpause
      ✔ Should revert if not EMERGENCY_ADMIN call pause (47ms)
      ✔ Should revert if not SMALL_TIMELOCK_ADMIN call unpause (46ms)

  BonusExecutor_unit
    searchNearestIndex
      ✔ Should return correct value when the bonusDeadline is one of the randomTimestamps
      ✔ Should return correct value when the bonusDeadline is the first timestamp of the randomTimestamps
      ✔ Should return correct value when the bonusDeadline is the last timestamp of the randomTimestamps
      ✔ Should return correct value when the randomTimestamps contains only one element which equal to
the bonusDeadline
      ✔ Should return correct value when the randomTimestamps contains only one element which less than
the bonusDeadline
      ✔ Should return correct value when the bonusDeadline does not match any timestamp of the array
    InterestIncreaser
      ✔ Should update the indexes via activateBonus when length of the updatedTimestamps is equal to zero
(94ms)
      ✔ Should update the indexes via activateBonus when last value of the updatedTimestamps plus 1 HOUR
is less than the current timestamp (119ms)
      ✔ Should not update the indexes via activateBonus when length of the updatedTimestamps isn't equal
to zero and last value of the updatedTimestamps plus 1 HOUR is greater than the current timestamp (115ms)
      ✔ Should revert if not BIG_TIMELOCK_ADMIN call setTierBonus (246ms)
      ✔ Should revert setTierBonus when bonuses length isn't equal tiers length
      ✔ setTierBonus should correct update state (269ms)
      ✔ Should revert activateBonus for not initialize in bonus executor bucket (287ms)
      ✔ Should revert if the nft contract does not support IPMXBonusNFT interface (255ms)
      ✔ Should revert if the registry does not support IAccessControl interface (245ms)
      ✔ Should revert activate when caller is not the NFT contract
      ✔ Should revert when the bonus is already activated (59ms)
      ✔ Should activate bonus and return the correct metadata (117ms)
      ✔ Should activate bonus with duration 0 and return correct metadata (118ms)
      ✔ Should revert update bonus when the contract is paused
      ✔ Should revert update bonus when the bucket is equal to zero
      ✔ Should revert update bonus  when the nftId does not match the passed one (77ms)
      ✔ Should revert updateBonus when the caller is not the pToken contract (76ms)
      ✔ Should updateBonus (289ms)
      ✔ Should not updateBonus when the bucket is equal to zero
      ✔ Should not updateBonus when the contract is paused (48ms)
      ✔ Should not updateBonus when the accumulatedAmount is equal to the maxAmount (479ms)
      ✔ Should updateBonus via the searchApproxIndex when the deadline is less than current timestamp
(344ms)
      ✔ Should not updateBonus when the deadline is less than current timestamp and the bonus has already
been updated via the searchApproxIndex (462ms)
      ✔ Should revert claim when the amount is zero
      ✔ Should revert claim when the bucket is equal to zero
      ✔ Should revert the claim when the contract is paused
      ✔ Should revert claim when the nftId does not match the passed one (89ms)
      ✔ Should not update the bonus when the accumulatedAmount is equal to the claimedAmount (520ms)
      ✔ Should claim and delete the bonus when the deadline is equal to the magic number and
claimedAmount is equal to accumulatedAmount (476ms)
      ✔ Should claim and update the bonus when the deadline is less than the current timestamp (559ms)
      ✔ Should claim and not update the bonus when the deadline is less than the current timestamp and
the approx index < lastUpdatedIndex (356ms)
      ✔ Should claim and update the bonus (445ms)
      ✔ Should claim and delete the bonus when maxAmount is equal to the claimedAmount (427ms)
      ✔ Should getAvailableAmount return 0 when bucket is equal to zero
      ✔ Should getAvailableAmount return correct accumulatedAmount when the deadline is less than current
timestamp (561ms)
      ✔ Should getAvailableAmount return correct accumulatedAmount when the deadline is equal to the max
of uint256 (726ms)
      ✔ Should getAvailableAmount return correct amount (448ms)
      ✔ Should getAvailableAmount return correct amount when the maxAmount is equal to zero (457ms)
```

```
      ✔ Should revert deactivateBonus when caller is not the NFT contract
      ✔ Should deactivateBonus (473ms)
      ✔ Should revert the if not EMERGENCY_ADMIN call pause
      ✔ Should revert the if not SMALL_TIMELOCK_ADMIN call unpause
    FeeDecreaser
      ✔ Should update the indexes via activateBonus when length of the updatedTimestamps is equal to zero
(98ms)
      ✔ Should update the indexes via activateBonus when last value of the updatedTimestamps plus 1 HOUR
is less than the current timestamp (113ms)
      ✔ Should not update the indexes via activateBonus when length of the updatedTimestamps isn't equal
to zero and last value of the updatedTimestamps plus 1 HOUR is greater than the current timestamp (101ms)
      ✔ Should revert if not BIG_TIMELOCK_ADMIN call setTierBonus (216ms)
      ✔ Should revert setTierBonus when bonuses length isn't equal tiers length
      ✔ setTierBonus should correct update state (252ms)
      ✔ Should revert activateBonus for not initialize in bonus executor bucket (264ms)
      ✔ Should revert if the nft contract does not support IPMXBonusNFT interface (227ms)
      ✔ Should revert if registry does not support IAccessControl interface (249ms)
      ✔ Should revert activate when caller is not the NFT contract
      ✔ Should revert when the bonus is already activated (58ms)
      ✔ Should activate bonus and return the correct metadata (119ms)
      ✔ Should revert update bonus when the contract is paused
      ✔ Should revert update bonus when the bucket is equal to zero
      ✔ Should revert update bonus  when the nftId does not match the passed one (71ms)
      ✔ Should revert updateBonus when the caller is not the debtToken contract (152ms)
      ✔ Should updateBonus (263ms)
      ✔ Should not updateBonus when the bucket is equal to zero
      ✔ Should not updateBonus when the contract is paused (53ms)
      ✔ Should not updateBonus when the accumulatedAmount is equal to the maxAmount (464ms)
      ✔ Should updateBonus via the searchApproxIndex when the deadline is less than current timestamp
(332ms)
      ✔ Should not updateBonus when the deadline is less than current timestamp and the bonus has already
been updated via the searchApproxIndex (452ms)
      ✔ Should revert FeeDecreaser when the amount is zero
      ✔ Should revert claim when the bucket is equal to zero
      ✔ Should revert the claim when the contract is paused
      ✔ Should revert claim when the nftId does not match the passed one (76ms)
      ✔ Should not update the bonus when the accumulatedAmount is equal to the claimedAmount (485ms)
      ✔ Should claim and delete the bonus when the deadline is equal to the magic number and
claimedAmount is equal to accumulatedAmount (378ms)
      ✔ Should claim and update the bonus when the deadline is less than the current timestamp (559ms)
      ✔ Should claim and not update the bonus when the deadline is less than the current timestamp and
the approx index < lastUpdatedIndex (376ms)
      ✔ Should claim and update the bonus (351ms)
      ✔ Should claim and delete the bonus when maxAmount is equal to the claimedAmount (323ms)
      ✔ Should getAvailableAmount return 0 when bucket is equal to zero
      ✔ Should getAvailableAmount return correct accumulatedAmount when the deadline is less than current
timestamp (574ms)
      ✔ Should getAvailableAmount return correct accumulatedAmount when the deadline is equal to the max
of uint256 (732ms)
      ✔ Should getAvailableAmount return correct amount (392ms)
      ✔ Should set max bonuses count for bucket (55ms)
      ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call setMaxBonusCount (59ms)
      ✔ Should revert when max bonuses count exceeded (487ms)
      ✔ Should reduce counter when bonus is claimed (377ms)
      ✔ Should revert deactivateBonus when caller is not the NFT contract
      ✔ Should deactivateBonus (164ms)
      ✔ Should revert the if not EMERGENCY_ADMIN call pause
      ✔ Should revert the if not SMALL_TIMELOCK_ADMIN call unpause

