Security Review Report
NM-0360 Berachain - Proof of Liquidity

NETHERMIND
SECURITY

(Nov 27, 2024)

Contents

1 Executive Summary

This document presents the security review performed by Nethermind Security for Berachain's Proof of Liquidity smart contracts. Berachain is an EVM Identical - Layer 1 blockchain that relies on Proof of Liquidity (PoL) as a consensus mechanism.

This novel mechanism creates a framework to reward not only validators but also other ecosystem participants like protocols/dApps and end users, incentivizing them to collaborate. Protocols/dApps create reward vaults where users (liquidity providers) stake their receipt tokens to earn extra rewards in Berachain Governance Tokens (BGT). Validators propose new blocks and distribute BGT rewards to reward vaults. BGT holders boost validators by staking BGT to accelerate the reward distribution further. Protocols can provide validators with additional incentives to attract them to distribute the rewards to their vaults. In this system, the dApps get liquidity, the users get extra rewards, and validators get incentives on top of the base reward for proposing a block.

The audited code comprises 2278 lines of code written in the Solidity language. The audit focuses on the implementation of Berachain's Proof of Liquidity contracts.

The audit was performed using (a) manual analysis of the codebase, (b) automated analysis tools, and (c) creation of test cases. **Along this document, we report** four points of attention, all of which are classified as `Informational` or `Best Practice`. 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 test suite evaluation. Section 9 concludes the document.

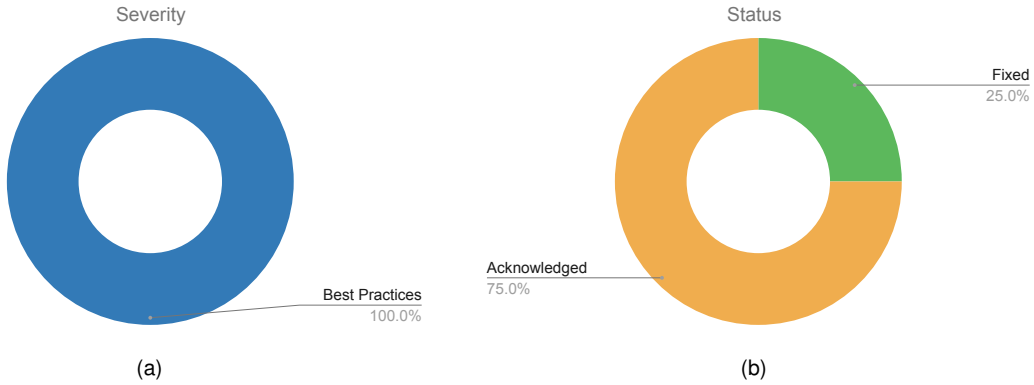

Fig. 1: Distribution of issues: Critical (0), **High** (0), **Medium** (0), **Low** (0), **Undetermined** (0), **Informational** (0), **Best Practices** (4). **Distribution of status: Fixed** (1), **Acknowledged** (3), **Mitigated** (0), **Unresolved** (0)

Summary of the Audit

Audit Type	Security Review
Initial Report	November 22, 2024
Final Report	November 27, 2024
Repository	contracts-monorepo
Commit	ba2fbe87beb75ded68b3a573e253ee1ab65b192c
Final Commit	97608d28b2b36777bbbd2616423e99a65dfca383
Documentation	Docs
Documentation Assessment	High
Test Suite Assessment	High

