Security Review Report
NM-0361-BERACHAIN-HONEY

NETHERMIND
SECURITY

(Jan 06, 2025)

Contents

1 Executive Summary

This document presents the security review performed by Nethermind Security for Berachain Honey smart contracts. The Berachain team is implementing a fully collateralized stablecoin, `$Honey`, backed by other stablecoins and designed to provide a stable and reliable means of exchange within and outside the Berachain ecosystem.

This security review focuses exclusively on the smart contracts listed in Section 2 (*Audited Files*).

The audit was performed using (a) manual analysis of the codebase and (b) creation of test cases. **Along this document, we report** four points of attention, classified as `Informational` and `Best Practices`. The issues are summarized in Fig. 1.

This document is organized as follows. Section 2 presents the files in the scope. Section 3 summarizes the issues. Section 4 presents the system overview. Section 5 discusses the risk rating methodology. Section 6 details the issues. Section 7 discusses the documentation provided by the client for this audit. Section 8 presents the compilation, tests, and automated tests. Section 9 concludes the document.

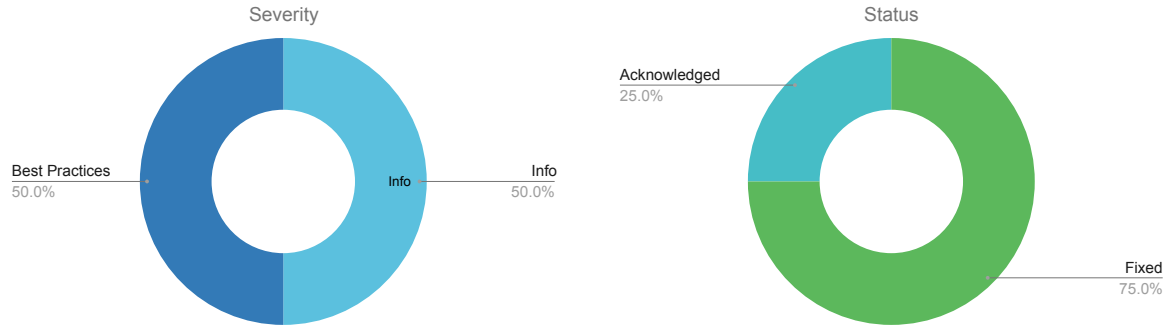

Fig. 1: Distribution of issues: Critical (0), **High** (0), **Medium** (0), **Low** (0), **Undetermined** (0), **Informational** (2), **Best Practices** (2). **Distribution of status: Fixed** (3), **Acknowledged** (1), **Mitigated** (0), **Unresolved** (0)

Summary of the Audit

Audit Type	Security Review
Initial Report	Nov 22, 2024
Response from Client	Regular responses during audit engagement
Final Report	Jan 06, 2025
Repository	contracts-monorepo
Commit (Audit)	e991b9fbaf2537f8ee5f1948a858534a3d342bcb
Commit (Final)	2b17de7b73ca65d2c1a1b430faae3311613dfc34
Documentation Assessment	High
Test Suite Assessment	High

2 Audited Files

	Contract	LoC	Comments	Ratio	Blank	Total
1	src/honey/IHoneyErrors.sol	14	13	92%	1	28
2	src/honey/IHoneyFactory.sol	20	47	235%	13	80
3	src/honey/Honey.sol	38	14	36%	13	65
4	src/honey/HoneyFactory.sol	182	69	37%	31	282
5	src/honey/CollateralVault.sol	122	95	77%	27	244
6	src/honey/VaultAdmin.sol	156	84	53%	39	278
7	src/honey/HoneyDeployer.sol	23	13	56%	7	43
	Total	**555**	**335**	**60.4%**	**131**	**1020**

3 Summary of Issues

	Finding	Severity	Update
1	Mint and redeem reverts on basket mode if one vault is paused	Info	Fixed
2	Untrusted price risk due to missing confidence interval validation	Info	Acknowledged
3	Incorrect description and comments	Best Practices	Fixed
4	Lack of input validation	Best Practices	Fixed

4 System Overview

Honey is a stablecoin backed by a diversified basket of other stablecoins. The system is composed of three main contracts: The `Honey` token, the `HoneyFactory`, and a set of `CollateralVault` contracts. In the following subsections, we present an overview of the main functionalities of each contract.

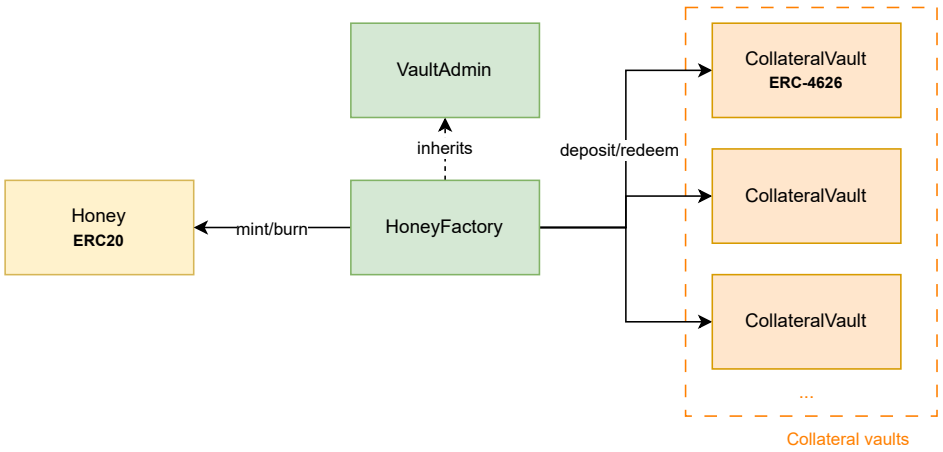

Fig. 2: Berachain Honey overview

4.1 Honey Token

The `Honey` token is an ERC-20 token representing the stablecoin of Berachain. It inherits from the Solady ERC-20 contract, overriding the `mint(...)` and `burn(...)` functions to restrict these operations to the `HoneyFactory` contract only.

```
function mint(address to, uint256 amount) external onlyFactory

function burn(address from, uint256 amount) external onlyFactory
```

4.2 HoneyFactory Contract

The `HoneyFactory` contract is responsible for the minting and redeeming of Honey tokens. It interacts with multiple `CollateralVault` contracts, each representing a different collateral asset that backs the Honey stablecoin.

4.2.1 Global and Relative Caps

The `HoneyFactory` introduces two types of optional caps to manage collateral backing.

- **Global Cap:** Defines the maximum percentage that any asset's weight can represent relative to the total value of all collateral assets in the system.
- **Relative Cap:** Sets a limit on the amount of a particular asset balance compared to another reference asset.

4.2.2 Minting Honey Tokens

Users can mint Honey tokens through the `mint(...)` function by specifying the collateral they want to deposit, the amount of collateral to deposit, and the receiver address.

```
function mint(address asset, uint256 amount, address receiver) external whenNotPaused returns (uint256 honeyToMint)
```

There are two scenarios for minting Honey:

- **Basket Mode Disabled:** The collateral asset must be pegged and not marked as bad collateral. The specified amount is deposited into the corresponding `CollateralVault`, and vault shares are minted to the contract. Honey tokens are then minted to the user at a predefined mint rate of 1:1 minus the fees.
- **Basket Mode Enabled:** Users are required to deposit all the registered assets following their current weights in the contract. The amount to be deposited for each asset is calculated, and an equivalent amount of Honey tokens is minted to the user.

4.2.3 Redeeming Honey Tokens

The `redeem(...)` function allows users to redeem their Honey tokens and receive the specified collateral asset.

```
function redeem(address asset, uint256 honeyAmount, address receiver) external whenNotPaused returns (uint256[] memory
↪  redeemed)
```

There are two scenarios for redeeming Honey:

- **Basket Mode Disabled:** The specified amount of Honey is converted into vault shares using the redemption rate, which is a 1:1 rate minus fees. The Honey tokens are then burned, the calculated share amount is redeemed from the vault, and assets are transferred to the receiver.
- **Basket Mode Enabled:** Users receive a proportional amount of all registered collateral assets based on the current weights in the system.

4.2.4 Liquidation

The `liquidate(...)` function allows users to remove bad collateral assets from the system. By depositing a good collateral asset, users can receive a bad collateral asset in return, with the conversion rate determined by the current asset prices fetched from the Pyth oracle and adjusted for a predefined liquidation rate. The integration of liquidation rates incentivizes liquidation by providing users with more value than they deposited.

```
function liquidate(address badCollateral, address goodCollateral, uint256 goodAmount)
```

4.2.5 Recapitalization

The `recapitalize(...)` function allows for the recapitalization of a vault by depositing additional collateral assets. The contract defines a minimum amount of vault shares to allow the recapitalization. Additionally, each asset has a balance threshold that should not be exceeded when calling the function.

```
function recapitalize(address asset, uint256 amount) external whenNotPaused
```

4.3 Collateral Vault

The `CollateralVault` contract represents a single collateral vault and follows the ERC-4626 standard, with overrides for the `deposit(...)`, `mint(...)`, `withdraw(...)`, and `redeem(...)` functions, ensuring they can only be called by the `HoneyFactory` contract.

```
function deposit(uint256 assets, address receiver) public override onlyFactory whenNotPaused returns (uint256)

function redeem(uint256 shares, address receiver, address owner) public override onlyFactory whenNotPaused returns
↪  (uint256)
```

Additionally, the `convertToShares(...)` and `convertToAssets(...)` functions are overridden to enforce a 1:1 rate in the vault.

5 Risk Rating Methodology

The risk rating methodology used by Nethermind Security follows the principles established by the OWASP Foundation. The severity of each finding is determined by two factors: **Likelihood** and **Impact**.

Likelihood measures how likely the finding is to be uncovered and exploited by an attacker. This factor will be one of the following values:

a) **High**: The issue is trivial to exploit and has no specific conditions that need to be met;

b) **Medium**: The issue is moderately complex and may have some conditions that need to be met;

c) **Low**: The issue is very complex and requires very specific conditions to be met.

When defining the likelihood of a finding, other factors are also considered. These can include but are not limited to motive, opportunity, exploit accessibility, ease of discovery, and ease of exploit.

Impact is a measure of the damage that may be caused if an attacker exploits the finding. This factor will be one of the following values:

a) **High**: The issue can cause significant damage, such as loss of funds or the protocol entering an unrecoverable state;

b) **Medium**: The issue can cause moderate damage, such as impacts that only affect a small group of users or only a particular part of the protocol;

c) **Low**: The issue can cause little to no damage, such as bugs that are easily recoverable or cause unexpected interactions that cause minor inconveniences.

When defining the impact of a finding, other factors are also considered. These can include but are not limited to Data/state integrity, loss of availability, financial loss, and reputation damage. After defining the likelihood and impact of an issue, the severity can be determined according to the table below.

		Severity Risk		
Impact	**High**	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Info/Best Practices	Low	Medium
	Undetermined	Undetermined	Undetermined	Undetermined
		Low	**Medium**	**High**
		Likelihood		

To address issues that do not fit a High/Medium/Low severity, Nethermind Security also uses three more finding severities: **Informational**, **Best Practices**, and **Undetermined**.

a) **Informational** findings do not pose any risk to the application, but they carry some information that the audit team intends to pass to the client formally;

b) **Best Practice** findings are used when some piece of code does not conform with smart contract development best practices;

c) **Undetermined** findings are used when we cannot predict the impact or likelihood of the issue.

6 Issues

6.1 [Info] Mint and redeem reverts on basket mode if one vault is paused

File(s): `HoneyFactory.sol`

Description: When the contract is in Basket mode, users are required to deposit and withdraw all registered assets according to their current weights. However, if any of the registered asset vaults are paused, calls to `deposit(...)` and `redeem(...)` will revert. This causes the `mint(...)` and `redeem(...)` operations on `HoneyFactory` to always fail, as these operations depend on the successful execution of deposits and redemptions for each asset in the basket.

Recommendation(s): Revise the system design to avoid breaking mint and redeem operations during the basket mode.