  EPMXToken_unit
    deploy
      ✔ Should contain tokens total supply equal to initialSupply after the contract deploy (74ms)
      ✔ Should deploy with zero recipient address and  mint total supply to deployer (65ms)
      ✔ Should deploy with non-zero recipient address and mint total supply to recipient (64ms)
      ✔ Should revert deploy if registry does not support interface (173ms)
    add addresses to white list
      ✔ Should revert addAddressToWhitelist if msg.sender doesn't have BIG_TIMELOCK_ADMIN role
      ✔ Should emit WhitelistedAddressAdded when a param 'address _address' is added to the white list
      ✔ Should revert when a param 'address _address' is already WHITELISTED
      ✔ Should revert addAddressesToWhitelist if msg.sender doesn't have BIG_TIMELOCK_ADMIN role
      ✔ Should emit WhitelistedAddressAdded when all addresses from a param 'address[] _addresses' are
added to the white list
```

```
      ✔ Should revert when some members of a param 'address[] _addresses' are already WHITELISTED
    remove addresses to white list
      ✔ Should revert removeAddressFromWhitelist if msg.sender doesn't have BIG_TIMELOCK_ADMIN role
      ✔ Should revert if a param 'address _address' is not in the white list
      ✔ Should emit WhitelistedAddressRemoved when a param 'address _address' is removed from the white
list (68ms)
      ✔ Should revert removeAddressesFromWhitelist if msg.sender doesn't have BIG_TIMELOCK_ADMIN role
(56ms)
      ✔ Should emit WhitelistedAddressRemoved when all addresses from a param 'address[] addresses' are
removed from the white list (328ms)
      ✔ Should revert if some members of a param 'address[] addresses' are not in the white list (141ms)
    transfer functions
      ✔ Should revert transfer if neither the sender nor the recipient is on the white list
      ✔ Should transfer when the sender is on the white list (65ms)
      ✔ Should transfer when the recipient is on the white list (47ms)
    burn
      ✔ Should revert burn if the sender is not on the white list
      ✔ Should burn and throw event

  InterestRateStrategy_unit
    ✔ Should calculate LAR=0 and BAR=b0 if utilizationRatio=0
    ✔ Should revert if UR > 1
    ur < urOptimal
      ✔ Should return LAR and BAR
    ur > urOptimal
      ✔ Should return LAR and BAR if ur = 1 and b1 < 0
      ✔ Should revert if b1 < 0 and there is BAR overflow
      ✔ Should calculate LAR and BAR if b1 > 0

  KeeperRewardDistributor_unit
    initialize
      ✔ Should revert if registry does not support IAccessControl interface (250ms)
      ✔ Should revert if price oracle does not support IPriceOracle interface (195ms)
      ✔ Should revert if treasury does not support ITreasury interface (191ms)
      ✔ Should revert if the PMX token does not support its interface (211ms)
      ✔ Should revert if the WhiteBlackList does not support its interface (220ms)
    set functions
      ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call setMaxGasPrice
      ✔ Should set defaultMaxGasPrice
      ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call setMaxGasAmount
      ✔ Should set maxGasAmount
      ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call setAdditionalGas
      ✔ Should set additionalGas
      ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call setOracleGasPriceTolerance
      ✔ Should set oracleGasPriceTolerance
      ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call setPmxPartInReward
      ✔ Should set pmxPartInReward
      ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call setNativePartInReward
      ✔ Should set nativePartInReward
      ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call setPositionSizeCoefficients
      ✔ Should set Position Size Coefficients
    updateReward
      ✔ Should revert when the caller is not LOM or PM (121ms)
      ✔ Should not add reward if positionSizeMultiplier is negative (304ms)
    claim
      ✔ Should not transfer tokens when user has zero balance of reward
      ✔ Should revert when contract is paused
      ✔ Should revert when the msg.sender is on the blacklist (158ms)
    pause & unpause
      ✔ Should revert if not EMERGENCY_ADMIN call pause
      ✔ Should revert if not SMALL_TIMELOCK_ADMIN call unpause

  LimitPriceCOM_unit
    canBeFilled
      ✔ Should revert when depositedAsset is borrowedAsset and depositInThirdAssetRoutes length isn't 0
      ✔ Should revert when firstAssetRoutes length is empty
    canBeFilled with exchangeRate param
      ✔ Should return false if params is empty
      ✔ Should return true if exchange rate is lower or equal to the limit price
      ✔ Should return false if exchange rate is greater than the limit price
    getLimitPrice
      ✔ Should decode limit price
```