2 Audited Files

	Contract	LoC	Comments	Ratio	Blank	Total
1	src/pol/POLDeployer.sol	53	26	49.1%	14	93
2	src/pol/BeaconRootsHelper.sol	71	66	93.0%	30	167
3	src/pol/FeeCollector.sol	78	32	41.0%	25	135
4	src/pol/BGTStaker.sol	62	35	56.5%	23	120
5	src/pol/BeaconDeposit.sol	112	66	58.9%	35	213
6	src/pol/BGT.sol	299	130	43.5%	81	510
7	src/pol/BGTFeeDeployer.sol	23	13	56.5%	6	42
8	src/pol/rewards/RewardVault.sol	290	109	37.6%	64	463
9	src/pol/rewards/BlockRewardController.sol	116	54	46.6%	38	208
10	src/pol/rewards/BeraChef.sol	202	85	42.1%	53	340
11	src/pol/rewards/Distributor.sol	100	41	41.0%	28	169
12	src/pol/rewards/RewardVaultFactory.sol	72	36	50.0%	27	135
13	src/pol/interfaces/IBGT.sol	59	217	367.8%	46	322
14	src/pol/interfaces/IRewardVaultFactory.sol	12	51	425.0%	12	75
15	src/pol/interfaces/IBGTStaker.sol	12	36	300.0%	11	59
16	src/pol/interfaces/IBeaconDeposit.sol	22	71	322.7%	13	106
17	src/pol/interfaces/IRewardVault.sol	42	122	290.5%	42	206
18	src/pol/interfaces/IFeeCollector.sol	18	49	272.2%	19	86
19	src/pol/interfaces/IBeraChef.sol	43	119	276.7%	28	190
20	src/pol/interfaces/IPOLErrors.sol	65	75	115.4%	19	159
21	src/pol/interfaces/IDistributor.sol	14	17	121.4%	5	36
22	src/pol/interfaces/IERC165.sol	4	8	200.0%	1	13
23	src/pol/interfaces/IBlockRewardController.sol	32	92	287.5%	22	146
24	src/libraries/SSZ.sol	66	28	42.4%	12	106
25	src/libraries/BeaconRoots.sol	22	9	40.9%	4	35
26	src/libraries/Utils.sol	85	14	16.5%	15	114
27	src/base/IStakingRewardsErrors.sol	10	9	90.0%	1	20
28	src/base/Create2Deployer.sol	35	35	100.0%	10	80
29	src/base/IStakingRewards.sol	23	58	252.2%	22	103
30	src/base/StakingRewards.sol	176	78	44.3%	51	305
31	src/base/FactoryOwnable.sol	60	30	50.0%	22	112
	Total	**2278**	**1811**	**79.5%**	**779**	**4868**

3 Summary of Issues

	Finding	Severity	Update
1	Missing `address(0)` checks	Best Practices	Acknowledged
2	Missing event emission in `intialize(...)` functions	Best Practices	Acknowledged
3	Missing initialization of upgradeable contracts	Best Practices	Acknowledged
4	The parameters affecting the `BGT` token mints should be constrained	Best Practices	Fixed

4 System Overview

Berachain is an EVM Identical - Layer 1 blockchain that relies on Proof of Liquidity (PoL) as a consensus mechanism. Proof of Liquidity separates the concept of the gas token that is used to pay for transactions and to secure the network from the token used to govern the chain rewards. The gas token of the Berachain is `BERA`, and the governance token minted as a reward for proposing the block is `BGT`. Compared to Ethereum, where only the validator's operator is rewarded with Ether for proposing a block, in Berachain, the BGT block rewards are also given to different Reward Vaults. The exact Reward Vaults that will be distributed are selected by the Validator.

End users interacting with Berachain dApps can stake their receipt tokens, e.g., DEX's LP tokens, in Reward Vaults to earn the BGT distribution. With BGT, users can boost the validators of their choice to mint more BGT rewards the next time they propose a block. This effectively means that if the user has a stake in the Reward Vault selected by the validator, it is in the user's best interest to boost this validator to distribute more BGT rewards to this vault in the future.

The Berachain Governance Token (BGT) is a soulbound token. It is non-transferable but can be redeemed by users for BERA tokens in a 1:1 ratio by interacting with the `BGT` contract's `redeem(...)` function.

PoL incentivizes all the network participants to collaborate to achieve the greatest benefits for all the parties involved.

4.1 Actors

- **Validators**: The validators are responsible for running the nodes, which are the underlying infrastructure of Berachain. Creating new validators is permissionless as long as the user (operator) can provide at least `32` BERA to the Beacon Deposit contract. The BGT block rewards are split between the validator and reward vaults. While part of the BGT distribution goes to the reward vaults, the validators can earn extra incentives by targeting specific vaults supplied with external incentives by Berachain protocols.
- **BGT Holders**: BGT is used to participate in the chain's governance by voting or delegating the voting power to another address. BGT holders can delegate their BGT to boost validators to increase the rewards distributed to reward vaults.
- **Protocols/dApps**: The protocols/dApps can launch new reward vaults and supply them with token incentives to attract validators to distribute the BGT rewards to these vaults. The incentives are proportional to the distributed rewards.
- **LPs**: Users that provide liquidity to protocols (LPs) are rewarded with receipt tokens. These tokens can be subsequently staked in reward vaults to earn BGT rewards.
- **Bera Foundation**: The foundation operates default dApps (Bex, Bend, Berps) and corresponding reward vaults. The fees generated from such dApps are distributed to BGT holders via the Fee Collection mechanism.