Status: Fixed

Update from the client: We recognize that the pausability feature was poorly described on our side and that this behavior may arise if misused. We have addressed it by improving how we computes the weights of the collateral assets used while the contract is in basket mode. Commit: e6413a3ad5651d618e53e49bd4fec2287959d25b

6.2 [Info] Untrusted price risk due to missing confidence interval validation

File(s): `HoneyFactory.sol`

Description: The `HoneyFactory` contract fetches asset prices using the Pyth oracle but does not validate the confidence level returned by the oracle. This omission may result in the contract using untrusted or unreliable price data.

According to Pyth's documentation, it is important to validate the confidence interval of price data to ensure its reliability. The price may be imprecise or unreliable when the confidence interval is too wide. In the case of the `HoneyFactory` contract, failing to check the confidence level could lead to using an inaccurate price, potentially missing out on detecting a depegged asset.

Recommendation(s): Follow the recommendations from Pyth documentation by validating the confidence interval returned by the oracle to ensure the price meets an acceptable threshold. Additionally, update the logic to handle cases where the price is considered untrusted.

Status: Acknowledged

Update from the client: Acknowledged, we plan to implement the following check: if lower/upper bound deviates more than 2 standard dev from mid value -> consider the asset depegged as a safety measure.

6.3 [Best Practices] Incorrect description and comments

File(s): HoneyFactory.sol

Description: The following functions contain incorrect descriptions or comments:

- The `setDepegOffsets(...)` function has an inaccurate description, as it referes to the behavior of the `setMaxFeedDelay(...)` function instead of describing its own functionality.
- The `createVault(...)` function contains a comment that states it sets the peg offset to the default value to check the peg status. However, the function actually sets it to the maximum value.

Recommendation(s): Update the comments within the code to reflect the implementation correctly.

Status: Fixed

Update from the client: Comments updated. Commits: 7c3aa48d2650002f1471eebb8cf32ce03e53c407 and e6413a3ad5651d618e53e49bd4fec2287959d25b

6.4 [Best Practices] Lack of input validation

File(s): HoneyFactory.sol

Description: The following functions lack validation for important inputs:

- The `initialize()` function lacks a check to ensure that the `_pythOracle` address is not `address(0)`
- The `setGlobalCap(...)` and `setRelativeCap(...)` functions do not validate the rate of the provided cap, allowing the cap to be set higher than the intended 100% maximum limit

Recommendation(s): Add a validation check in the initialize() function to ensure that the `_pythOracle` address is not `address(0)`. Additionally, a validation mechanism should be implemented in the `setGlobalCap(. . . )` and `setRelativeCap(. . . )` functions to ensure that the cap rate does not exceed 100%.

Status: Fixed

Update from the client: We have addressed the lack of check in the ‘initialize‘ function. Regarding the validation mechanism, it seems unneeded on the general case as it won’t change the behavior and the contract is already near the EIP-170 code size limit; also, we are aware of a rare scenario where a paused collateral vault may lead to a deadend that can only be overcome by setting a relative cap above 100%. Commit: e6413a3ad5651d618e53e49bd4fec2287959d25b

7 Documentation Evaluation

Software documentation refers to the written or visual information that describes the functionality, architecture, design, and implementation of software. It provides a comprehensive overview of the software system and helps users, developers, and stakeholders understand how the software works, how to use it, and how to maintain it. Software documentation can take different forms, such as user manuals, system manuals, technical specifications, requirements documents, design documents, and code comments. Software documentation is critical in software development, enabling effective communication between developers, testers, users, and other stakeholders. It helps to ensure that everyone involved in the development process has a shared understanding of the software system and its functionality. Moreover, software documentation can improve software maintenance by providing a clear and complete understanding of the software system, making it easier for developers to maintain, modify, and update the software over time. Smart contracts can use various types of software documentation. Some of the most common types include:

- Technical whitepaper: A technical whitepaper is a comprehensive document describing the smart contract's design and technical details. It includes information about the purpose of the contract, its architecture, its components, and how they interact with each other;
- User manual: A user manual is a document that provides information about how to use the smart contract. It includes step-by-step instructions on how to perform various tasks and explains the different features and functionalities of the contract;
- Code documentation: Code documentation is a document that provides details about the code of the smart contract. It includes information about the functions, variables, and classes used in the code, as well as explanations of how they work;
- API documentation: API documentation is a document that provides information about the API (Application Programming Interface) of the smart contract. It includes details about the methods, parameters, and responses that can be used to interact with the contract;
- Testing documentation: Testing documentation is a document that provides information about how the smart contract was tested. It includes details about the test cases that were used, the results of the tests, and any issues that were identified during testing;
- Audit documentation: Audit documentation includes reports, notes, and other materials related to the security audit of the smart contract. This type of documentation is critical in ensuring that the smart contract is secure and free from vulnerabilities.

These types of documentation are essential for smart contract development and maintenance. They help ensure that the contract is properly designed, implemented, and tested, and they provide a reference for developers who need to modify or maintain the contract in the future.

Remarks about Berachain Honey documentation

The `Berachain` team provided comprehensive documentation outlining the protocol's functionalities. Additionally, they effectively addressed concerns and questions raised by the Nethermind Security team during their regular calls.

8 Test Suite Evaluation

8.1 Compilation Output

```
> forge build
[] Compiling...
[] Compiling 20 files with Solc 0.8.26
[] Solc 0.8.26 finished in 7.10s
Compiler run successful!
```