```
  LiquidityMiningRewardDistributor_unit
    initialize
      ✔ Should revert deploy LiquidityMiningRewardDistributor if PrimexDNS address does not support
IPrimexDNS interface (254ms)
      ✔ Should revert deploy LiquidityMiningRewardDistributor if TraderBalanceVault address does not
support ITraderBalanceVault interface (251ms)
      ✔ Should revert deploy LiquidityMiningRewardDistributor if Registry address does not support
IAccessControl interface (234ms)
      ✔ Should revert deploy LiquidityMiningRewardDistributor if Treasury address does not support
ITreasury interface (244ms)
      ✔ Should revert deploy LiquidityMiningRewardDistributor if PMXToken address is not supported
(257ms)
      ✔ Should revert deploy LiquidityMiningRewardDistributor if _reinvestmentRate more than WAD (134ms)
      ✔ Should deploy LiquidityMiningRewardDistributor and set arguments
    updateBucketReward and getter getBucketInfo
      ✔ Should revert updateBucketReward if caller isn't PrimexDNS
      ✔ getBucketInfo return correct initial value
      ✔ updateBucketReward and getTotalPmxRewardForBucket return correct initial value
    addPoints and getters(getBucketInfo,getLenderAmountInMining,getLenderInfo)
      ✔ Should revert addPoints if caller isn't bucket in system
      ✔ getLenderInfo should return correct values when totalPoints amount is 0
      ✔ Should addPoints and getters return correct values (217ms)
    reinvest and getters(getBucketInfo,getLenderAmountInMining,getLenderInfo)
      ✔ Should revert reinvest if caller isn't bucket in system
      ✔ Should revert reinvest if reinvestment deadline is passed
      ✔ getLenderInfo should return correct values when liquidity mining in bucket is failed (79ms)
      ✔ Should correct update state if bucketTo2 is launched and emit ClaimedReward event (200ms)
      ✔ Should correct update state if bucketTo1 and bucketTo2 aren't launched (197ms)
    removePoints and getters(getBucketInfo,getLenderAmountInMining,getLenderInfo)
      ✔ Should revert removePoints if caller isn't bucket in system
      ✔ Should revert removePoints if _amount is more than amount in LM
      ✔ removePoints should remove part of extra reward (124ms)
      ✔ Should removePoints and getters return correct values (155ms)
      ✔ Should correct removePoints when _amount is MaxUint256 (171ms)
    claimReward and getters(getBucketInfo,getLenderAmountInMining,getLenderInfo)
      ✔ Should revert claimReward when bucket isBucketStable is false (54ms)
      ✔ Should revert claimReward if LiquidityMiningRewardDistributor is paused
      ✔ Should claimReward and getters return correct values (407ms)
      ✔ Should emit ClaimedReward event if claimReward is successful (516ms)
    withdrawFromDelistedBucket
      ✔ Should revert withdrawPmxByAdmin if caller isn't MEDIUM_TIMELOCK_ADMIN (138ms)
    pause & unpause
      ✔ Should revert if not EMERGENCY_ADMIN call pause (38ms)
      ✔ Should revert if not SMALL_TIMELOCK_ADMIN call unpause (40ms)

  PrimexNFT_unit
    ✔ tokenUri should return correct state (51ms)
    ✔ tokenUri should return correct state when nft activated (175ms)
    ✔ Should revert getNft when the id doesn't exist (52ms)
    ✔ Should return the correct uri when the token is not activated (63ms)
    ✔ Should return the correct uri when the token is activated (169ms)
    ✔ Should revert activate when id does not exist
    ✔ Should revert activate when the msg.sender is on the blacklist (279ms)
    ✔ Should revert activate when wrong caller (55ms)
    ✔ Should activate NFT (180ms)
    ✔ Should revert activate when token already activated (174ms)
    ✔ Should revert token activation when the program is paused (61ms)
    initialize
      ✔ Should revert deploy if primexDNS does not support interface (252ms)
      ✔ Should revert deploy if registry does not support interface (272ms)
    blockNft
      ✔ Should revert if not SMALL_TIMELOCK_ADMIN call blockNft (78ms)
      ✔ Should block NFT (100ms)
    unblockNft
      ✔ Should revert if not SMALL_TIMELOCK_ADMIN call blockNft (41ms)
      ✔ Should  unblock NFT (82ms)
    setExecutor
      ✔ Should revert if not BIG_TIMELOCK_ADMIN call setExecutor (40ms)
      ✔ Should revert setExecutor if executor ADDRESS_NOT_SUPPORTED (131ms)
      ✔ Should setExecutor
    mint with signature
```

```
    ✔ Should mint and returns the correct meta data and minter's balanceOf (59ms)
    ✔ Should mint NFT to recipient when message sender isn't recipient (61ms)
    ✔ Should revert mint when message signer doesn't have NFT_MINTER role (56ms)
    ✔ Should revert mint when msg.sender is on the blacklist (143ms)
    ✔ Should revert mint when message isn't that owner sign (59ms)
    ✔ Should revert mint when chainId doesn't match
    ✔ Should revert double mint by one signature (74ms)
  mint by minter
    ✔ Should mint and returns the correct meta data and recipient balanceOf (70ms)
    ✔ Should revert mint when msg.sender isn't minter (44ms)
    ✔ Should revert mint when chainId doesn't match (43ms)
    ✔ Should revert double mint with the same id (69ms)
  pause & unpause
    ✔ Should revert if not EMERGENCY_ADMIN call pause
    ✔ Should revert if not SMALL_TIMELOCK_ADMIN call unpause (54ms)

PMXPriceFeed_unit
  constructor
    ✔ Should set var
    ✔ Should revert deploy if address not supported (52ms)
  setAnswer
    ✔ Should revert if not BIG_TIMELOCK_ADMIN call setAnswer
    ✔ Should update values (42ms)

PositionManager_unit
  setOracleTolerableLimitMultiplier
    ✔ Should allow only MEDIUM_TIMELOCK_ADMIN to setOracleTolerableLimitMultiplier
    ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call setOracleTolerableLimitMultiplier
    ✔ Should not setOracleTolerableLimitMultiplier if newMultiplier < WAD or newMultiplier >= 10 WAD
  setSpotTradingRewardDistributor
    ✔ Should allow BIG_TIMELOCK_ADMIN to setSpotTradingRewardDistributor
    ✔ Should revert if not BIG_TIMELOCK_ADMIN call setSpotTradingRewardDistributor
    ✔ Should setSpotTradingRewardDistributor to zero address

PriceOracle_unit
  initialize
    ✔ Storage
    ✔ Should revert if registry does not support IAccessControl interface (196ms)
  setPairVolatility
    ✔ Should setPairVolatility
    ✔ Should revert if not SMALL_TIMELOCK_ADMIN call setPairVolatility
    ✔ Should revert when one of the assets is equal to zero
    ✔ Should revert when the asset addresses are identical
    ✔ Should revert when pairVolatilities is not correct
  increasePairVolatility
    ✔ Should increasePairVolatility (45ms)
    ✔ Should revert if not EMERGENCY_ADMIN call increasePairVolatility
    ✔ Should revert when one of the assets is equal to zero
    ✔ Should revert when the asset addresses are identical
    ✔ Should revert when pairVolatilities is not correct
  updatePriceFeed()
    ✔ Should revert not BIG_TIMELOCK_ADMIN call updatePriceFeed (157ms)
    ✔ Should revert if token addresses in pair to add are identical (130ms)
    ✔ Should add a price feed if token addresses in pair are different (131ms)
  getDirectPriceFeed()
    ✔ Should revert if token addresses in pair are identical
    ✔ Should revert if no price feed found
    ✔ Should get a price feed if token addresses in pair are different and the price feed exists
  getExchangeRate()
    ✔ tokenA/tokenB - price feed exists, direction = isForward, returns rate in 10**18 decimals (122ms)
    ✔ tokenB/tokenA - price feed exists, direction = !isForward, returns rate in 10**18 decimals
(118ms)
    ✔ tokenA/tokenB - price feed does not exist, tokenA/eth - price feed exists, tokenB/eth - price
feed exists, direction = isForward, returns rate in 10**18 decimals (175ms)
    ✔ should revert if basePrice from oracle is negative (196ms)
    ✔ should revert if quotePrice from oracle is negative (183ms)
    ✔ Should revert if no price feed found
  updateVolatilityFeed()
    ✔ Should revert if msg.sender is not granted with a role MEDIUM_TIMELOCK_ADMIN (181ms)
    ✔ Should revert if token addresses in pair to add are identical
    ✔ Should add a volatility feed if token addresses in pair are different
  getOracleVolatilityFeed()
```