4.2 Rewards Distribution

An active validator selected to propose a new block is responsible for BGT distribution. A set of reward vaults, along with their weights set by the validator in the BeraChef contract as reward allocation, will get the reward distributed according to their respective weights. While vault deployment is permissionless via the factory contract, each of the reward vaults must be whitelisted by the governance. If the validator chooses no vault, the default selection set by the governance will be used. Vault managers can add incentive tokens to the vault to attract validators to select their vaults for BGT distribution. The incentive tokens given to the validator are proportional to the amount of BGT distributed to the vault.

4.3 Earning & Delegating BGT

Whenever the validator proposes a new block, BGT is minted. The validator is rewarded with a portion of BGT equal to the base rate set by the governance, while the rest of the BGT is distributed to reward vaults in the form of rewards. With more BGT being delegated to a validator, the more boost it will have, and the higher the rewards minted on the blocks proposed in the future will be. As a result, the reward rate that the users will get on their staked BGT will also be higher. With more BGT being distributed as rewards in the reward vault, more users will be inclined to provide liquidity to the protocol to get the extra BGT yield.

Validators will favor the vaults that provide them with the most incentives. The validator's operator can select the new reward allocation by interacting with the Bera Chef contract.

4.4 Fee collection

During their operations, native dApps generate fees that will be sent to the Fee Collector contract as extra rewards for BGT holders which can be exchanged for a pre-set amount of WBERA tokens in a permissionless manner. Once the opportunity becomes profitable, the MEV Searcher can profit from the fee token price difference and buy the tokens cheaper by providing WBERA to the protocol. These wrapped tokens are sent to the BGT Staker contract.

BGT Holders who delegate their BGT to boost validators are considered stakers in the BGT Staker contract and are eligible for WBERA token rewards provided by the searcher.

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 [Best Practices] Missing `address(0)` checks

File(s): `src/pol/POLDeployer.sol`, `src/pol/BGTFeeDeployer.sol`

Description: The construction arguments of `POLDeployer` and `BGTFeeDeployer` contracts are validated in the respective `initialize(...)` functions called in the constructors. Input validation is performed on all arguments except for the `bgt` address.

Recommendation(s): To avoid deploying the protocol with an incorrect state, consider validating that the `bgt` argument passed to the constructor is non-zero.

Status: Acknowledged.

Update from the client: ACK. Checks are already performed in deployment scripts before creating those contracts.

6.2 [Best Practices] Missing event emission in `intialize(...)` functions

File(s): `src/pol/BGT.sol`, `src/pol/rewards/RewardVaultFactory.sol`

Description: During the project's deployment, it is worth emitting events with the initial values for important protocol parameters. These make tracking on-chain issues easier in case they arise post-deployment. No such events are emitted:

- In the `initialize(...)` function of the `BGT` contract for `activateBoostDelay`, `dropBoostDelay`, and `activateCommissionChangeDelay` state variables;
- In the `initialize(...)` function of the `RewardVaultFactory` contract for the `vaultImpl` address;

Such events are already emitted in the respective setter functions.

Recommendation(s): Consider emitting events with initial values in the `initialize(...)` functions.

Status: Acknowledged.

6.3 [Best Practices] Missing initialization of upgradeable contracts

File(s): `src/pol/*`

Description: The contracts that inherit OpenZeppelin's upgradeable contracts require initialization through their respective `_init(...)` functions, e.g., `__Pausable_init()`, `__UUPSUpgradeable_init()`, `__ERC20Votes_init()`, `__Votes_init()`, `__ReentrancyGuard_init()` or `__AccessControl_init()`. While some of these initializers are empty in the current version of OpenZeppelin contracts, this might change in the future. If the OZ version was upgraded in subsequent code iterations, some of the contracts could be left uninitialized by accident.

While some of the Berachain contracts follow this practice, e.g., the `BerachainGovernance` contract, there are places where initializers could be added for consistency reasons:

- `FeeCollector`;
- `BGTStaker`;
- `BGT`;
- `RewardVaultFactory`;
- `RewardVault`;
- `Distributor`;
- `BlockRewardController`;
- `BeraChef`;

Recommendation(s): Consider adding respective `init` functions in the `initialize()` functions of contracts outlined above.

Status: Acknowledged.

Update from the client: ACK.