8.2 Tests Output

```
> forge test
[PASS] test_GetParentBlockRootAt_Failure() (gas: 14128)
[PASS] test_GetParentBlockRootAt_Success() (gas: 18711)
[PASS] test_IsParentBlockRootAt_Failure() (gas: 9271)
[PASS] test_IsParentBlockRootAt_Success() (gas: 13389)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 1.66ms (459.08µs CPU time)

Ran 4 tests for test/base/Create2Deployer.t.sol:Create2DeployerTest
[PASS] test_DeployProxyWithCreate2() (gas: 2432477)
[PASS] test_DeployProxyWithCreate2_FailIfAlreadyDeployed() (gas: 1024285682)
[PASS] test_DeployWithCreate2() (gas: 2350973)
[PASS] test_DeployWithCreate2_FailIfAlreadyDeployed() (gas: 1024282355)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 2.48ms (1.28ms CPU time)

Ran 1 test for test/pol/POLDeployer.t.sol:POLDeployerTest
[PASS] test_DeployPOL() (gas: 15814197)
Logs:
  POLDeployer init code size 52522
  BeraChef implementation address 0xF59ed392C804898DDfCfEFBB0a32aa360765AebF
  BeraChef init code hash
  0x677948307bd0b407e503f3b99df75ac6472782b912c4863f9ac3a1acc3baa94f
  BeraChef address 0x6aE0D2beA59Af03988091a3876e5C1924b5dd726
  BlockRewardController implementation address 0x418544cD25d9bD8Ef88eF489895FA1799FDEcff1
  BlockRewardController init code hash
  0xb9454149b2eb115442d47175b0a79776848f40fdedac286b603002581e53d5ec
  BlockRewardController address 0x18f4d0a2DCC60EFDb0A08C81Fd44C67EC0F0b982
  Distributor implementation address 0xF89057eF3c925d425A7c39f1688737951438798a
  Distributor init code hash
  0x88e784a85cad1528789f2ee79188f3567a7704ca073a92a092fdedc04fc42ca9
  Distributor address 0x5A50404787565c8A9c5F14c653A09fCDA01961e0

Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 3.60ms (1.64ms CPU time)

Ran 4 tests for test/gov/GovDeployer.t.sol:GovDeployerTest
[PASS] test_GovDeployRevertAvgBlockTimeIsZero() (gas: 8602101)
[PASS] test_GovDeployRevertVoteTokenIsAddressZero() (gas: 51315)
[PASS] test_GovDeployWithNoVotesToken() (gas: 925262)
[PASS] test_GovDeployer() (gas: 9020116)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 3.92ms (2.32ms CPU time)

Ran 6 tests for test/berps/v0/TestOrders.t.sol:TestOrders
[PASS] testGetOpenLimitOrders() (gas: 399308)
[PASS] testGetOpenTrades() (gas: 603038)
[PASS] testStoreOpenLimitOrder() (gas: 236065)
[PASS] testStoreTrade() (gas: 330287)
[PASS] testUnregisterOpenLimitOrder() (gas: 190843)
[PASS] testUnregisterTrade() (gas: 267682)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 1.59ms (400.88µs CPU time)
```

```
Ran 8 tests for test/berps/utils/TestPriceUtils.t.sol:TestPriceUtils
[PASS] testCorrectSl() (gas: 10484)
[PASS] testCorrectTp() (gas: 7095)
[PASS] testCurrentPercentProfit() (gas: 5059)
[PASS] testCurrentPercentProfitRevertsIfPriceZero() (gas: 3167)
[PASS] testMaxSlDist() (gas: 3681)
[PASS] testMaxTpDist() (gas: 3596)
[PASS] testPythTriangular() (gas: 5934)
[PASS] testSimpleTriangular() (gas: 5996)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 198.42µs (126.04µs CPU time)

Ran 1 test for test/berps/v0/TestReferrals.t.sol:TestReferrals
[PASS] testReferralE2E() (gas: 392189)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 630.83µs (321.88µs CPU time)

Ran 3 tests for test/berps/integration/TestTradingFee.t.sol:TestTradingFee
[PASS] testTradingCloseMarket() (gas: 1670221)
Logs:
  honey balance: 50000000000000000000
  honey balance: 84100000003599999995

[PASS] testTradingCloseSL() (gas: 1761405)
Logs:
  honey balance: 50000000000000000000
  honey balance: 43510000000438750000

[PASS] testTradingCloseTP() (gas: 1686761)
Logs:
  honey balance: 50000000000000000000
  honey balance: 93100000004499999994

Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 5.81ms (3.58ms CPU time)

Ran 2 tests for test/berps/integration/TestTradingMisc.t.sol:TestTradingMisc
[PASS] testOpenCloseSameBlock() (gas: 1451892)
[PASS] testTradingSL() (gas: 1612156)
Logs:
  honey balance: 50000000000000000000
  honey balance: 40000000000000000000
  honey balance: 40000000000000000000
  honey balance: 48099999999693243635

Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 7.09ms (5.32ms CPU time)

Ran 5 tests for test/berps/integration/TestTradingValue.t.sol:TestTradingValue
[PASS] testRefundForDelegate() (gas: 1154389)
[PASS] testRefundForDelegateWithMulticall() (gas: 1733328)
[PASS] testRefundInMulticall() (gas: 1704826)
[PASS] testRefundNoMulticall() (gas: 1238084)
[PASS] testRefundWithMulticall() (gas: 1811065)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 14.49ms (10.99ms CPU time)

Ran 39 tests for test/pol/BeraChef.t.sol:BeraChefTest
[PASS] testFail_SetDefaultRewardAllocationWithZeroPercentageWeight() (gas: 48692)
[PASS] testFuzz_FailUpdateWhitelistedVaultsNotRegistered() (gas: 2636250)
[PASS] testFuzz_QueueANewRewardAllocation(uint32) (runs: 1024, : 311, ~: 311)
[PASS] testFuzz_SetMaxNumWeightsPerRewardAllocation(uint8) (runs: 1024, : 31033, ~: 31034)
[PASS] testFuzz_SetRewardAllocationBlockDelay(uint64) (runs: 1024, : 30276, ~: 30037)
[PASS] testFuzz_SetRewardAllocationBlockDelay_FailsIfDelayTooLarge(uint64) (runs: 1024, : 22988, ~: 22919)
[PASS] testFuzz_updateWhitelistedVaults(bool) (runs: 1024, : 38032, ~: 29310)
[PASS] test_ActivateARewardAllocation() (gas: 170165)
[PASS] test_DeleteQueuedRewardAllocationAfterActivation() (gas: 220836)
[PASS] test_EditDefaultRewardAllocationBeforeRemoveAVaultFromWhitelist() (gas: 1541259)
[PASS] test_FailIfCallerNotDistributor() (gas: 22385)
[PASS] test_FailIfInitializeAgain() (gas: 24488)
[PASS] test_FailIfInvalidRewardAllocationWeights() (gas: 39577)
[PASS] test_FailIfNotOwner() (gas: 2398017)
[PASS] test_FailIfNotWhitelistedVault() (gas: 48823)
[PASS] test_FailIfRemovingAVaultFromWhitelistInvalidatesDefaultRewardAllocation() (gas: 111393)
[PASS] test_FailIfTheNewRewardAllocationHasTooManyWeights() (gas: 664137)
[PASS] test_FailIfTheNewRewardAllocationIsNotInTheFuture() (gas: 34116)
[PASS] test_FailIfZeroRewardAllocationWeight() (gas: 43293)
```

```
[PASS] test_FailQueuedRewardAllocationExists() (gas: 112565)
[PASS] test_GetActiveRewardAllocation() (gas: 177434)
[PASS] test_GetActiveRewardAllocationReturnsDefaultRewardAllocation() (gas: 1664082)
[PASS] test_GetActiveRewardAllocationReturnsDefaultRewardAllocationAfterSetMaxWeights() (gas: 1722884)
[PASS] test_GetSetActiveRewardAllocation() (gas: 272315)
[PASS] test_OwnerIsGovernance() (gas: 17517)
[PASS] test_QueueANewRewardAllocation() (gas: 113589)
[PASS] test_QueueANewRewardAllocationWithMultipleWeights() (gas: 173953)
[PASS] test_ReturnsIfTheRewardAllocationIsNotQueued() (gas: 28378)
[PASS] test_ReturnsfStartBlockGreaterThanCurrentBlock() (gas: 125377)
[PASS] test_SetDefaultRewardAllocation() (gas: 105701)
[PASS] test_SetMaxNumWeightsPerRewardAllocation() (gas: 29015)
[PASS] test_SetMaxNumWeightsPerRewardAllocation_FailIfInvalidateDefaultRewardAllocation() (gas: 130184)
[PASS] test_SetMaxNumWeightsPerRewardAllocation_FailWithZero() (gas: 17954)
[PASS] test_SetRewardAllocationBlockDelay() (gas: 26826)
[PASS] test_UpgradeTo() (gas: 2360881)
[PASS] test_updateWhitelistedVaultMetadata() (gas: 27043)
[PASS] test_updateWhitelistedVaultMetadata_FailIfNotOwner() (gas: 18098)
[PASS] test_updateWhitelistedVaultMetadata_FailIfNotWhitelisted() (gas: 21972)
[PASS] test_updateWhitelistedVaults() (gas: 50407)
Suite result: ok. 39 passed; 0 failed; 0 skipped; finished in 128.49ms (125.44ms CPU time)

Ran 3 tests for test/berps/v0/TestVault.t.sol:TestVault
[PASS] testAssetsPnL(uint48) (runs: 1024, : 208121, ~: 208787)
[PASS] testBalanceOf() (gas: 454733)
[PASS] testDistributeReward(uint48) (runs: 1024, : 273688, ~: 276876)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 168.35ms (167.16ms CPU time)


Ran 29 tests for test/gov/BerachainGovernance.t.sol:BerachainGovernanceTest
[PASS] testFuzz_UpgradeGovernance_FailsIfNotOwner(address) (runs: 1024, : 5889373, ~: 5889373)
[PASS] testFuzz_UpgradeTimelock_FailsIfNotOwner(address) (runs: 1024, : 2334611, ~: 2334611)
[PASS] test_AcceleratedProposal() (gas: 979134)
[PASS] test_CancelProposal() (gas: 463168)
[PASS] test_CancelProposalFailsIfCallerNotProposer() (gas: 453725)
[PASS] test_CancelProposalFailsIfProposalActive() (gas: 469939)
[PASS] test_CreateFailedProposal() (gas: 523954)
[PASS] test_CreatePendingProposal() (gas: 471580)
[PASS] test_CreateSucceededProposal() (gas: 526497)
[PASS] test_ExecuteProposal() (gas: 738127)
[PASS] test_ExecuteProposal_FailsIfProposalNonExistent() (gas: 55290)
[PASS] test_ExecuteProposal_FailsIfProposalNotQueued() (gas: 452179)
[PASS] test_ExecuteProposal_FailsIfSucceededButNotQueued() (gas: 574178)
[PASS] test_GuardianCanCancelProposalWhenReady() (gas: 653604)
[PASS] test_GuardianCanCancelProposalWhenWaiting() (gas: 653095)
[PASS] test_GuardianCancelProposalFailsIfAddressDoesNotHaveRole() (gas: 661040)
[PASS] test_ProposeFailsIfInsufficienVotes() (gas: 65656)
[PASS] test_ProposeFailsWithInvalidDescription() (gas: 56421)
[PASS] test_QueueProposal() (gas: 649555)
[PASS] test_QueueProposalFailsIfProposalCanceled() (gas: 468268)
[PASS] test_QueueProposalFailsIfProposalNonExistent() (gas: 55211)
[PASS] test_QueueProposalFailsIfProposalNotSuccess() (gas: 452131)
[PASS] test_UpgradeGovernanceViaVoting() (gas: 6424465)
[PASS] test_UpgradeGovernance_FailsIfNotOwner() (gas: 5889260)
[PASS] test_UpgradeTimelockViaVoting() (gas: 2850600)
[PASS] test_UpgradeTimelockWithCallerAsTimelock() (gas: 2339925)
[PASS] test_VoteCastingFailsIfMissingVotingPower() (gas: 482365)
[PASS] test_state_WhenAcceleratedProposalAndCanceledByGuardianship() (gas: 1121246)
[PASS] test_state_WhenAcceleratedProposalAndQueued() (gas: 1114493)
Suite result: ok. 29 passed; 0 failed; 0 skipped; finished in 281.62ms (279.97ms CPU time)

Ran 32 tests for test/honey/CollateralVault.t.sol:CollateralVaultTest
[PASS] testFuzz_depositIntoUSTVault(uint256) (runs: 1024, : 129367, ~: 130263)
[PASS] testFuzz_deposit_succeedsWithCorrectSender(uint256) (runs: 1024, : 129259, ~: 129913)
[PASS] testFuzz_mintFromUSTVault(uint256) (runs: 1024, : 130901, ~: 132107)
[PASS] testFuzz_mint_succeedsWithCorrectSender(uint256) (runs: 1024, : 146493, ~: 147102)
[PASS] testFuzz_redeemFromUSTVault(uint256) (runs: 1024, : 162907, ~: 163274)
[PASS] testFuzz_redeem_failWithInsufficientAllowance(uint256) (runs: 1024, : 131648, ~: 131610)
[PASS] testFuzz_redeem_failsWithInsufficientBalance(uint256) (runs: 1024, : 29622, ~: 29622)
[PASS] testFuzz_redeem_succeedsWithCorrectSender(uint256) (runs: 1024, : 162265, ~: 162459)
[PASS] testFuzz_withdrawFromUSTVault(uint256) (runs: 1024, : 164632, ~: 164998)
[PASS] testFuzz_withdraw_failsWithIncorrectSender(uint128) (runs: 1024, : 22666, ~: 22666)