```
      ✔ Should revert if token addresses in pair are identical
      ✔ Should revert if no volatility feed found
      ✔ Should get a volatility feed if token addresses in pair are different and the price feed exists
    getOracleVolatility
      ✔ Should return zero if no chainLink volatility feed
      ✔ Should return correct value of pairVolatility from chainLink volatility feed in 10**18 decimals
(136ms)
    getVolatilityRate
      ✔ Should return hardcoded value of pairVolatility when pairVolatility >= oraclePairVolatility
(156ms)
      ✔ Should return oraclePairVolatility when pairVolatility < oraclePairVolatility (207ms)

  PrimexPricingLibrary_unit
    getOracleAmountsOut
      ✔ Should revert if priceOracle does not support IPriceOracle interface (161ms)
      ✔ Should return amountAssetA if tokens are identical
      ✔ Should return amount according to exchange rate (171ms)
    getAmountOut
      ✔ Should revert when assets are identical
      ✔ Should revert when _routes is empty list
      ✔ Should revert if primexDNS does not support IPrimexDNS interface (435ms)
    getDepositedAmountInBorrowed
      ✔ Should revert if primexDNS does not support IPrimexDNS interface (274ms)
      ✔ Should revert if priceOracle does not support IPriceOracle interface (158ms)
      ✔ Should revert assets are identical and routes is not empty (62ms)
    multiSwap
      ✔ Should revert if primexDNS does not support IPrimexDNS interface (182ms)
      ✔ Should revert if priceOracle does not support IPriceOracle interface (724ms)
      ✔ Should revert if routes length is 0
      ✔ Should revert if sum of shares is 0 (71ms)
    getLiquidationPrice
      ✔ Should revert if positionAsset is zero address
      ✔ Should revert if positionAsset is not allowed (60ms)
    getLiquidationPriceByOrder
      ✔ Should revert if positionAsset is zero address
      ✔ Should revert if positionAsset is not allowed (70ms)

  PrimexProxyAdmin_unit
    constructor
      ✔ Should revert initialize when the registry is not supported (127ms)
      ✔ Should set right vars
    changeProxyAdmin
      ✔ Should revert not BIG_TIMELOCK_ADMIN call changeProxyAdmin
      ✔ Should transfer pmx from msg.sender in activityRewardDistributor
    upgrade
      ✔ Should revert not BIG_TIMELOCK_ADMIN call upgrade
      ✔ Should transfer pmx from msg.sender in activityRewardDistributor
    upgradeAndCall
      ✔ Should revert not BIG_TIMELOCK_ADMIN call upgradeAndCall
      ✔ Should transfer pmx from msg.sender in activityRewardDistributor

  PrimexRegsitry_unit
    deploy
      ✔ Admin of SMALL_TIMELOCK_ADMIN is MEDIUM_TIMELOCK_ADMIN
      ✔ Admin of EMERGENCY_ADMIN is SMALL_TIMELOCK_ADMIN
    setRoleAdmin
      ✔ Should revert if not BIG_TIMELOCK_ADMIN call setRoleAdmin (87ms)
      ✔ Should set new admin role

  PrimexTimelock_unit
    ✔ updateDelay is not supported
    ✔ schedule revert if timelock is paused
    ✔ scheduleBatch revert if timelock is paused
    ✔ execute revert if timelock is paused
    ✔ executeBatch revert if timelock is paused
    constructor
      ✔ Should deploy (77ms)
      ✔ Should revert initialize when the registry is not supported (145ms)
    cancel
      ✔ Should revert if the caller doesn't have CANCELLER_ROLE (181ms)
      ✔ Should cancel operation and don't give to caller CANCELLER_ROLE if caller has GUARDIAN_ADMIN
(169ms)
```

```
  pause & unpause
    ✔ Should revert if not GUARDIAN_ADMIN call pause (57ms)
    ✔ Should revert if not GUARDIAN_ADMIN call unpause (42ms)

Redeemer_unit
  deploy
    ✔ Should revert when earlyPmx is not supported (129ms)
    ✔ Should revert when pmx is not supported (139ms)
    ✔ Should revert when registry is not supported (144ms)
    ✔ Should deploy (432ms)
  changeRate
    ✔ Should revert if not BIG_TIMELOCK_ADMIN call changeRate
    ✔ Should change rate
  pause and unpause
    ✔ Should revert if not EMERGENCY_ADMIN call pause
    ✔ Should revert if not SMALL_TIMELOCK_ADMIN call unpause

ReferralProgram_unit
  setReferrers
    ✔ Should revert if caller is not granted with MEDIUM_TIMELOCK_ADMIN
    ✔ Should set referrers
  setReferrals
    ✔ Should revert if caller is not granted with MEDIUM_TIMELOCK_ADMIN
    ✔ Should set referrals of particular referrer (43ms)
    ✔ Should set referrer for a particular referral

  Reserve_unit
    initialize
      ✔ Should revert initialize when the dns is not supported (229ms)
    paybackPermanentLoss
      ✔ Should revert if Reserve contract is paused (163ms)
      ✔ Should paybackPermanentLoss if bucket is active (259ms)
      ✔ Should paybackPermanentLoss if bucket is frozen (253ms)
      ✔ Should revert if bucket is not found in dnsBucket (242ms)
      ✔ Should revert if amount to burn is zero (109ms)
      ✔ Should set (burnAmount == permanentLoss) if (permanentLoss < balance of Reserve contract) (103ms)
      ✔ Should set (burnAmount == balance of Reserve contract) if (permanentLoss > balance of Reserve
contract) (103ms)
    payBonus
      ✔ Should revert when caller is not pToken or DebtToken
      ✔ Should payBonus when caller is feeDecreaser (42ms)
      ✔ Should payBonus when caller is interestIncreaser (164ms)
    pause & unpause
      ✔ Should revert if not EMERGENCY_ADMIN call pause (41ms)
      ✔ Should revert if not SMALL_TIMELOCK_ADMIN call unpause (40ms)
    transferToTreasury
      ✔ Should revert if Reserve contract is paused (54ms)
      ✔ Should revert if called by non MEDIUM_TIMELOCK_ADMIN (44ms)
    setTransferRestrictions
      ✔ Should revert if called by non MEDIUM_TIMELOCK_ADMIN (42ms)