6.4 [Best Practices] The parameters affecting the BGT token mints should be constrained

File(s): src/pol/rewards/BlockRewardController.sol

Description: The BlockRewardsController contract is responsible for processing the block rewards. It computes the BGT reward amounts and mints the tokens to the Distributor contract. It is also responsible for minting the base rate of BGT tokens to the validator's operator as a reward for proposing a block. The amount of BGT tokens minted whenever the new block is proposed depends on a couple of parameters controlled by the governance, i.e., the reward rate, convexity, and boost multiplier. Even though the governance controls these values, they could still be set to unreasonable values, e.g., in the case of a governance attack. The Berachain team projected the values for these parameters in a quantitative analysis. Given that the reasonable boundaries are known, constraining these parameters is considered a best practice.

Recommendation(s): Consider constraining the range of values that can be used in the setter functions of the BlockRewardsController contract. In particular, the setRewardRate(...) currently has no boundaries.

Status: Fixed.

Update from the client: Fixed in #482.

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 the Berachain documentation

The documentation for the Berachain PoL contracts was provided through their official docs as well as the extra reference materials such as the contract's specifications, the technical whitepaper and quantitative analysis documents. This documentation provided a high-level overview of the protocol and details of its implementation. Moreover, the Berachain team addressed all questions and concerns raised by the Nethermind Security team, providing valuable insights and a comprehensive understanding of the project's technical aspects.

8 Test Suite Evaluation

Berachain's test suite is comprehensive and covers all the core flows of the protocol. To further enhance it and determine potential uncovered code paths, the Nethermind Security team applied a technique called mutation testing. This technique introduces slight modifications to the code called "mutations" or "mutants." An example of a mutation is, for example, changing the operator in an expression `(a + b)` to `(a - b)`, or removing a `require(a > b)` statement entirely from the code. With these changes, the code no longer follows the expected business logic of the application, and the test suite should reflect that by failing.

Evaluation of the test suite with mutation testing consists of two phases:

a. Generating the modified version of each contract, called "mutants."

b. Inserting the mutant into the original codebase and running the test suite.

Only one modification can be tested at a time. If the contract has ten mutations, the test suite must run ten times (once for every mutation). If any of the tests fail, it means that the test suite caught the change in the code. Whenever that happens, the particular mutant is considered "slain" or "killed" and is removed from the mutant's set. If that does not occur, a new test case can be added to cover the code branch to "kill" the mutant.

The following table outlines the results of the analysis performed on Berachain's PoL core contracts. The first column lists the contracts that were tested using mutation testing. The second column indicates the number of mutants generated and how many were "slain" (i.e., caught by the test suite). The third column provides the percentage of mutants slain, reflecting the effectiveness of the test suite in covering the particular contract.

Contract	Mutants (slain / total generated)	Score
`RewardVault.sol`	204/211	96.68%
`FeeCollector.sol`	35/42	83.33%
`StakingRewards.sol`	183/186	98.39%
`RewardVaultFactory.sol`	15/16	93.75%
`BGTFeeDeployer.sol`	4/4	100%
`BGTStaker.sol`	16/18	88.89%
`BeaconRootsHelper.sol`	44/53	83.02%
`SSZ.sol`	92/97	94.85%
`Utils.sol`	8/10	80.00%
`BGT.sol`	161/178	90.45%
`BeraChef.sol`	101/110	91.82%
`BlockRewardController.sol`	93/106	87.74%
`Distributor.sol`	37/43	86.05%
`FactoryOwnable.sol`	10/10	100%
`BeaconDeposit.sol`	66/71	92.96%
`POLDeployer.sol`	5/5	100%

The following is a list of code modifications not caught by the existing test suite. Each point highlights a scenario that could benefit from higher test coverage to enhance the protocol's overall security and resilience in the next code change iterations as the project evolves.

RewardsVault.sol

- No test case ensures that the function `delegateStake(...)` reverts when the `msg.sender` is equal to the `account`. A test can be added to validate this behavior.
- The exact amount of `rateDelta` in the `addIncentive(...)` function (calculated as `incentiveRate - storedRate`) can be arbitrarily changed e.g. using modulo operation (`%`) instead of subtraction (`-`), and the tests still pass. A test case could be added to ensure that `rateDelta` is calculated and validated as expected.
- The `_processIncentives(...)` function is not tested for cases where the `amount` is equal to `0`. A test case could ensure that this edge case is handled as intended.

RewardVaultFactory.sol

- No test case for `_disableInitializers()`. Initializers on the implementation contracts shouldn't be callable after the deployment.