[PASS] testFuzz_withdraw_failsWithInsufficientAllowance(uint256) (runs: 1024, : 132954, ~: 132916)
```

```
[PASS] testFuzz_withdraw_failsWithInsufficientBalance(uint256) (runs: 1024, : 35267, ~: 35266)
[PASS] testFuzz_withdraw_succeedsWithCorrectSender(uint256) (runs: 1024, : 163843, ~: 164069)
[PASS] test__codesize() (gas: 75335)
[PASS] test_deposit() (gas: 129704)
[PASS] test_deposit_whileItsPaused() (gas: 52859)
[PASS] test_deposit_withOutOwner() (gas: 20343)
[PASS] test_initializeVault_failsIfZeroAsset() (gas: 145233)
[PASS] test_initializeVault_failsIfZeroFactory() (gas: 122712)
[PASS] test_mint() (gas: 131561)
[PASS] test_mint_failsWithIncorrectSender() (gas: 22473)
[PASS] test_mint_whileItsPaused() (gas: 52903)
[PASS] test_pauseVault_succeedsWithCorrectSender() (gas: 47270)
[PASS] test_pausingVault_failsIfNotFactory() (gas: 20135)
[PASS] test_redeem() (gas: 157240)
[PASS] test_redeem_whileItsPaused() (gas: 52922)
[PASS] test_redeem_withOutOwner() (gas: 20408)
[PASS] test_unpausingVault_failsIfNotFactory() (gas: 20136)
[PASS] test_unpausingVault_succeedsWithCorrectSender() (gas: 36002)
[PASS] test_vaultParams() (gas: 34953)
[PASS] test_withdraw() (gas: 158620)
[PASS] test_withdraw_whileItsPaused() (gas: 52987)
Suite result: ok. 32 passed; 0 failed; 0 skipped; finished in 490.55ms (793.76ms CPU time)

Ran 34 tests for test/honey/Honey.t.sol:HoneyTest
[PASS] testFuzz_Approve(address,uint256) (runs: 1024, : 40903, ~: 41040)
[PASS] testFuzz_Burn(uint256,uint256) (runs: 1024, : 73142, ~: 73412)
[PASS] testFuzz_Burn_FailIfInsufficientBalance(uint256,uint256) (runs: 1024, : 65730, ~: 66770)
[PASS] testFuzz_Mint(uint256) (runs: 1024, : 70875, ~: 71342)
[PASS] testFuzz_Mint_TotalSupplyOverflow(uint256,uint256) (runs: 1024, : 66437, ~: 66451)
[PASS] testFuzz_Transfer(address,uint256) (runs: 1024, : 75807, ~: 76235)
[PASS] testFuzz_TransferFrom(address,uint256) (runs: 1024, : 94660, ~: 95234)
[PASS] testFuzz_TransferFrom_FailsIfInsufficientAllowance(address,uint256) (runs: 1024, : 93891, ~: 94144)
[PASS] testFuzz_TransferFrom_FailsIfInsufficientBalance(address,uint256) (runs: 1024, : 93891, ~: 94398)
[PASS] testFuzz_Transfer_FailsIfInsufficientBalance(address,uint256) (runs: 1024, : 65943, ~: 65931)
[PASS] testFuzz_UpgradeTo_FailsIfNotOwner(address) (runs: 1024, : 1485008, ~: 1485008)
[PASS] test_Approve() (gas: 43041)
[PASS] test_Burn() (gas: 73376)
[PASS] test_Burn_FailIfInsufficientBalance() (gas: 26901)
[PASS] test_Burn_FailIfNotFactory() (gas: 15454)
[PASS] test_Initialize_FailsIfZeroAddresses() (gas: 1587479)
[PASS] test_MetaData() (gas: 14056)
[PASS] test_Mint() (gas: 71294)
[PASS] test_Mint_FailIfNotFactory() (gas: 15466)
[PASS] test_Permit() (gas: 95502)
[PASS] test_Permit_BadDeadlineReverts() (gas: 46272)
[PASS] test_Permit_BadNonceReverts() (gas: 46639)
[PASS] test_Permit_PastDeadlineReverts() (gas: 40838)
[PASS] test_Permit_ReplayReverts() (gas: 97557)
[PASS] test_Transfer() (gas: 78204)
[PASS] test_TransferFrom() (gas: 96937)
[PASS] test_TransferFrom_FailsIfInsufficientAllowance() (gas: 96110)
[PASS] test_TransferFrom_FailsIfInsufficientBalance() (gas: 96389)
[PASS] test_Transfer_FailsIfInsufficientBalance() (gas: 67862)
[PASS] test_UpgradeTo() (gas: 1221872)
[PASS] test_UpgradeToFaultyHoney() (gas: 1187044)
[PASS] test_UpgradeToMockHoneyGovernanceCannotUpgradeWhenOwnerOfProxyChanges() (gas: 2267814)
[PASS] test_UpgradeTo_FailIfNotOwner() (gas: 1484918)
[PASS] test__codesize() (gas: 85471)
Suite result: ok. 34 passed; 0 failed; 0 skipped; finished in 559.45ms (557.70ms CPU time)

Ran 24 tests for test/pol/BlockRewardController.t.sol:BlockRewardControllerTest
[PASS] testFuzz_ComputeRewards(uint256,uint256,uint256,int256) (runs: 1024, : 18508, ~: 18612)
[PASS] testFuzz_ProcessRewards(uint256,uint256,uint256,uint256,uint256,uint256) (runs: 1024, : 772662, ~: 773322)
[PASS] testFuzz_SetBoostMultiplier(uint256) (runs: 1024, : 44610, ~: 44542)
[PASS] testFuzz_SetMinBoostedRewardRate(uint256) (runs: 1024, : 44198, ~: 44335)
[PASS] testFuzz_SetRewardConvexity(uint256) (runs: 1024, : 44639, ~: 44543)
[PASS] testFuzz_SetRewardRate(uint256) (runs: 1024, : 44022, ~: 44314)
[PASS] test_ComputeRewards() (gas: 26212)
[PASS] test_FailIfInitializeAgain() (gas: 24472)
[PASS] test_FailIfNotDistributor() (gas: 22475)
[PASS] test_FailIfNotOwner() (gas: 1765641)
[PASS] test_OwnerIsGovernance() (gas: 17516)
```

```
[PASS] test_ProcessRewards() (gas: 339200)
[PASS] test_ProcessRewardsMax() (gas: 551292)
[PASS] test_ProcessRewardsMin() (gas: 340041)
[PASS] test_ProcessRewardsWithCommission() (gas: 382544)
[PASS] test_ProcessZeroRewards() (gas: 70303)
[PASS] test_SetBaseRate() (gas: 44254)
[PASS] test_SetBoostMultiplier() (gas: 44493)
[PASS] test_SetDistributor() (gas: 26286)
[PASS] test_SetDistributor_FailIfZeroAddress() (gas: 17907)
[PASS] test_SetMinBoostedRewardRate() (gas: 44285)
[PASS] test_SetRewardConvexity() (gas: 44515)
[PASS] test_SetRewardRate() (gas: 44286)
[PASS] test_UpgradeTo() (gas: 1750629)
Suite result: ok. 24 passed; 0 failed; 0 skipped; finished in 579.42ms (578.54ms CPU time)

Ran 16 tests for test/pol/Distributor.t.sol:DistributorTest
[PASS] testFuzz_DistributeDoesNotLeaveDust(uint256) (runs: 1024, : 1877086, ~: 1877282)
[PASS] testFuzz_DistributeDoesNotLeaveDust(uint256,uint256,uint256,uint256,uint256) (runs: 1024, : 1936382, ~: 1939387)
[PASS] test_Distribute() (gas: 501993)
[PASS] test_DistributeAndActivateQueuedRewardAllocation() (gas: 590100)
[PASS] test_DistributeForNonReentrant() (gas: 3469148)
[PASS] test_DistributeMulticall() (gas: 658437)
[PASS] test_FailIfInitializeAgain() (gas: 24563)
[PASS] test_FailIfNotManager() (gas: 32216)
[PASS] test_FailIfNotOwner() (gas: 2128386)
[PASS] test_IsTimestampActionable_OutOfBuffer() (gas: 19266)
[PASS] test_IsTimestampActionable_Processing() (gas: 488499)
[PASS] test_OwnerIsGovernance() (gas: 16975)
[PASS] test_ProcessTimestamp_OutOfBuffer() (gas: 173923)
[PASS] test_ProcessTimestamp_Processing() (gas: 594389)
[PASS] test_UpgradeTo() (gas: 2109166)
[PASS] test_ZeroRewards() (gas: 296549)
Suite result: ok. 16 passed; 0 failed; 0 skipped; finished in 1.04s (1.03s CPU time)

Ran 2 tests for test/pol/e2e/POLGasSimulationMax.t.sol:POLGasSimulationMax
[PASS] testGasPOLDistribution() (gas: 1381287)
[PASS] testGasPOLDistributionCatchUp() (gas: 3099130)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 16.15ms (5.48ms CPU time)

Ran 24 tests for test/pol/BeaconDeposit.t.sol:BeaconDepositTest
[PASS] testFuzz_AcceptOperatorChange_FailsIfNotEnoughTime(uint256) (runs: 1024, : 167965, ~: 168272)
[PASS] testFuzz_DepositCount(uint256) (runs: 1024, : 1535047, ~: 1520903)
[PASS] testFuzz_DepositNotDivisibleByGwei(uint256) (runs: 1024, : 58591, ~: 58656)
[PASS] testFuzz_DepositWrongCredentials(bytes) (runs: 1024, : 21993, ~: 21993)
[PASS] testFuzz_DepositWrongMinAmount(uint256) (runs: 1024, : 90346, ~: 90382)
[PASS] testFuzz_DepositsWrongPubKey(bytes) (runs: 1024, : 26299, ~: 26292)
[PASS] testFuzz_RequestOperatorChange(address) (runs: 1024, : 161918, ~: 161918)
[PASS] test_AcceptOperatorChange() (gas: 158947)
[PASS] test_AcceptOperatorChange_FailsIfNotEnoughTime() (gas: 167562)
[PASS] test_AcceptOperatorChange_FailsIfNotNewOperator() (gas: 166833)
[PASS] test_AcceptOperatorChange_FailsIfNotQueued() (gas: 132094)
[PASS] test_CancelOperatorChange() (gas: 156323)
[PASS] test_CancelOperatorChange_FailsIfNotCurrentOperator() (gas: 166836)
[PASS] test_Deposit() (gas: 125185)
[PASS] test_DepositNotDivisibleByGwei() (gas: 94571)
[PASS] test_DepositWrongAmount() (gas: 90312)
[PASS] test_DepositWrongCredentials() (gas: 21694)
[PASS] test_DepositWrongPubKey() (gas: 26026)
[PASS] test_DepositZeroOperator() (gas: 32498)
[PASS] test_Deposit_FailsIfOperatorAlreadySet() (gas: 137010)
[PASS] test_RequestOperatorChange() (gas: 163893)
[PASS] test_RequestOperatorChange_FailsIfNotCurrentOperator() (gas: 130378)
[PASS] test_RequestOperatorChange_FailsIfZeroAddress() (gas: 128253)
[PASS] test_SupportsInterface() (gas: 10926)
Suite result: ok. 24 passed; 0 failed; 0 skipped; finished in 1.45s (1.63s CPU time)

Ran 3 tests for test/base/SSZ.t.sol:SSZTest
[PASS] test_ValidatorPubkeyRoot() (gas: 3495)
[PASS] test_addressHashTreeRoot() (gas: 3731)
[PASS] test_uint64HashTreeRoot() (gas: 4955)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 93.17µs (31.21µs CPU time)
```

```
Ran 1 test for test/pol/e2e/POLGasSim.t.sol:POLGasSimulationSimple
[PASS] testGasPOLDistribution() (gas: 401791)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 3.66ms (262.33µs CPU time)

Ran 1 test for test/pol/e2e/POLGasSimulationAdvance.t.sol:POLGasSimulationAdvance
[PASS] testGasPOLDistribution() (gas: 941331)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 32.61ms (5.70ms CPU time)

Ran 16 tests for test/pol/RewardVaultFactory.t.sol:RewardVaultFactoryTest
[PASS] testFuzz_CreateRewardVault(address) (runs: 1024, : 303169, ~: 303169)
[PASS] testFuzz_SetRewardVaultImplementation_FailIsNotOwner(address) (runs: 1024, : 539976, ~: 539976)
[PASS] testFuzz_UpgradeToFailsIfNotOwner(address) (runs: 1024, : 1305829, ~: 1305829)
[PASS] test_CreateMultipleVaults() (gas: 1437033)
[PASS] test_CreateRewardVault() (gas: 304770)
[PASS] test_CreateRewardVault_CreateNewVaultWithoutUpdatingMapping() (gas: 1002076)
[PASS] test_CreateRewardVault_DoesNotFailIfVaultImplChange() (gas: 1005620)
[PASS] test_CreateRewardVault_FailIfVaultImplDoesNotChange() (gas: 1024187348)
[PASS] test_CreateRewardVault_RevertsIfStakingTokenIsNotAContract() (gas: 24421)
[PASS] test_GetVaultsLength() (gas: 307898)
[PASS] test_InitialState() (gas: 35967)
[PASS] test_SetRewardVaultImplementation() (gas: 549237)
[PASS] test_SetRewardVaultImplementation_FailIfNewVaultIsNotAContract() (gas: 22079)
[PASS] test_UpgradeToAndCall() (gas: 1311129)
[PASS] test_UpgradeToFailsIfImplIsNotUUPS() (gas: 19340)
[PASS] test_UpgradeToFailsIfNotOwner() (gas: 1305714)
Suite result: ok. 16 passed; 0 failed; 0 skipped; finished in 193.77ms (190.63ms CPU time)

Ran 25 tests for test/pol/FeeCollector.t.sol:FeeCollectorTest
[PASS] testFuzz_Donate(uint256) (runs: 1024, : 203034, ~: 203176)
[PASS] testFuzz_Donate_FailsIfAmountLessThanPayoutAmount(uint256) (runs: 1024, : 18153, ~: 17816)
[PASS] testFuzz_Donate_FailsIfNotApproved(uint256) (runs: 1024, : 52973, ~: 52916)
[PASS] testFuzz_Donate_FailsIfPaused(uint256) (runs: 1024, : 154927, ~: 154684)
[PASS] testFuzz_PayoutAmountChangeAfterClaim(uint256) (runs: 1024, : 177694, ~: 177706)
[PASS] testFuzz_PayoutAmountDoesntChangeIfClaimFails(uint256) (runs: 1024, : 103258, ~: 103256)
[PASS] testFuzz_QueuePayoutAmountChange(uint256) (runs: 1024, : 49479, ~: 49476)
[PASS] test_ClaimFees() (gas: 161699)