  SpotTradingRewardDistributor_unit
    initialize
      ✔ Should deploy (192ms)
      ✔ Should revert initialize when the registry is not supported (170ms)
      ✔ Should revert initialize when the periodDuration is zero (74ms)
      ✔ Should revert initialize when the priceOracle is not supported (176ms)
      ✔ Should revert initialize when the traderBalanceVault is not supported (185ms)
      ✔ Should revert initialize when the PMX address is not supported (194ms)
      ✔ Should revert initialize when the Treasury address is not supported (183ms)
    updateTraderActivity
      ✔ Should revert if caller is not granted with PM_ROLE (134ms)
      ✔ Should allow to update trader activity if caller is granted with PM_ROLE (41ms)
      ✔ Should not set trader activity and total activity if rewardPerPeriod is zero (42ms)
      ✔ Should not set trader activity and total activity if rewardPerPeriod > availablePMX (45ms)
      ✔ Should not set trader activity and total activity if not sufficient availablePMX top-up (46ms)
      ✔ Should set trader activity and total activity if rewardPerPeriod <= availablePMX (51ms)
      ✔ Should add a period number to periodsWithTraderActivity array and should not duplicate period
numbers (128ms)
      ✔ Should not set totalReward for the period if there is not enough availablePMX on a contract
balance (88ms)
      ✔ Should update trader activity if it was in the same period (129ms)
```

```
    claimReward
      ✔ Should emit SpotTradingClaimReward event if claim is successful (75ms)
      ✔ Should revert if a trader does not have any activity (79ms)
      ✔ Should emit SpotTradingClaimReward event if claim is successful from several periods (112ms)
      ✔ Should revert if spotTradingRewardDistributor is paused
      ✔ Should not remove period number from trader's periodsWithTraderActivity if it is equal to the
current period number (111ms)
      ✔ Should remove all period numbers from trader's periodsWithTraderActivity if no activity during
the current period (110ms)
      ✔ Should revert if amount of reward is zero (254ms)
    setRewardPerPeriod
      ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call setRewardPerPeriod (284ms)
      ✔ Should set reward
    decreaseRewardPerPeriod
      ✔ Should revert if not EMERGENCY_ADMIN call decreaseRewardPerPeriod (50ms)
      ✔ Should revert if the current rewardPerPeriod is less than the passed one
      ✔ Should successfully set reward per period
    getSpotTraderActivity
      ✔ Should get spot trader activity (62ms)
    getPeriodInfo
      ✔ Should get information for the corresponding period when totalReward is 0
      ✔ Should get information for the corresponding period (70ms)
      ✔ Should return totalReward = 0 if rewardPerPeriod is not set
    withdrawPmx
      ✔ Should revert if not BIG_TIMELOCK_ADMIN call withdrawPmx (43ms)
      ✔ Should revert while attempt to withdraw more than availablePMX
    pause & unpause
      ✔ Should revert if not EMERGENCY_ADMIN call pause (49ms)
      ✔ Should revert if not SMALL_TIMELOCK_ADMIN call unpause (41ms)

  SwapManager_unit
    constructor
      ✔ Should deploy (91ms)
      ✔ Should revert deploy when registry address not supported (145ms)
      ✔ Should revert deploy when dns address not supported (147ms)
      ✔ Should revert deploy when traderBalanceVault address not supported (133ms)
      ✔ Should revert deploy when priceOracle address not supported (561ms)
      ✔ Should revert deploy when WhiteBlackList address not supported (166ms)
    setSwapRate
      ✔ change swapRate
      ✔ Should revert if not BIG_TIMELOCK_ADMIN call setSwapRate
    setSwapRateInPmx
      ✔ change swapRateInPmx if called owner
      ✔ Should revert if not BIG_TIMELOCK_ADMIN call setSwapRateInPmx
    pause & unpause
      ✔ Should revert if not EMERGENCY_ADMIN call pause
      ✔ Should revert if not SMALL_TIMELOCK_ADMIN call unpause

  TakeProfitStopLossCCM_unit
    canBeClosed
      ✔ Should return false when params are empty
    getTakeProfitStopLossPrices
      ✔ Should decode TakeProfit, StopLoss price

  TraderBalanceVault_unit
    initialize
      ✔ Should deploy 'TraderBalanceVault' contract with correct registry address
      ✔ Should revert deploy if registry address does not support 'IAccessControl' interface (185ms)
    deposit()
      ✔ Should revert when the traderBalanceVault is paused
      ✔ Should revert when the msg.sender is on the blacklist (150ms)
      ✔ Should revert when a param 'uint256 _amount' is 0
      ✔ Should revert when deposit asset returns incorrect decimals (174ms)
      ✔ Should revert when an asset is native currency and the msg.value amount is zero
      ✔ Should revert when an asset is native currency and the msg.value > 0, but an amount > 0 too
      ✔ Should revert when an asset is erc-20 token, amount > 0, but the msg.value > 0 too
      ✔ Should emit Deposit event if an asset is native currency and deposit is successful
      ✔ Should emit Deposit event when deposit was called via the receive function (180ms)
      ✔ Should emit Deposit event if deposit is successful
    withdraw()
      ✔ Should revert when the msg.sender is on the blacklist (180ms)
      ✔ Should revert when a param 'uint256 _amount' is 0
```

```
    ✔ Should revert when amount to withdraw exceeds balance
    ✔ Should emit Withdraw event if withdraw is successful (51ms)
    ✔ Should emit Withdraw event if an asset is native currency and withdraw is successful
  topUpAvailableBalance()
    ✔ Should revert when a msg.sender is not granted with VAULT_ACCESS_ROLE (416ms)
  depositAndLock()
    ✔ Should revert when a msg.sender is not granted with VAULT_ACCESS_ROLE (461ms)
    ✔ Should revert when a param 'uint256 _amount' is 0
    ✔ Should revert when deposit asset returns incorrect decimals (156ms)
    ✔ Should emit Deposit event if deposit and lock is successful
  lockAsset()
    ✔ Should revert when a msg.sender is not granted with VAULT_ACCESS_ROLE (367ms)
    ✔ Should revert if OpenType is not OPEN_BY_ORDER and depositedAmount exceeds availableBalance
  unlockAsset()
    ✔ Should revert when a msg.sender is not granted with VAULT_ACCESS_ROLE (371ms)
  pause & unpause
    ✔ Should revert if not EMERGENCY_ADMIN call pause (131ms)
    ✔ Should revert if not SMALL_TIMELOCK_ADMIN call unpause (140ms)

TrailingStopCCM_unit
  constructor
    ✔ Should initialize with correct values
    ✔ Should revert when initialized with wrong priceOracle address (75ms)
  canBeClosed
    ✔ Should return false when params are empty
    ✔ Should return false when additionalParams are empty
    ✔ Should revert when lowPriceRoundNumber < highPriceRoundNumber