StakingRewards.sol

- `_updateReward(...)`: No test case for when `account == 0` (e.g., to test that account info was not updated).
- `_computeLeftOverReward(...)`: No test case for the exact remaining time computed (e.g., subtraction can be replaced with modulo or addition operations, and tests still pass).
- `_getReward(...)`: No test case for when `reward == 0` (e.g., unclaimed rewards are `0`).

SSZ.sol

- No test case for revert with invalid public key length.

- `uint64HashTreeRoot(...)`: The bitwise OR operator (`|`) can be replaced with addition (`+`) in all three occurrences without affecting the test suite.

Utils.sol

- `safeIncreaseAllowance(...)`: No test case for `IncreaseAllowanceOverflow` error.

BGT.sol

- No test case for `NAME` and `SYMBOL` values after deployment.
- The `activateBoostDelay` can be left unset (`0` default value) with no effect on the test suite.
- The `activateCommissionChangeDelay` can be left unset (`0` default value) with no effect on the test suite.
- No test case for setting the beacon deposit contract to the `0` address in `setBeaconDepositContract(...)`.
- No test case for setting the staker to the `0` address in `setStaker(...)`.
- In `activateBoost(...)`, no test case ensures that the `userBoosts[user]` queue boost amounts are updated correctly (e.g., subtracting queued boost - can be replaced with a different operand and tests still pass).
- In `dropBoost(...)`, there is no test case ensuring that the element `dropBoostQueue[user][pubkey]` is no longer accessible after the `dropBoost(...)` call.
- In `dropBoost(...)`, the expression `staker.withdraw(user, amount)` can be removed entirely with no effect on the test suite.
- In `activateCommissionChange(...)`, there is no test case ensuring that the element `queuedCommissions[pubkey]` is no longer accessible after deletion.
- No test case exists for the value returned by the `unboostedBalanceOf(...)` function.

BGTStaker.sol

- No test case for `_disableInitializers()`.
- The token transfer in `recoverERC20(...)` can be removed with no effect on the test suite.

BeaconDeposit.sol

- The `InvalidSignatureLength` revert is not tested in `deposit(...)`.
- The `DepositValueTooHigh` revert is not tested in `_deposit(...)`.
- In `_deposit(...)`, the expression `_safeTransferETH(address(0), msg.value)` can be removed with no effect on the test suite.

BeaconRootsHelper.sol

- The revert `InvalidProof` in the `_verifyProposerIndexInBeaconBlock(...)` function is not tested (this is also the case in `_verifyValidatorPubkeyInBeaconBlock(...)`).
- The revert `IndexOutOfRange` in the `_verifyValidatorPubkeyInBeaconBlock(...)` function is not tested.

FeeCollector.sol

- No test case for `_disableInitializers()`.
- In the `claimFees(...)` function, the expression `BGTStaker(rewardReceiver).notifyRewardAmount(payoutAmount)` can be removed with no effect on the test suite.
- In the `donate(...)` function, the expression `BGTStaker(rewardReceiver).notifyRewardAmount(amount)` can be removed with no effect on the test suite.
- There is no test case ensuring that the `queuedPayoutAmount` was reset to `0` after calling `_setPayoutAmount(...)`.

BeraChef.sol

- No test case for `_disableInitializers()`.
- In the `initialize(...)` function, the revert `MaxNumWeightsPerRewardAllocationIsZero` is not tested.
- No test case for `NotOperator` revert in the `onlyOperator` modifier.
- No test case ensures that `setDefaultRewardAllocation(...)` validates the weights.
- No test case for the `InvalidStartBlock` revert in the `queueNewRewardAllocation(...)` function at the boundary timestamps (e.g. the condition `if (startBlock <= block.number + rewardAllocationBlockDelay)` can be changed to use `-` instead of `+`).

BlockRewardController.sol

- No test case for `_disableInitializers()`.
- No test case for revert `InvalidBoostMultiplier` in `setBoostMultiplier(...)`.
- No test case for revert `InvalidRewardConvexity` in `setRewardConvexity(...)`.

- There is no test case validating if the `distributor` was actually set in `setDistributor(...)`.
- There is no test case hitting the `_rewardConvexity == 0 || boostPower == one` branch in `computeReward(...)`.
- There is no test case checking the exact value of the coefficient in the `coeff > _boostMultiplier` branch used for reward computation. The `coeff` can be set to an arbitrary number, which impacts reward calculation without affecting the test suite.
- In `processRewards(...)` there is no test case where `BGT` is not minted to the operator (e.g. when `base + commission == 0`). The same issue occurs with the `BGT` mint to the distributor (`reward == 0`).