[PASS] test_ClaimFees_FailsIfNotApproved() (gas: 64594)
[PASS] test_ClaimFees_FailsIfPaused() (gas: 111321)
[PASS] test_Donate() (gas: 202713)
[PASS] test_Donate_FailsIfAmountLessThanPayoutAmount() (gas: 17766)
[PASS] test_Donate_FailsIfNotApproved() (gas: 52610)
[PASS] test_GovernanceIsOwner() (gas: 16999)
[PASS] test_Initialize_FailsIfGovernanceAddrIsZero() (gas: 1648663)
[PASS] test_Initialize_FailsIfPayoutAmountIsZero() (gas: 1650809)
[PASS] test_Initialize_FailsIfPayoutTokenAddrIsZero() (gas: 1648690)
[PASS] test_Initialize_FailsIfRewardReceiverAddrIsZero() (gas: 1648650)
[PASS] test_Pause() (gas: 42572)
[PASS] test_Pause_FailIfNotVaultManager() (gas: 17940)
[PASS] test_QueuePayoutAmountChange() (gas: 49405)
[PASS] test_QueuePayoutAmountChange_FailsIfNotOwner() (gas: 18032)
[PASS] test_QueuePayoutAmountChange_FailsIfZero() (gas: 18022)
[PASS] test_Unpause() (gas: 32087)
[PASS] test_Unpause_FailIfNotVaultManager() (gas: 48794)
Suite result: ok. 25 passed; 0 failed; 0 skipped; finished in 632.84ms (629.21ms CPU time)

Ran 14 tests for test/base/Utils.t.sol:UtilsTest
[PASS] testFuzz_Allowance(address,address,uint256) (runs: 1024, : 36633, ~: 36828)
[PASS] testFuzz_SafeIncreaseAllowance(address,uint256) (runs: 1024, : 40354, ~: 40332)
[PASS] testFuzz_SafeIncreaseAllowance_Overflow(uint256,uint256) (runs: 1024, : 41279, ~: 41352)
[PASS] testFuzz_TrySafeTransferDoesNotExeedGasLimit(address) (runs: 1024, : 375299, ~: 375299)
[PASS] testFuzz_TrySafeTransferERC20(address,uint256) (runs: 1024, : 64941, ~: 65486)
[PASS] testFuzz_TrySafeTransferExeedGasLimit(address) (runs: 1024, : 2007880, ~: 2007880)
[PASS] testFuzz_TrySafeTransferMissingReturnToken(address,uint256) (runs: 1024, : 41940, ~: 42182)
[PASS] testFuzz_TrySafeTransferReturnFalseToken(address,uint256) (runs: 1024, : 17030, ~: 17030)
[PASS] testFuzz_TrySafeTransferReturnTwoToken(address,uint256) (runs: 1024, : 16961, ~: 16961)
[PASS] testFuzz_TrySafeTransferRevertingToken(address,uint256) (runs: 1024, : 16951, ~: 16951)
[PASS] test_Allowance() (gas: 38737)
[PASS] test_Allowance_WhenTokenNotExists() (gas: 5735)
[PASS] test_SafeIncreaseAllowance() (gas: 42321)
[PASS] test_SafeIncreaseAllowance_IfTokenNotExists() (gas: 10476)
Suite result: ok. 14 passed; 0 failed; 0 skipped; finished in 2.76s (2.90s CPU time)
```

```
Ran 96 tests for test/pol/BGT.t.sol:BGTTest
[PASS] testFuzz_ActivateBoost(address,address,uint256) (runs: 1024, : 349151, ~: 348918)
[PASS] testFuzz_ActivateBoost_ReturnsFalseIfNotEnoughTimePassed(uint256) (runs: 1024, : 218002, ~: 218128)
[PASS] testFuzz_ActivateCommissionChange(uint256) (runs: 1024, : 88234, ~: 88568)
[PASS] testFuzz_ActivateCommissionChange_FailsIfHistoryBufferNotPassed(uint256) (runs: 1024, : 78525, ~: 78650)
[PASS] testFuzz_Approve(address,address,uint256) (runs: 1024, : 66746, ~: 67019)
[PASS] testFuzz_CancelBoost(address,uint256,uint256) (runs: 1024, : 221342, ~: 222776)
[PASS] testFuzz_CancelBoost_FailsIfCancelledMoreThanQueuedBoost(uint256,uint256) (runs: 1024, : 201530, ~: 201901)
[PASS] testFuzz_CancelCommissionChange(uint256) (runs: 1024, : 52160, ~: 52208)
[PASS] testFuzz_CancelCommissionChange_FailsIfNotOperator(address) (runs: 1024, : 63767, ~: 63767)
[PASS] testFuzz_CancelDropBoost(address,uint256,uint256) (runs: 1024, : 408879, ~: 408942)
[PASS] testFuzz_CancelDropBoost_FailsIfCancelledMoreThanQueuedDropBoost(uint256,uint256) (runs: 1024, : 399648, ~:
↪  399574)
[PASS] testFuzz_DropBoost(address,uint256,uint256) (runs: 1024, : 408778, ~: 410918)
[PASS] testFuzz_DropBoost_ReturnsFalseIfNotEnoughTimePassed(uint256) (runs: 1024, : 396937, ~: 397115)
[PASS] testFuzz_Mint(address,uint256) (runs: 1024, : 111950, ~: 112759)
[PASS] testFuzz_MultiCall_CancelBoostInBatch(address,uint256,uint256) (runs: 1024, : 271938, ~: 272056)
[PASS] testFuzz_MultiCall_QueueBoostInBatch(address,uint256,uint256) (runs: 1024, : 252370, ~: 252483)
[PASS] testFuzz_MultiCall_QueueDropBoostInBatch(uint256,uint256,uint256) (runs: 1024, : 389498, ~: 389785)
[PASS] testFuzz_QueueBoost(address,uint256) (runs: 1024, : 194152, ~: 195132)
[PASS] testFuzz_QueueCommissionChange(uint256) (runs: 1024, : 59233, ~: 59290)
[PASS] testFuzz_QueueCommissionChange_FailsIfNotOperator(address) (runs: 1024, : 26941, ~: 26941)
[PASS] testFuzz_QueueCommissionChange_FailsIfOverTenPercent(uint256) (runs: 1024, : 26571, ~: 26448)
[PASS] testFuzz_QueueDropBoost(address,uint256,uint256) (runs: 1024, : 393320, ~: 393394)
[PASS] testFuzz_QueueDropBoost_FailsIfDroppedMoreThanBoost(uint256,uint256) (runs: 1024, : 357630, ~: 357601)
[PASS] testFuzz_Redeem(uint256) (runs: 1024, : 168630, ~: 168988)
[PASS] testFuzz_SetActivateBoostDelay(uint256) (runs: 1024, : 21035, ~: 21203)
[PASS] testFuzz_SetActivateCommissionChangeDelay(uint256) (runs: 1024, : 20973, ~: 21169)
[PASS] testFuzz_SetDropBoostDelay(uint256) (runs: 1024, : 20993, ~: 21208)
[PASS] testFuzz_SetMinter(address) (runs: 1024, : 25329, ~: 25329)
[PASS] testFuzz_Transfer(address,address,uint256) (runs: 1024, : 155313, ~: 155891)
[PASS] testFuzz_TransferFrom(address,address,uint256,uint256) (runs: 1024, : 156866, ~: 158303)
[PASS] testFuzz_WhitelistSender(address,bool) (runs: 1024, : 29834, ~: 20137)
[PASS] testMetadata() (gas: 12649)
[PASS] test_ActivateBoost() (gas: 350794)
[PASS] test_ActivateBoost_ReturnsFalseIfNotEnoughTimePassed() (gas: 217952)
[PASS] test_ActivateBoost_ReturnsFalseIfNotQueued() (gas: 18527)
[PASS] test_ActivateCommissionChange() (gas: 87857)
[PASS] test_ActivateCommissionChange_FailsIfCancelled() (gas: 65413)
[PASS] test_ActivateCommissionChange_FailsIfHistoryBufferNotPassed() (gas: 78383)
[PASS] test_ActivateCommissionChange_FailsIfNotQueued() (gas: 17814)
[PASS] test_Approve() (gas: 41273)
[PASS] test_CancelBoost() (gas: 204199)
[PASS] test_CancelBoost_FailsIfCancelledMoreThanQueuedBoost() (gas: 201173)
[PASS] test_CancelCommissionChange() (gas: 51724)
[PASS] test_CancelCommissionChange_FailsIfNotOperator() (gas: 63635)
[PASS] test_CancelDropBoost() (gas: 410223)
[PASS] test_CancelDropBoost_FailsIfCancelledMoreThanQueuedDropBoost() (gas: 398778)
[PASS] test_DropBoost() (gas: 410087)
[PASS] test_DropBoost_FailsIfQueueAmountIsZero() (gas: 401831)
[PASS] test_DropBoost_ReturnsFalseIfNotEnoughTimePassed() (gas: 396361)
[PASS] test_FailIfMinterIsZero() (gas: 13096)
[PASS] test_FailIfNotApprovedSender() (gas: 21244)
[PASS] test_FailIfNotOwner() (gas: 18499)
[PASS] test_FailIndirectTransferFromInsufficientAllowance() (gas: 145907)
[PASS] test_FailMintIfNotMinter() (gas: 12699)
[PASS] test_FailMintOverMaxUint() (gas: 111831)
[PASS] test_FailTransferFromInsufficientAllowance() (gas: 144523)
[PASS] test_FailTransferFromInsufficientBalance() (gas: 43648)
[PASS] test_FailTransferInsufficientBalance() (gas: 17928)
[PASS] test_Mint() (gas: 114784)
[PASS] test_Mint_FailsIfInvariantCheckFails() (gas: 112027)
[PASS] test_MinterIsBlockRewardController() (gas: 12678)
[PASS] test_MultiCall_ActivateBoostInBatch() (gas: 450427)
[PASS] test_MultiCall_ActivateBoostInBatch_DoesNotFailIfNotEnoughTimePassed() (gas: 381233)
[PASS] test_MultiCall_CancelBoostInBatch() (gas: 273251)
[PASS] test_MultiCall_QueueBoostInBatch() (gas: 253757)
[PASS] test_MultiCall_QueueBoostInBatch_FailsIfNotEnoughUnboostedBalance() (gas: 25259)
[PASS] test_MultiCall_QueueDropBoostInBatch() (gas: 388308)
[PASS] test_MultiCall_SetPramsInBatch() (gas: 33211)
[PASS] test_MultiCall_SetPramsInBatch_FailsIfNotOwner() (gas: 15029)
[PASS] test_OwnerIsGovernance() (gas: 12701)
[PASS] test_QueueBoost() (gas: 197132)
```

```
[PASS] test_QueueCommissionChange() (gas: 58651)
[PASS] test_QueueCommissionChange_FailsIfNotOperator() (gas: 26829)
[PASS] test_QueueCommissionChange_FailsIfOverTenPercent() (gas: 26426)
[PASS] test_QueueDropBoost() (gas: 394672)
[PASS] test_QueueDropBoost_FailsIfDroppedMoreThanBoost() (gas: 356872)
[PASS] test_Redeem() (gas: 168514)
[PASS] test_Redeem_FailsIfInvariantCheckFails() (gas: 160950)
[PASS] test_Redeem_FailsIfNotEnoughUnboostedBalance() (gas: 199593)
[PASS] test_Redeem_FailsWithETHTransferFail() (gas: 100770)
[PASS] test_SetActivateBoostDelay() (gas: 20445)
[PASS] test_SetActivateBoostDelay_FailsIfGreaterThanHistoryBufferLength() (gas: 13587)
[PASS] test_SetActivateBoostDelay_FailsIfNotOwner() (gas: 11030)
[PASS] test_SetActivateBoostDelay_FailsIfZero() (gas: 13433)
[PASS] test_SetActivateCommissionChangeDelay() (gas: 20479)
[PASS] test_SetActivateCommissionChangeDelay_FailsIfGreaterThanHistoryBufferLength() (gas: 13520)
[PASS] test_SetActivateCommissionChangeDelay_FailsIfNotOwner() (gas: 11053)
[PASS] test_SetActivateCommissionChangeDelay_FailsIfZero() (gas: 13433)
[PASS] test_SetDropBoostDelay() (gas: 20543)
[PASS] test_SetDropBoostDelay_FailsIfGreaterThanHistoryBufferLength() (gas: 13540)
[PASS] test_SetDropBoostDelay_FailsIfNotOwner() (gas: 11051)
[PASS] test_SetDropBoostDelay_FailsIfZero() (gas: 13402)
[PASS] test_SetMinter() (gas: 24830)
[PASS] test_Transfer() (gas: 131960)
[PASS] test_TransferFrom() (gas: 137462)
[PASS] test_WhitelistSender() (gas: 39759)
Suite result: ok. 96 passed; 0 failed; 0 skipped; finished in 4.59s (5.04s CPU time)

Ran 119 tests for test/pol/RewardVault.t.sol:RewardVaultTest
[PASS] testFuzz_AddIncentive_FailsIfAmountLessThanMinRate(uint256) (runs: 1024, : 298103, ~: 297788)
[PASS] testFuzz_AddIncentive_FailsIfRateMoreThanMaxIncentiveRate(uint256) (runs: 1024, : 297160, ~: 297433)
[PASS] testFuzz_AddIncentive_IncentiveRateNotChanged(uint256) (runs: 1024, : 566563, ~: 566580)
[PASS] testFuzz_AddIncentive_IncreaseIncentiveRate(uint256) (runs: 1024, : 567066, ~: 567162)
[PASS] testFuzz_AddIncentive_UpdatesIncentiveRate(uint256,uint256) (runs: 1024, : 556303, ~: 556584)
[PASS] testFuzz_ChangeInFactoryOwner(address) (runs: 1024, : 66443, ~: 66443)
[PASS] testFuzz_DelegateStake(address,address,uint256) (runs: 1024, : 216466, ~: 216269)
[PASS] testFuzz_DelegateWithdraw(address,address,uint256,uint256) (runs: 1024, : 258327, ~: 258570)
[PASS] testFuzz_DelegateWithdrawFailsIfNotEnoughStakedByDelegate(uint256,uint256,uint256) (runs: 1024, : 405957, ~:
↪  405856)
[PASS] testFuzz_DistributeDoesNotLeaveDust(uint256) (runs: 1024, : 1877128, ~: 1877323)
[PASS] testFuzz_DistributeDoesNotLeaveDust(uint256,uint256,uint256,uint256,uint256) (runs: 1024, : 1935380, ~: 1939847)
[PASS] testFuzz_Exit(uint256,uint256) (runs: 1024, : 979811, ~: 948016)
[PASS] testFuzz_GetRewardToRecipient(address) (runs: 1024, : 886530, ~: 886530)
[PASS] testFuzz_NotifyRewardsFailsIfRewardInsolvent(uint256,uint256) (runs: 1024, : 220486, ~: 220548)
[PASS] testFuzz_PartialWithdraw(address,uint256,uint256) (runs: 1024, : 326928, ~: 326885)