Treasury_unit
  initialize
    ✔ Should deploy the Treasury contract if registry supports IAccessControl interface (98ms)
    ✔ Should revert if registry does not support IAccessControl interface (170ms)
  transferFromTreasury
    ✔ Should revert when function is on Pause
    ✔ Should revert when transfer amount equal zero
    ✔ Should revert when transfer amount > maxAmountPerTransfer
    ✔ Should revert when minTimeBetweenTransfers is not reached
    ✔ Should revert when amount > maxTotalAmount
    ✔ Should revert when amount of native token > balance on Treasury
    ✔ Should revert when amount > balance on Treasury (38ms)
  setMaxSpendingLimit
    ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call setMaxSpendingLimit (43ms)
    ✔ Should revert if timeframeDuration is zero
    ✔ Should set MaxSpendingLimit
  decreaseLimits
    ✔ Should revert if not SMALL_TIMELOCK_ADMIN call decreaseLimits (41ms)
    ✔ Should revert when newMaxTotalAmount > maxTotalAmount
    ✔ Should revert when newMaxAmountPerTransfer > maxAmountPerTransfer
    ✔ Should revert when newMaxPercentPerTransfer > maxPercentPerTransfer
    ✔ Should revert when newMinTimeBetweenTransfers < minTimeBetweenTransfers
    ✔ Should revert when newTimeframeDuration < timeframeDuration
    ✔ Should revert when newMaxAmountDuringTimeframe > maxAmountDuringTimeframe
    ✔ Should set new spending Limits (49ms)
  canTransferByTime
    ✔ Should revert if spender is not valid
    ✔ Should return true when lastWithdrawalTimestamp + minTimeBetweenTransfers < current timestamp
    ✔ Should return false when lastWithdrawalTimestamp + minTimeBetweenTransfers >= current timestamp
  pause and unpause
    ✔ Should revert not EMERGENCY_ADMIN call pause (47ms)
    ✔ pause can call only EMERGENCY_ADMIN
    ✔ Should revert if not SMALL_TIMELOCK_ADMIN call unpause (42ms)
    ✔ Only caller with SMALL_TIMELOCK_ADMIN role can set unpause

WhiteBlackList_unit
  WhiteBlackList
    initialize
      ✔ Should initialize if contract is not initialized
      ✔ Should revert if param 'address _registry' does not support IAccessControlUpgradeable (123ms)
      ✔ Should revert if contract is already initialized (92ms)
    addAddressToBlacklist
      ✔ Should revert if msg.sender is not granted with a BIG_TIMELOCK_ADMIN (139ms)
      ✔ Should revert if a param 'address _address' is already BLACKLISTED (38ms)
```

```
      ✔ Should revert if a param 'address _address' is already WHITELISTED (38ms)
      ✔ Should emit BlacklistedAddressAdded when a param 'address _address' is added to the black list
    addAddressToWhitelist
      ✔ Should emit WhitelistedAddressAdded when a param 'address _address' is added to the black list
(38ms)
  initialize
    ✔ Should initialize if contract is not initialized
    ✔ Should revert if param 'address _registry' does not support IAccessControlUpgradeable (121ms)
    ✔ Should revert if contract is already initialized
  registerInWhitelist
    ✔ Should whitelist an address of msg.sender if a signer granted with a BIG_TIMELOCK_ADMIN
    ✔ Should revert if a signer is not granted with a BIG_TIMELOCK_ADMIN (54ms)
    ✔ Should revert if an address of a msg.sender is already WHITELISTED (49ms)
    ✔ Should revert if an address of a msg.sender is already BLACKLISTED (43ms)
  addAddressToWhitelist
    ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call addAddressToWhitelist (51ms)
    ✔ Should emit WhitelistedAddressAdded when a param 'address _address' is added to the white list
    ✔ Should revert when a param 'address _address' is already WHITELISTED
    ✔ Should revert when a param 'address _address' is already BLACKLISTED (39ms)
  addAddressesToWhitelist
    ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call addAddressesToWhitelist (51ms)
    ✔ Should emit WhitelistedAddressAdded when all addresses from a param 'address[] _addresses' are
added to the white list (38ms)
    ✔ Should revert when some members of a param 'address[] _addresses' are already WHITELISTED (38ms)
  removeAddressFromWhitelist
    ✔ Should revert if not BIG_TIMELOCK_ADMIN call removeAddressFromWhitelist (48ms)
    ✔ Should revert if a param 'address _address' is not in the white list (43ms)
    ✔ Should emit WhitelistedAddressRemoved when a param 'address _address' is removed from the white
list (46ms)
  removeAddressesFromWhitelist
    ✔ Should revert if not BIG_TIMELOCK_ADMIN call removeAddressesFromWhitelist (49ms)
    ✔ Should emit WhitelistedAddressRemoved when all addresses from a param 'address[] addresses' are
removed from the white list (53ms)
    ✔ Should revert if some members of a param 'address[] addresses' are not in the white list (50ms)
  addAddressToBlacklist
    ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call addAddressToBlacklist (50ms)
    ✔ Should revert if a param 'address _address' is already BLACKLISTED (39ms)
    ✔ Should emit BlacklistedAddressAdded when a param 'address _address' is added to the black list
  addAddressesToBlacklist
    ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call addAddressesToBlacklist (52ms)
    ✔ Should emit BlacklistedAddressAdded when all addresses from a param 'address[] _addresses' are
added to the black list (38ms)
    ✔ Should revert when some members of a param 'address[] _addresses' are already BLACKLISTED (39ms)
  removeAddressFromBlacklist
    ✔ Should revert if not SMALL_TIMELOCK_ADMIN call removeAddressFromBlacklist (62ms)
    ✔ Should revert if a param 'address _address' is not in the black list (162ms)
    ✔ Should emit BlacklistedAddressRemoved when a param 'address _address' is removed from the black
list (51ms)
  removeAddressesFromBlacklist
    ✔ Should revert if not MEDIUM_TIMELOCK_ADMIN call removeAddressesFromBlacklist (54ms)
    ✔ Should emit BlacklistedAddressRemoved when all addresses from a param 'address[] addresses' are
removed from the black list (67ms)
    ✔ Should revert if some members of a param 'address[] addresses' are not in the black list (48ms)
  getAccessType
    ✔ Should return AccessType.UNLISTED when removeAddressFromWhitelist() (227ms)
    ✔ Should return AccessType.UNLISTED when removeAddressFromBlacklist() (41ms)
    ✔ Should return AccessType.WHITELISTED when addAddressToWhitelist()
    ✔ Should return AccessType.BLACKLISTED when addAddressToBlacklist()


  1789 passing (18m)
  4 pending

Done in 1330.15s.
```

Code Coverage

We followed the README and run the following commands:

```
COVERAGE=true
```

```
cd src
yarn hardhat coverage
```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	95.65	88.98	90.57	94.19	
BatchManager.sol	100	95.24	100	100	
Constants.sol	100	100	100	100	
DexAdapter.sol	98.05	95.31	100	98.13	131,192,194
EPMXToken.sol	94.12	92.86	90	95.45	77
InterestRateStrategy.sol	100	75	100	100	
PMXPriceFeed.sol	82.35	66.67	91.67	88.46	89,127,139
PMXToken.sol	100	75	100	100	
PrimexProxyAdmin.sol	100	100	100	100	
PrimexRegistry.sol	100	100	100	100	
PrimexTimelock.sol	93.33	87.5	90	94.12	138
PrimexUpkeep.sol	98.28	80	90	89.53	... 329,384,385
Redeemer.sol	77.78	71.43	75	85.71	71,78
SwapManager.sol	96.55	90	90	97.5	138
UniswapInterfaceMulticall.sol	0	100	0	0	... 36,37,38,39
contracts/ActivityRewardDistributor/	97.65	89.47	95.24	97.06	
ActivityRewardDistributor.sol	97.65	89.47	95.24	97.06	229,257,258,259
ActivityRewardDistributorStorage.sol	100	100	100	100	
IActivityRewardDistributor.sol	100	100	100	100	
IActivityRewardDistributorStorage.sol	100	100	100	100	
contracts/BonusExecutor/	92.86	83.33	97.22	95.39	
BonusExecutor.sol	90	70	87.5	94.12	73
FeeDecreaser.sol	89.66	78.57	100	90.63	48,49,50

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
FeeExecutor.sol	96.55	90.74	100	100	
FeeExecutorStorage.sol	100	100	100	100	
IBonusExecutor.sol	100	100	100	100	
IFeeExecutor.sol	100	100	100	100	
IFeeExecutorStorage.sol	100	100	100	100	
InterestIncreaser.sol	89.66	78.57	100	90.63	53,54,55
contracts/Bucket/	97.55	86.54	98.04	98.54	
Bucket.sol	97.86	87.5	100	98.8	536,537,601
BucketStorage.sol	100	100	100	100	
BucketsFactory.sol	94.12	75	85.71	95.65	144
IBucket.sol	100	100	100	100	
IBucketStorage.sol	100	100	100	100	
IBucketsFactory.sol	100	100	100	100	
contracts/DebtToken/	89.06	70.59	100	90	
DebtToken.sol	88.14	71.88	100	89.23	... 149,150,161
DebtTokenFactory.sol	100	50	100	100	
DebtTokenStorage.sol	100	100	100	100	
IDebtToken.sol	100	100	100	100	
IDebtTokenStorage.sol	100	100	100	100	
IDebtTokensFactory.sol	100	100	100	100	
contracts/KeeperRewardDistributor/	96.77	89.13	93.75	96.88	
IKeeperRewardDistributor.sol	100	100	100	100	
IKeeperRewardDistributorStorage.sol	100	100	100	100	
KeeperRewardDistributor.sol	96.77	89.13	93.75	96.88	78,184
KeeperRewardDistributorStorage.sol	100	100	100	100	
contracts/LimitOrderManager/	99.08	83.33	96	99.26	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
ILimitOrderManager.sol	100	100	100	100	
ILimitOrderManagerStorage.sol	100	100	100	100	
LimitOrderManager.sol	99.08	83.33	96	99.26	340
LimitOrderManagerStorage.sol	100	100	100	100	
contracts/LiquidityMiningRewardDistributor/	96.67	75	88.89	98	
ILiquidityMiningRewardDistributor.sol	100	100	100	100	
ILiquidityMiningRewardDistributorStorage.sol	100	100	100	100	
LiquidityMiningRewardDistributor.sol	96.67	75	88.89	98	162,273
LiquidityMiningRewardDistributorStorage.sol	100	100	100	100	
contracts/PMXBonusNFT/	93.94	86.67	88.89	95.83	
IPMXBonusNFT.sol	100	100	100	100	
IPMXBonusNFTStorage.sol	100	100	100	100	
PMXBonusNFT.sol	93.94	86.67	88.89	95.83	90,186
PMXBonusNFTStorage.sol	100	100	100	100	
contracts/PToken/	94.5	86.21	100	95.49	
IPToken.sol	100	100	100	100	
IPTokenStorage.sol	100	100	100	100	
IPTokensFactory.sol	100	100	100	100	
PToken.sol	94.23	87.5	100	95.31	... 190,393,408
PTokenStorage.sol	100	100	100	100	
PTokensFactory.sol	100	50	100	100	
contracts/PositionManager/	99.22	72.22	100	99.41	
IBasePositionManager.sol	100	100	100	100	
IPositionManager.sol	100	100	100	100	
IPositionManagerStorage.sol	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
PositionManager.sol	99.22	72.22	100	99.41	429
PositionManagerStorage.sol	100	100	100	100	
contracts/PriceOracle/	100	89.29	100	100	
IPriceOracle.sol	100	100	100	100	
IPriceOracleStorage.sol	100	100	100	100	
PriceOracle.sol	100	89.29	100	100	
PriceOracleStorage.sol	100	100	100	100	
contracts/PrimexDNS/	100	93.75	100	100	
IPrimexDNS.sol	100	100	100	100	
IPrimexDNSStorage.sol	100	100	100	100	
PrimexDNS.sol	100	93.75	100	100	
PrimexDNSStorage.sol	100	100	100	100	
contracts/ReferralProgram/	90.91	70	77.78	92.59	
IReferralProgram.sol	100	100	100	100	
IReferralProgramStorage.sol	100	100	100	100	
ReferralProgram.sol	90.91	70	77.78	92.59	41,107
ReferralProgramStorage.sol	100	100	100	100	
contracts/Reserve/	100	85	100	100	
IReserve.sol	100	100	100	100	
IReserveStorage.sol	100	100	100	100	
Reserve.sol	100	85	100	100	
ReserveStorage.sol	100	100	100	100	
contracts/SpotTradingRewardDistributor/	95.65	85	87.5	96.97	
ISpotTradingRewardDistributor.sol	100	100	100	100	
ISpotTradingRewardDistributorStorage.sol	100	100	100	100	
SpotTradingRewardDistributor.sol	95.65	85	87.5	96.97	103,211

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
SpotTradingRewardDistributorStorage.sol	100	100	100	100	
contracts/TestnetServices/	45.63	43.42	57.58	45.83	
BalancerBotLens.sol	0	0	0	0	... 58,61,62,63
CurveBotLens.sol	100	92.86	100	100	
PriceFeedUpdaterTestService.sol	100	100	100	100	
PrimexAggregatorV3TestService.sol	64.71	50	69.23	76.92	... 91,95,96,98
SynchronizationBotLens.sol	0	0	0	0	... 119,120,122
SynchronizationBotLensQuickswapTestService.sol	0	0	0	0	... 40,47,48,49
SynchronizationBotLensUniV3TestService.sol	0	0	0	0	... 40,47,48,49
contracts/TestnetServices/interfaces/	100	100	100	100	
IBalancerBotLens.sol	100	100	100	100	
ICurveBotLens.sol	100	100	100	100	
IPrimexAggregatorV3TestService.sol	100	100	100	100	
ISynchronizationBotLens.sol	100	100	100	100	
ISynchronizationBotLensQuickswapTestService.sol	100	100	100	100	
ISynchronizationBotLensUniV3TestService.sol	100	100	100	100	
contracts/TraderBalanceVault/	97.78	85.19	92.86	98.39	
ITraderBalanceVault.sol	100	100	100	100	
ITraderBalanceVaultStorage.sol	100	100	100	100	
TraderBalanceVault.sol	97.78	85.19	92.86	98.39	195
TraderBalanceVaultStorage.sol	100	100	100	100	