[PASS] testFuzz_ProcessIncentives(uint256) (runs: 1024, : 803125, ~: 765773)
[PASS] testFuzz_RemoveIncentiveToken(address) (runs: 1024, : 134378, ~: 134365)
[PASS] testFuzz_RemoveIncentiveToken_Multiple(uint8,uint256) (runs: 1024, : 4053566, ~: 3634983)
[PASS] testFuzz_SetRewardDuration(uint256) (runs: 1024, : 38136, ~: 38132)
[PASS] testFuzz_SetRewardsDurationDuringCycleEarned(uint256,uint256) (runs: 1024, : 705303, ~: 705353)
[PASS] testFuzz_Stake(address,uint256) (runs: 1024, : 295255, ~: 295388)
[PASS] testFuzz_UpdateIncentiveManager(address,address) (runs: 1024, : 165350, ~: 165350)
[PASS] testFuzz_WhitelistIncentiveToken(address,address) (runs: 1024, : 154022, ~: 154022)
[PASS] testFuzz_WhitelistIncentiveToken_FailsIfMinIncentiveRateMoreThanMax(uint256) (runs: 1024, : 35660, ~: 35911)
[PASS] testFuzz_Withdraw(address,uint256) (runs: 1024, : 326922, ~: 326961)
[PASS] testFuzz_Withdraw_FailsIfInsufficientSelfStake(uint256,uint256,uint256) (runs: 1024, : 409169, ~: 409117)
[PASS] test_AddIncentive_FailIfNotManager() (gas: 294201)
[PASS] test_AddIncentive_FailsIfAmountLessThanMinIncentiveRate() (gas: 294344)
[PASS] test_AddIncentive_FailsIfNotWhitelisted() (gas: 37706)
[PASS] test_AddIncentive_FailsIfNotWhitelistedToken() (gas: 37709)
[PASS] test_AddIncentive_FailsIfRateIsMoreThanMaxIncentiveRate() (gas: 297384)
Logs:
  Bound result 1157920892373161954235709850086879078532690846656405640394575840079131296399 35

[PASS] test_ChangeInFactoryOwner() (gas: 67734)
[PASS] test_DelegateStake() (gas: 220059)
[PASS] test_DelegateStakeFailsIfPused() (gas: 77735)
[PASS] test_DelegateStakeWithSelfStake() (gas: 481465)
[PASS] test_DelegateWithdraw() (gas: 233722)
[PASS] test_DelegateWithdrawFailsIfNotDelegate() (gas: 33388)
[PASS] test_DelegateWithdrawFailsIfNotEnoughStakedByDelegate() (gas: 215757)
[PASS] test_Distribute() (gas: 502084)
[PASS] test_DistributeAndActivateQueuedRewardAllocation() (gas: 590208)
[PASS] test_DistributeForNonReentrant() (gas: 3469301)
[PASS] test_DistributeMulticall() (gas: 658493)
```

```
[PASS] test_DoNotUpdateIncentiveRateWhenAmountIsInsufficient() (gas: 563180)
[PASS] test_Exit() (gas: 1019253)
Logs:
  Bound result 10000000000000000000
  Bound result 10000000000000000000

[PASS] test_ExitWithZeroBalance() (gas: 42231)
[PASS] test_FactoryOwner() (gas: 15351)
[PASS] test_FailIfInitializeAgain() (gas: 22289)
[PASS] test_FailIfNotManager() (gas: 32261)
[PASS] test_FailIfNotOwner() (gas: 92707)
[PASS] test_FailNotifyRewardNoSufficientAllowance(int256) (runs: 1024, : 829979, ~: 832957)
[PASS] test_GetRewardNotOperatorOrUser() (gas: 40188)
[PASS] test_GetRewardWithDelegateStake() (gas: 844659)
[PASS] test_GetReward_AfterDuration(uint256,uint256) (runs: 1024, : 663466, ~: 663334)
[PASS] test_GetReward_BeforeDuration(uint256,uint256) (runs: 1024, : 663330, ~: 663343)
[PASS] test_GetReward_Notified_Twice(uint256,uint256) (runs: 1024, : 755053, ~: 754910)
[PASS] test_GetRewards() (gas: 825595)
[PASS] test_GetWhitelistedTokens() (gas: 352356)
[PASS] test_IncreaseIncentiveRate() (gas: 563219)
[PASS] test_InitialState() (gas: 33029)
[PASS] test_IsTimestampActionable_OutOfBuffer() (gas: 19355)
[PASS] test_IsTimestampActionable_Processing() (gas: 488568)
[PASS] test_NotifyRewardsDoesNotSetRewardRate() (gas: 418601)
[PASS] test_NotifyRewardsFailsIfRewardInsolvent() (gas: 220286)
[PASS] test_NotifyRewardsSetRewardRate() (gas: 741121)
[PASS] test_OperatorWorks() (gas: 913916)
[PASS] test_OwnerIsGovernance() (gas: 24206)
[PASS] test_PanprogL1() (gas: 593652)
[PASS] test_Pause() (gas: 51659)
[PASS] test_Pause_FailIfNotVaultManager() (gas: 25152)
[PASS] test_ProcessIncentives() (gas: 685852)
[PASS] test_ProcessIncentives_NotFailWithMaliciusIncentive() (gas: 991122)
[PASS] test_ProcessTimestamp_OutOfBuffer() (gas: 173901)
[PASS] test_ProcessTimestamp_Processing() (gas: 594368)
[PASS] test_RecoverERC20() (gas: 106199)
[PASS] test_RecoverERC20_FailIfNotOwner() (gas: 27323)
[PASS] test_RecoverERC20_FailIfStakingToken() (gas: 31596)
[PASS] test_RecoverERC20_FailsIfIncentiveToken() (gas: 165070)
[PASS] test_RemoveIncentiveToken() (gas: 239080)
[PASS] test_RemoveIncentiveToken_FailsIfNotVaultManager() (gas: 276005)
[PASS] test_RemoveIncentiveToken_FailsIfNotWhitelisted() (gas: 31673)
[PASS] test_RewardRateRemains_Zero_UntilFirstStake() (gas: 258500)
[PASS] test_RewardRateWithMultipleDistributeRewards() (gas: 1126745)
[PASS] test_SelfStake() (gas: 304156)
[PASS] test_SetDistributor() (gas: 37606)
[PASS] test_SetDistributor_FailIfNotOwner() (gas: 25218)
[PASS] test_SetDistributor_FailWithZeroAddress() (gas: 27367)
[PASS] test_SetMaxIncentiveTokensCount() (gas: 38128)
[PASS] test_SetMaxIncentiveTokensCount_FailsIfLessThanCurrentWhitelistedCount() (gas: 274165)
[PASS] test_SetMaxIncentiveTokensCount_FailsIfNotOwner() (gas: 25190)
[PASS] test_SetOperator() (gas: 42761)
[PASS] test_SetRewardDuration() (gas: 38106)
[PASS] test_SetRewardDurationDuringCycleEarned() (gas: 704896)
[PASS] test_SetRewardDuration_FailIfNotOwner() (gas: 25221)
[PASS] test_SetRewardDuration_FailsIfZero() (gas: 27354)
[PASS] test_SetRewardRateAfterFirstStake() (gas: 579442)
[PASS] test_SetRewardsDurationDuringCycle() (gas: 713260)
[PASS] test_SetRewardsDurationDuringCycleMultipleUsers() (gas: 934521)
[PASS] test_Stake() (gas: 295348)
[PASS] test_StakeFailsIfPaused() (gas: 75568)
[PASS] test_Stake_FailsIfOverflow(address) (runs: 1024, : 302878, ~: 303078)
[PASS] test_Stake_FailsIfZeroAmount() (gas: 35768)
[PASS] test_TotalSupply() (gas: 509412)
[PASS] test_Unpause() (gas: 41184)
[PASS] test_Unpause_FailIfNotVaultManager() (gas: 58026)
[PASS] test_UpdateIncentiveManager() (gas: 169418)
[PASS] test_UpdateIncentiveManager_FailsIfNotFactoryOwner() (gas: 27481)
[PASS] test_UpdateIncentiveManager_FailsIfTokenNotWhitelisted() (gas: 52048)
[PASS] test_UpdateIncentiveManager_FailsIfZeroAddress() (gas: 274471)
[PASS] test_UpgradeTo() (gas: 2109233)
[PASS] test_WhitelistIncentiveToken() (gas: 269421)
[PASS] test_WhitelistIncentiveToken_FailsIfAlreadyWhitelisted() (gas: 274936)
[PASS] test_WhitelistIncentiveToken_FailsIfCountEqualToMax() (gas: 847961)
```

```
[PASS] test_WhitelistIncentiveToken_FailsIfMinIncentiveRateIsZero() (gas: 31813)
[PASS] test_WhitelistIncentiveToken_FailsIfMinIncentiveRateMoreThanMax() (gas: 35180)
Logs:
  Bound result 100000000000000000000000000000000000

[PASS] test_WhitelistIncentiveToken_FailsIfNotOwner() (gas: 29636)
[PASS] test_WhitelistIncentiveToken_FailsIfZeroAddress() (gas: 49732)
[PASS] test_Withdraw_FailsIfInsufficientSelfStake() (gas: 408969)
Logs:
  Bound result 10000000000000000000
  Bound result 10000000000000000000
  Bound result 10000000000000000000

[PASS] test_Withdraw_FailsIfZeroAmount() (gas: 38000)
[PASS] test_ZeroRewards() (gas: 296594)
Suite result: ok. 119 passed; 0 failed; 0 skipped; finished in 3.84s (11.56s CPU time)

Ran 44 tests for test/pol/BGTStaker.t.sol:BGTStakerTest
[PASS] testFuzz_NotifyRewardsFailsIfNotFeeCollector(address) (runs: 1024, : 18379, ~: 18379)
[PASS] testFuzz_NotifyRewardsFailsIfRewardInsolvent(uint256,uint256) (runs: 1024, : 214332, ~: 214390)
[PASS] testFuzz_PartialWithdraw(address,uint256,uint256) (runs: 1024, : 99402, ~: 99695)
[PASS] testFuzz_RecoverERC20(uint256) (runs: 1024, : 1036556, ~: 1036776)
[PASS] testFuzz_RecoverERC20FailsIfNotOwner(address) (runs: 1024, : 18794, ~: 18794)
[PASS] testFuzz_SetRewardDuration(uint256) (runs: 1024, : 28829, ~: 28828)
[PASS] testFuzz_SetRewardDurationFailsIfNotOwner(address) (runs: 1024, : 18685, ~: 18685)
[PASS] testFuzz_SetRewardsDurationDuringCycleEarned(uint256,uint256) (runs: 1024, : 431734, ~: 431785)
[PASS] testFuzz_Stake(address,uint256) (runs: 1024, : 85907, ~: 85948)
[PASS] testFuzz_StakeFailsIfNotBGT(address) (runs: 1024, : 18466, ~: 18466)
[PASS] testFuzz_Withdraw(address,uint256) (runs: 1024, : 75497, ~: 75534)
[PASS] testFuzz_WithdrawFailsIfInsufficientStake(address,uint256,uint256) (runs: 1024, : 88109, ~: 88319)
[PASS] testFuzz_WithdrawFailsIfNotBGT(address) (runs: 1024, : 18398, ~: 18398)
[PASS] testFuzz_getRewardWithLowTotalSupply(uint256) (runs: 1024, : 567042, ~: 567025)
[PASS] test_GetReward_AfterDuration(uint256,uint256) (runs: 1024, : 403842, ~: 403608)
[PASS] test_GetReward_BeforeDuration(uint256,uint256) (runs: 1024, : 403689, ~: 403699)
[PASS] test_GetReward_Notified_Twice(uint256,uint256) (runs: 1024, : 482814, ~: 482671)
[PASS] test_GetRewards() (gas: 480596)
[PASS] test_NotifyRewardsDoesNotSetRewardRate() (gas: 301999)
[PASS] test_NotifyRewardsFailsIfNotFeeCollector() (gas: 18245)
[PASS] test_NotifyRewardsFailsIfRewardInsolvent() (gas: 214163)
[PASS] test_NotifyRewardsSetRewardRate() (gas: 403588)
[PASS] test_OwnerIsGovernance() (gas: 17483)
[PASS] test_RecoverERC20() (gas: 1036143)
[PASS] test_RecoverERC20FailsIfNotOwner() (gas: 18725)
[PASS] test_RecoverERC20FailsIfRewardToken() (gas: 186132)
[PASS] test_RewardRateRemains_Zero_UntilFirstStake() (gas: 222968)
[PASS] test_RewardRateWithMultipleDistributeRewards() (gas: 693393)
[PASS] test_SetRewardDuration() (gas: 28778)
[PASS] test_SetRewardDurationDuringCycleEarned() (gas: 431284)
[PASS] test_SetRewardDurationFailsIfNotOwner() (gas: 18617)
[PASS] test_SetRewardDuration_FailsIfZero() (gas: 17871)
[PASS] test_SetRewardRateAfterFirstStake() (gas: 328704)
[PASS] test_SetRewardsDurationDuringCycle() (gas: 440369)
[PASS] test_SetRewardsDurationDuringCycleMultipleUsers() (gas: 477750)
[PASS] test_Stake() (gas: 85795)
[PASS] test_StakeFailsIfNotBGT() (gas: 18331)
[PASS] test_Stake_FailsIfOverflow(address) (runs: 1024, : 93249, ~: 93305)
[PASS] test_Stake_FailsIfZeroAmount() (gas: 17987)
[PASS] test_UpgradeFailsIfNewImplNotUUPS() (gas: 22652)
[PASS] test_UpgradeFailsIfNotOwner() (gas: 1597520)
[PASS] test_UpgradeTo() (gas: 1606013)
[PASS] test_WithdrawFailsIfNotBGT() (gas: 18286)
[PASS] test_Withdraw_FailsIfZeroAmount() (gas: 17985)
Suite result: ok. 44 passed; 0 failed; 0 skipped; finished in 5.18s (3.53s CPU time)

Ran 144 tests for test/honey/HoneyFactory.t.sol:HoneyFactoryTest
[PASS] testFuzz_BasketModeDisabledWhenAnAssetDepegOverOneDollar_WhenMint(uint256) (runs: 1024, : 69056, ~: 69054)
[PASS] testFuzz_BasketModeDisabledWhenAnAssetDepegUnderOneDollar_WhenMint(uint256) (runs: 1024, : 69091, ~: 69086)
[PASS] testFuzz_BasketModeEnabledWhenAllAssetsDepegOverOneDollar_WhenMint(uint256) (runs: 1024, : 121087, ~: 121084)
[PASS] testFuzz_BasketModeEnabledWhenAllAssetsDepegUnderOneDollar_WhenMint(uint256) (runs: 1024, : 116306, ~: 116301)
[PASS] testFuzz_BasketModeEnabledWhenAnAssetDepegOverOneDollar_WhenRedeem(uint256) (runs: 1024, : 114017, ~: 114015)
[PASS] testFuzz_BasketModeEnabledWhenAnAssetDepegUnderOneDollar_WhenRedeem(uint256) (runs: 1024, : 113767, ~: 113762)
[PASS] testFuzz_CollectedFeesWithDifferentFeeRate(uint256,uint256) (runs: 1024, : 383320, ~: 393672)
[PASS] testFuzz_InflationAttack(uint256) (runs: 1024, : 415024, ~: 415786)
[PASS] testFuzz_PreviewHoneyToRedeem(uint64) (runs: 1024, : 51803, ~: 51803)
```

```
[PASS] testFuzz_PreviewRequiredCollateral(uint128) (runs: 1024, : 131030, ~: 131030)
[PASS] testFuzz_PreviewRequiredCollateralReturnsAllZeroWhenBasketModeIsEnabledWithoutAnyDeposit(uint256) (runs: 1024, :
↪  138589, ~: 138589)
[PASS] testFuzz_PreviewRequiredCollateralWhenBasketModeIsEnabled(uint256,uint256,uint256,uint256) (runs: 1024, :
↪  987643, ~: 987651)
[PASS] testFuzz_liquidate_WhenBadCollateralDepegOverOneDollar(uint256,uint256,uint256,uint256,uint256) (runs: 1024, :
↪  757597, ~: 754672)
[PASS] testFuzz_liquidate_WhenBadCollateralDepegUnderOneDollar(uint256,uint256,uint256,uint256,uint256) (runs: 1024, :
↪  759412, ~: 754780)
[PASS] testFuzz_liquidate_failsWhenExceedsRelativeCap(uint256,uint256,uint256) (runs: 1024, : 926495, ~: 927337)
[PASS] testFuzz_mint(uint256) (runs: 1024, : 367029, ~: 368925)
[PASS] testFuzz_mintWithHigherDecimalAsset(uint256) (runs: 1024, : 583901, ~: 583939)
[PASS] testFuzz_mintWithLowerDecimalAsset(uint256) (runs: 1024, : 580860, ~: 583957)
[PASS] testFuzz_mint_WhenBasketModeIsEnabledAndAllAssetsAreDepeggedOrBadCollateral(uint256) (runs: 1024, : 1219670, ~:
↪  1212250)
[PASS] testFuzz_mint_WhenBasketModeIsEnabledBecauseOfAllAssetsAreDepegged(uint256) (runs: 1024, : 1165575, ~: 1171428)
[PASS] testFuzz_mint_failsIfExceedsGlobalCap(uint256) (runs: 1024, : 708591, ~: 708611)
[PASS] testFuzz_mint_failsIfExceedsRelativeCap(uint256,uint256) (runs: 1024, : 276072, ~: 276078)
[PASS] testFuzz_mint_failsWithInsufficientAllowance(uint256) (runs: 1024, : 95189, ~: 95751)
[PASS] testFuzz_mint_failsWithPausedFactory(uint128) (runs: 1024, : 74464, ~: 74892)
[PASS] testFuzz_mint_failsWithUnregisteredAsset(uint32) (runs: 1024, : 654071, ~: 656637)
[PASS] testFuzz_recapitalize(uint256,uint256) (runs: 1024, : 451460, ~: 451476)
[PASS] testFuzz_recapitalize_failsWhenExceedRelativeCap(uint256,uint256) (runs: 1024, : 692056, ~: 692127)
[PASS] testFuzz_recapitalize_failsWhenExceedsGlobalCap(uint256) (runs: 1024, : 719411, ~: 719430)
[PASS] testFuzz_recapitalize_failsWhenTargetBalanceIsNotSet(uint256) (runs: 1024, : 77458, ~: 77512)
[PASS] testFuzz_recapitalize_failsWhenUserProvideAmountLessThanTheMinimumAllowed(uint256) (runs: 1024, : 77171, ~:
↪  77395)
[PASS] testFuzz_redeem(uint256) (runs: 1024, : 495283, ~: 495654)
[PASS] testFuzz_redeemWithHigherDecimalAsset(uint256) (runs: 1024, : 684101, ~: 684023)
[PASS] testFuzz_redeemWithLowerDecimalAsset(uint256) (runs: 1024, : 684032, ~: 683950)
[PASS] testFuzz_redeem_WhenBasketModeIsEnabled(uint256,uint256,uint256,uint256,uint256) (runs: 1024, : 925001, ~:
↪  925022)
[PASS] testFuzz_redeem_WhenBasketModeIsEnabledAssetWithWeightZero(uint256) (runs: 1024, : 515954, ~: 516574)
[PASS] testFuzz_redeem_failWithPausedFactory(uint128) (runs: 1024, : 49304, ~: 49304)
[PASS] testFuzz_redeem_failsWithInsufficientHoneys(uint256) (runs: 1024, : 123693, ~: 123639)
[PASS] testFuzz_redeem_failsWithInsufficientShares(uint256) (runs: 1024, : 424762, ~: 424683)
[PASS] testFuzz_redeem_failsWithUnregisteredAsset(uint128) (runs: 1024, : 584012, ~: 584012)
[PASS] testFuzz_setDepegOffsets(uint256,uint256) (runs: 1024, : 36570, ~: 36693)
[PASS] testFuzz_setFeeReceiver(address) (runs: 1024, : 27302, ~: 27302)
[PASS] testFuzz_setMaxDelay(uint256) (runs: 1024, : 26771, ~: 26985)
[PASS] testFuzz_setMinSharesToRecapitalize(uint256) (runs: 1024, : 26869, ~: 26878)
[PASS] testFuzz_setMinSharesToRecapitalize_failsAmountOutOfRange(uint256) (runs: 1024, : 18855, ~: 18772)
[PASS] testFuzz_setMintRate(uint256) (runs: 1024, : 30379, ~: 30374)
[PASS] testFuzz_setMintRate_failsWithOverOneHundredPercentRate(uint256) (runs: 1024, : 21035, ~: 21023)
[PASS] testFuzz_setMintRate_failsWithUnderNinetyEightPercentRate(uint256) (runs: 1024, : 21036, ~: 20953)
[PASS] testFuzz_setPOLFeeCollector(address) (runs: 1024, : 27388, ~: 27391)
[PASS] testFuzz_setPOLFeeCollectorFeeRate(uint256) (runs: 1024, : 26767, ~: 26778)
[PASS] testFuzz_setRecapitalizeBalanceThreshold(uint256) (runs: 1024, : 46648, ~: 46921)
[PASS] testFuzz_setRedeemRate(uint256) (runs: 1024, : 30457, ~: 30488)
[PASS] testFuzz_setRedeemRate_failsWithOverOneHundredPercentRate(uint256) (runs: 1024, : 21016, ~: 21005)
[PASS] testFuzz_setRedeemRate_failsWithUnderNinetyEightPercentRate(uint256) (runs: 1024, : 21082, ~: 21000)
[PASS] testFuzz_withdrawAllFee(uint256) (runs: 1024, : 388841, ~: 390035)
[PASS] testFuzz_withdrawFee(uint256) (runs: 1024, : 371746, ~: 373528)
[PASS] testFuzz_withdrawFee_failsWithAssetNotRegistered() (gas: 580458)
[PASS] test_BaskedModeIsDisabledWhenBadAssetIsDepeggedAndFullyLiquidated_WhenRedeem() (gas: 757349)
[PASS] test_BasketModeEnabledWhenAllFeedsAreStale_WhenMint() (gas: 99276)
[PASS] test_BasketModeEnabledWhenStalePriceOnOneFeed_DisabledWhenMint() (gas: 134769)
[PASS] test_BasketModeEnabledWhenStalePriceOnOneFeed_EnabledWhenRedeem() (gas: 183907)
[PASS] test_CollectedFees() (gas: 393472)
[PASS] test_CreateVault() (gas: 999172)
[PASS] test_ForceBasketModeWhenMint() (gas: 63978)
[PASS] test_ForceBasketModeWhenRedeem() (gas: 107218)
[PASS] test_InitializeFactoryWithZeroAddresses() (gas: 8614788)
[PASS] test_Initialize_ParamsSet() (gas: 36050)
[PASS] test_IntegrationTest() (gas: 1159471)
[PASS] test_SetGlobalCap() (gas: 26512)
[PASS] test_SetPriceFeed_WhenAssetIsPegged() (gas: 391974)
[PASS] test_SetReferenceCollateral() (gas: 35392)
[PASS] test_SetRelativeCap() (gas: 29781)
[PASS] test_TransferOwnershipOfBeacon() (gas: 29504)
[PASS] test_TransferOwnershipOfBeaconFailsIfNotOwner() (gas: 22052)
[PASS] test_TransferOwnershipOfBeaconFailsIfZeroAddress() (gas: 23213)
[PASS] test_UpgradeAndDowngradeOfBeaconProxy() (gas: 1973450)
```

```
[PASS] test_UpgradeBeaconProxy() (gas: 343046)
[PASS] test_UpgradeBeaconProxyImplFailsIfNotOwner() (gas: 318697)
[PASS] test_UpgradeBeaconProxyOfPausedCollateralVault() (gas: 392316)
[PASS] test_UpgradeBeaconProxyToFaultyVault() (gas: 1950157)
[PASS] test_WithdrawFee_TestEvent() (gas: 602884)
[PASS] test__codesize() (gas: 214268)
[PASS] test_createAlreadyRegisteredVault() (gas: 25171)
[PASS] test_createVault_failsIfAssetIsDepegged() (gas: 1325766)
[PASS] test_createVault_failsWithZeroAddressAsset() (gas: 159422)
[PASS] test_createVault_failsWithoutDefaultAdmin() (gas: 21664)
[PASS] test_factoryPause() (gas: 42482)
[PASS] test_factoryPause_failsWhenAlreadyPaused() (gas: 43417)
[PASS] test_factoryPause_failsWithoutManager() (gas: 15985)
[PASS] test_factoryUnPause_failsWhenAlreadyUnpaused() (gas: 20230)
[PASS] test_factoryUnPause_failsWithoutManager() (gas: 16008)
[PASS] test_factoryUnpause() (gas: 32368)
[PASS] test_forceBasketMode_FailsWithoutManager() (gas: 16080)
[PASS] test_liquidate_WhenThereIsNoSufficientBadCollateral() (gas: 694993)
[PASS] test_liquidate_failsIfGoodCollateralIsBadCollateral() (gas: 55031)
[PASS] test_liquidate_failsIfGoodCollateralIsNotRegistered() (gas: 584408)
[PASS] test_liquidate_failsIfLiquidationIsNotEnabled() (gas: 38852)
[PASS] test_liquidate_failsIfReferenceCollateralIsBadCollateral() (gas: 61082)
[PASS] test_liquidate_failsWhenBadCollateralIsNotRegistered() (gas: 582227)
[PASS] test_liquidate_failsWhenBadCollateralIsNotRegisteredAsBadCollateral() (gas: 30885)
[PASS] test_liquidate_failsWithNoAllowance() (gas: 89762)
[PASS] test_mint() (gas: 355229)
[PASS] test_mint_failsIfDepositAssetWithZeroWeightWhenBasketModeIsEnabled() (gas: 408837)
[PASS] test_mint_failsWhenAssetIsDepegged() (gas: 398496)
[PASS] test_mint_failsWithBadCollateralAsset() (gas: 143791)
[PASS] test_pauseVault() (gas: 64264)
[PASS] test_pauseVault_failsWithoutManager() (gas: 18258)
[PASS] test_recapitalize_failsForInsufficientAllowance() (gas: 418844)
[PASS] test_recapitalize_failsIfAssetIsNotRegistered() (gas: 579958)
[PASS] test_recapitalize_failsIfBadAsset() (gas: 50565)
[PASS] test_redeem() (gas: 667322)
[PASS] test_redeem_failsWhenExceedsGlobalCap() (gas: 856973)
[PASS] test_redeem_failsWhenReferenceCollateralIsRedeemedAndExceedsRelativeCapOnOtherAssets() (gas: 848403)
[PASS] test_setCollateralAssetStatus() (gas: 49357)
[PASS] test_setCollateralAssetStatus_failsWithUnregisteredAsset() (gas: 582677)
[PASS] test_setCollateralAssetStatus_failsWithoutManager() (gas: 18298)
[PASS] test_setDepegOffsets_failsIfAssetIsNotRegistered() (gas: 582610)
[PASS] test_setDepegOffsets_failsOutOfRance() (gas: 22862)
[PASS] test_setDepegOffsets_failsWithoutManager() (gas: 18238)
[PASS] test_setFeeReceiver() (gas: 28546)
[PASS] test_setFeeReceiver_failsWithZeroAddress() (gas: 18155)
[PASS] test_setFeeReceiver_failsWithoutAdmin() (gas: 18190)
[PASS] test_setGlobalCap_failsWithoutManager() (gas: 15978)
[PASS] test_setLiquidationRate() (gas: 47464)
[PASS] test_setLiquidationRate_failsIfAssetIsNotRegistered() (gas: 582631)
[PASS] test_setLiquidationRate_failsWithoutAdminRole() (gas: 18258)
[PASS] test_setMaxDelay_failsOutOfRange() (gas: 18335)
[PASS] test_setMaxDelay_failsWithoutManager() (gas: 15999)
[PASS] test_setMinSharesToRecapitalize_failsWithoutAdminRole() (gas: 16001)
[PASS] test_setMintRate_failsWithoutManager() (gas: 18228)
[PASS] test_setPOLFeeCollector() (gas: 28595)
[PASS] test_setPOLFeeCollectorFeeRate() (gas: 26756)
[PASS] test_setPOLFeeCollectorFeeRate_failsWithOverOneHundredPercentRate(uint256) (runs: 1024, : 18778, ~: 18765)
[PASS] test_setPOLFeeCollectorFeeRate_failsWithoutAdminRole() (gas: 16013)
[PASS] test_setPOLFeeCollector_failsWithZeroAddress() (gas: 18154)
[PASS] test_setPOLFeeCollector_failsWithoutAdmin() (gas: 18186)
[PASS] test_setPriceFeed_failsWithoutManager() (gas: 18227)
[PASS] test_setRecapitalizeBalanceThreshold_failsWithoutAdminRole() (gas: 18230)
[PASS] test_setRedeemRate_failsWithoutManager() (gas: 18274)
[PASS] test_setReferenceCollateral_failsIfAssetIsNotRegistered() (gas: 582570)
[PASS] test_setReferenceCollateral_failsWithoutManager() (gas: 18149)
[PASS] test_setRelativeCap_failsWithoutManager() (gas: 18215)
[PASS] test_unpauseVault() (gas: 51443)
[PASS] test_unpauseVault_failsWithoutManager() (gas: 18193)
[PASS] test_withdrawFee_WithZeroCollectedFee() (gas: 43178)
Suite result: ok. 144 passed; 0 failed; 0 skipped; finished in 5.68s (15.66s CPU time)

Ran 2 tests for test/pol/e2e/POLE2EFuzz.t.sol:POLE2EFuzz
[PASS] testFuzzDistribution(uint96[],uint256,uint256) (runs: 1024, : 17225700, ~: 17923922)
[PASS] testGasPOLDistribution() (gas: 293999)
```