contracts/Treasury/	100	87.5	100	100	
ITreasury.sol	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
ITreasuryStorage.sol	100	100	100	100	
Treasury.sol	100	87.5	100	100	
TreasuryStorage.sol	100	100	100	100	
contracts/WhiteBlackList/WhiteBlackList/	100	83.33	100	100	
IWhiteBlackList.sol	100	100	100	100	
WhiteBlackList.sol	100	83.33	100	100	
contracts/WhiteBlackList/WhiteBlackListBase/	100	80	100	100	
WhiteBlackListBase.sol	100	80	100	100	
contracts/WhiteBlackList/WhiteBlackListReferral/	100	87.5	100	100	
IWhiteBlackListReferral.sol	100	100	100	100	
WhiteBlackListReferral.sol	100	87.5	100	100	
WhiteBlackListReferralStorage.sol	100	100	100	100	
contracts/conditionalManagers/	96.1	83.87	100	97.94	
LimitPriceCOM.sol	92.59	85.71	100	97.78	151
TakeProfitStopLossCCM.sol	97.14	83.33	100	97.3	169
TrailingStopCCM.sol	100	80	100	100	
contracts/hardhat-dependency-**compiler/@openzeppelin/contracts/token/ERC20/**	100	100	100	100	
IERC20.sol	100	100	100	100	
contracts/interfaces/	100	100	100	100	
IAsset.sol	100	100	100	100	
IBalancer.sol	100	100	100	100	
IBatchManager.sol	100	100	100	100	
IBestDexLens.sol	100	100	100	100	
IConditionalClosingManager.sol	100	100	100	100	
IConditionalOpeningManag	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
er.sol					
IDexAdapter.sol	100	100	100	100	
IEPMXToken.sol	100	100	100	100	
IERC20Mock.sol	100	100	100	100	
IInterestRateStrategy.sol	100	100	100	100	
ILimitPriceCOM.sol	100	100	100	100	
IPMXPriceFeed.sol	100	100	100	100	
IPausable.sol	100	100	100	100	
IPriceFeedUpdaterTestService.sol	100	100	100	100	
IPrimexLens.sol	100	100	100	100	
IPrimexPricingLibraryMock.sol	100	100	100	100	
IPrimexRegistry.sol	100	100	100	100	
IPrimexUpkeep.sol	100	100	100	100	
IRedeemer.sol	100	100	100	100	
ISwapManager.sol	100	100	100	100	
ITakeProfitStopLossCCM.sol	100	100	100	100	
ITrailingStopCCM.sol	100	100	100	100	
IWhitelist.sol	100	100	100	100	
contracts/interfaces/routers/	100	100	100	100	
ICurveCalc.sol	100	100	100	100	
ICurvePool.sol	100	100	100	100	
ICurveRegistry.sol	100	100	100	100	
ICurveRouter.sol	100	100	100	100	
contracts/lens/	90.83	87.21	85.71	90.06	
BestDexLens.sol	97.73	95	81.82	98.51	516,523
PrimexLens.sol	86.52	80.43	87.5	83.71	... 563,564,567
contracts/libraries/	97.46	92.22	100	97.36	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
Errors.sol	100	100	100	100	
LimitOrderLibrary.sol	100	95.65	100	100	
PositionLibrary.sol	99.07	94.64	100	99.39	531
PrimexPricingLibrary.sol	94.84	88.16	100	94.29	... 652,653,660
TokenTransfersLibrary.sol	100	100	100	100	
contracts/libraries/utils/	13.79	15	44.44	19.3	
BytesLib.sol	22.22	11.11	25	25	... 150,151,154
V3Path.sol	0	100	20	12.5	... 40,41,48,58
WadRayMath.sol	12.5	18.18	66.67	18.18	... 94,95,96,98
contracts/mocks/	80.56	75	51.35	86.21	
AttackerBucket.sol	0	100	66.67	66.67	22
BucketMock.sol	68.75	50	40.43	77.78	... 190,191,197
ERC20Mock.sol	88.89	87.5	80	92.31	54
MaliciousDexMock.sol	100	100	100	100	
MockNearestSearch.sol	100	50	37.5	100	
PrimexPricingLibraryMock.sol	100	100	100	100	
TokenTransfersLibraryMock.sol	100	100	66.67	100	
contracts/mocks/mocksInterfaces/	100	100	100	100	
IAttackerBucket.sol	100	100	100	100	
IBucketMock.sol	100	100	100	100	
IMaliciousDexMock.sol	100	100	100	100	
IMockNearestSearch.sol	100	100	100	100	
contracts/mocks/upgradeMocks/	100	100	100	100	
BucketV2.sol	100	100	100	100	
DebtTokenV2.sol	100	100	100	100	
FeeDecreaserV2.sol	100	100	100	100	
InterestIncreaserV2.sol	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
KeeperRewardDistributorV2.sol	100	100	100	100	
LimitOrderManagerV2.sol	100	100	100	100	
LiquidityMiningRewardDistributorV2.sol	100	100	100	100	
PMXBonusNFTV2.sol	100	100	100	100	
PTokenV2.sol	100	100	100	100	
PositionManagerV2.sol	100	100	100	100	
PriceOracleV2.sol	100	100	100	100	
PrimexDNSV2.sol	100	100	100	100	
ReferralProgramV2.sol	100	100	100	100	
ReserveV2.sol	100	100	100	100	
SpotTradingRewardDistributorV2.sol	100	100	100	100	
TraderBalanceVaultV2.sol	100	100	100	100	
TreasuryV2.sol	100	100	100	100	
WhiteBlackListReferralV2.sol	100	100	100	100	
contracts/mocks/upgradeMocksInterfaces/	100	100	100	100	
IBucketV2.sol	100	100	100	100	
IDebtTokenV2.sol	100	100	100	100	
IFeeDecreaserV2.sol	100	100	100	100	
IInterestIncreaserV2.sol	100	100	100	100	
IKeeperRewardDistributorV2.sol	100	100	100	100	
ILimitOrderManagerV2.sol	100	100	100	100	
ILiquidityMiningRewardDistributorV2.sol	100	100	100	100	
IPMXBonusNFTV2.sol	100	100	100	100	
IPTokenV2.sol	100	100	100	100	
IPositionManagerV2.sol	100	100	100	100	
IPriceOracleV2.sol	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
IPrimexDNSV2.sol	100	100	100	100	
IReferralProgramV2.sol	100	100	100	100	
IReserveV2.sol	100	100	100	100	
ISpotTradingRewardDistributorV2.sol	100	100	100	100	
ITraderBalanceVaultV2.sol	100	100	100	100	
ITreasuryV2.sol	100	100	100	100	
IWhiteBlackListReferralV2.sol	100	100	100	100	
All files	91.72	82.25	87.97	91.63	

Changelog

- 2023-08-18 - Initial report
- 2023-09-20 - fix review report
- 2023-10-12 - updated client statements
- 2023-10-19 - deployment review & repository migration update

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

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites&aspo; owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on any website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any output generated by such software.

Disclaimer

The review and this report are provided on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Quantstamp disclaims all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. You agree that your access and/or use of the report and other results of the review, including but not limited to any associated services, products, 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.