8.2.1 Slither

All the relevant issues raised by Slither have been incorporated into the issues described in this report.

8.2.2 AuditAgent

All the relevant issues raised by the AuditAgent have been incorporated into this report. The AuditAgent is an AI-powered smart contract auditing tool that analyses code, detects vulnerabilities, and provides actionable fixes. It accelerates the security analysis process, complementing human expertise with advanced AI models to deliver efficient and comprehensive smart contract audits. Available at https://app.auditagent.nethermind.io.

9 About Nethermind

Nethermind is a Blockchain Research and Software Engineering company. Our work touches every part of the web3 ecosystem - from layer 1 and layer 2 engineering, cryptography research, and security to application-layer protocol development. We offer strategic support to our institutional and enterprise partners across the blockchain, digital assets, and DeFi sectors, guiding them through all stages of the research and development process, from initial concepts to successful implementation.

We offer security audits of projects built on EVM-compatible chains and Starknet. We are active builders of the Starknet ecosystem, delivering a node implementation, a block explorer, a Solidity-to-Cairo transpiler, and formal verification tooling. Nethermind also provides strategic support to our institutional and enterprise partners in blockchain, digital assets, and decentralized finance (DeFi). In the next paragraphs, we introduce the company in more detail.

Blockchain Security: At Nethermind, we believe security is vital to the health and longevity of the entire Web3 ecosystem. We provide security services related to Smart Contract Audits, Formal Verification, and Real-Time Monitoring. Our Security Team comprises blockchain security experts in each field, often collaborating to produce comprehensive and robust security solutions. The team has a strong academic background, can apply state-of-the-art techniques, and is experienced in analyzing cutting-edge Solidity and Cairo smart contracts, such as ArgentX and StarkGate (the bridge connecting Ethereum and StarkNet). Most team members hold a Ph.D. degree and actively participate in the research community, accounting for 240+ articles published and 1,450+ citations in Google Scholar. The security team adopts customer-oriented and interactive processes where clients are involved in all stages of the work.

Blockchain Core Development: Our core engineering team, consisting of over 20 developers, maintains, improves, and upgrades our flagship product - the Nethermind Ethereum Execution Client. The client has been successfully operating for several years, supporting both the Ethereum Mainnet and its testnets, and now accounts for nearly a quarter of all synced Mainnet nodes. Our unwavering commitment to Ethereum's growth and stability extends to sidechains and layer 2 solutions. Notably, we were the sole execution layer client to facilitate Gnosis Chain's Merge, transitioning from Aura to Proof of Stake (PoS), and we are actively developing a full-node client to bolster Starknet's decentralization efforts. Our core team equips partners with tools for seamless node set-up, using generated docker-compose scripts tailored to their chosen execution client and preferred configurations for various network types.

DevOps and Infrastructure Management: Our infrastructure team ensures our partners' systems operate securely, reliably, and efficiently. We provide infrastructure design, deployment, monitoring, maintenance, and troubleshooting support, allowing you to focus on your core business operations. Boasting extensive expertise in Blockchain as a Service, private blockchain implementations, and node management, our infrastructure and DevOps engineers are proficient with major cloud solution providers and can host applications in-house or on clients' premises. Our global in-house SRE teams offer 24/7 monitoring and alerts for both infrastructure and application levels. We manage over 5,000 public and private validators and maintain nodes on major public blockchains such as Polygon, Gnosis, Solana, Cosmos, Near, Avalanche, Polkadot, Aptos, and StarkWare L2. Sedge is an open-source tool developed by our infrastructure experts, designed to simplify the complex process of setting up a proof-of-stake (PoS) network or chain validator. Sedge generates docker-compose scripts for the entire validator set-up based on the chosen client, making the process easier and quicker while following best practices to avoid downtime and being slashed.

Cryptography Research: At Nethermind, our Cryptography Research team is dedicated to continuous internal research while fostering close collaboration with external partners. The team has expertise across a wide range of domains, including cryptography protocols, consensus design, decentralized identity, verifiable credentials, Sybil resistance, oracles, and credentials, distributed validator technology (DVT), and Zero-knowledge proofs. This diverse skill set, combined with strong collaboration between our engineering teams, enables us to deliver cutting-edge solutions to our partners and clients.

Smart Contract Development & DeFi Research: Our smart contract development and DeFi research team comprises 40+ world-class engineers who collaborate closely with partners to identify needs and work on value-adding projects. The team specializes in Solidity and Cairo development, architecture design, and DeFi solutions, including DEXs, AMMs, structured products, derivatives, and money market protocols, as well as ERC20, 721, and 1155 token design. Our research and data analytics focuses on three key areas: technical due diligence, market research, and DeFi research. Utilizing a data-driven approach, we offer in-depth insights and outlooks on various industry themes.

Our suite of L2 tooling: Warp is Starknet's approach to EVM compatibility. It allows developers to take their Solidity smart contracts and transpile them to Cairo, Starknet's smart contract language. In the short time since its inception, the project has accomplished many achievements, including successfully transpiling Uniswap v3 onto Starknet using Warp.

- **Voyager** is a user-friendly Starknet block explorer that offers comprehensive insights into the Starknet network. With its intuitive interface and powerful features, Voyager allows users to easily search for and examine transactions, addresses, and contract details. As an essential tool for navigating the Starknet ecosystem, Voyager is the go-to solution for users seeking in-depth information and analysis;
- **Horus** is an open-source formal verification tool for StarkNet smart contracts. It simplifies the process of formally verifying Starknet smart contracts, allowing developers to express various assertions about the behavior of their code using a simple assertion language;
- **Juno** is a full-node client implementation for Starknet, drawing on the expertise gained from developing the Nethermind Client. Written in Golang and open-sourced from the outset, Juno verifies the validity of the data received from Starknet by comparing it to proofs retrieved from Ethereum, thus maintaining the integrity and security of the entire ecosystem.

Learn more about us at nethermind.io.

General Advisory to Clients

As auditors, we recommend that any changes or updates made to the audited codebase undergo a re-audit or security review to address potential vulnerabilities or risks introduced by the modifications. By conducting a re-audit or security review of the modified codebase, you can significantly enhance the overall security of your system and reduce the likelihood of exploitation. However, we do not possess the authority or right to impose obligations or restrictions on our clients regarding codebase updates, modifications, or subsequent audits. Accordingly, the decision to seek a re-audit or security review lies solely with you.

Disclaimer

This report is based on the scope of materials and documentation provided by you to Nethermind in order that Nethermind could conduct the security review outlined in **1. Executive Summary** and **2. Audited Files**. The results set out in this report may not be complete nor inclusive of all vulnerabilities. Nethermind has provided the review and this report on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. This report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on this report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, Nethermind disclaims any liability in connection with this report, its content, and any related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. Nethermind does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, 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, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and Nethermind will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. 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. 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.