Distributor.sol

- No test case for `_disableInitializers()`.
- No test case covering the effects of calling `setZeroValidatorPubkeyGIndex(...)` and `setProposerIndexGIndex(...)`.
- In `distributeFor(...)` a call to `_verifyProposerIndexInBeaconBlock(...)` can be removed with no effect on the test suite. The same applies to `_verifyValidatorPubkeyInBeaconBlock(...)`.
- There is no test case testing the early return in `_distributeFor(...)` when `!beraChef.isReady() || rewardRate == 0`.

RewardVault.sol

- No test case for `_disableInitializers()`.
- No test case for revert with `NotDistributor` in `onlyDistributor` modifier.

8.1 Tests Output

```
 forge test
[] Compiling...
[] Compiling 262 files with Solc 0.8.26
[] Solc 0.8.26 finished in 20.75s
Compiler run successful!

Ran 4 tests for test/base/BeaconRoots.t.sol:BeaconRootsTest
[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 5.91ms (1.63ms 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 6.34ms (1.31ms 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 567.33µs (376.00µs 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 395.08µs (167.92µs CPU time)

Ran 3 tests for test/berps/integration/TestTradingFee.t.sol:TestTradingFee
[PASS] testTradingCloseMarket() (gas: 1670236)
Logs:
  honey balance: 5000000000000000000
  honey balance: 84100000003599999995

[PASS] testTradingCloseSL() (gas: 1761424)
Logs:
  honey balance: 5000000000000000000
  honey balance: 43510000000438750000

[PASS] testTradingCloseTP() (gas: 1686776)
Logs:
  honey balance: 5000000000000000000
  honey balance: 93100000004499999994

Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 9.62ms (5.75ms CPU time)

Ran 2 tests for test/berps/integration/TestTradingMisc.t.sol:TestTradingMisc
[PASS] testOpenCloseSameBlock() (gas: 1451906)
[PASS] testTradingSL() (gas: 1612173)
Logs:
  honey balance: 5000000000000000000
  honey balance: 4000000000000000000
  honey balance: 4000000000000000000
  honey balance: 48099999999693243635

Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 4.97ms (2.69ms CPU time)
```

```
Ran 5 tests for test/berps/integration/TestTradingValue.t.sol:TestTradingValue
[PASS] testRefundForDelegate() (gas: 1154395)
[PASS] testRefundForDelegateWithMulticall() (gas: 1733340)
[PASS] testRefundInMulticall() (gas: 1704838)
[PASS] testRefundNoMulticall() (gas: 1238093)
[PASS] testRefundWithMulticall() (gas: 1811080)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 15.82ms (12.47ms 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, : 31061, ~: 31034)
[PASS] testFuzz_SetRewardAllocationBlockDelay(uint64) (runs: 1024, : 30245, ~: 30037)
[PASS] testFuzz_SetRewardAllocationBlockDelay_FailsIfDelayTooLarge(uint64) (runs: 1024, : 22966, ~: 22919)
[PASS] testFuzz_updateWhitelistedVaults(bool) (runs: 1024, : 38771, ~: 47318)
[PASS] test_ActivateARewardAllocation() (gas: 170165)
[PASS] test_DeleteQueuedRewardAllocationAfterActivation() (gas: 220836)
[PASS] test_EditDefaultRewardAllocationBeforeRemoveAVaultFromWhitelist() (gas: 1509267)
[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: 1632090)
[PASS] test_GetActiveRewardAllocationReturnsDefaultRewardAllocationAfterSetMaxWeights() (gas: 1690892)
[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 165.65ms (162.31ms 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 1.65ms (1.35ms CPU time)

Ran 3 tests for test/berps/v0/TestVault.t.sol:TestVault
[PASS] testAssetsPnL(uint48) (runs: 1024, : 207914, ~: 208668)
[PASS] testBalanceOf() (gas: 454733)
[PASS] testDistributeReward(uint48) (runs: 1024, : 272181, ~: 276879)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 258.27ms (257.02ms CPU time)

Ran 27 tests for test/gov/BerachainGovernance.t.sol:BerachainGovernanceTest
[PASS] testFuzz_UpgradeGovernance_FailsIfNotOwner(address) (runs: 1024, : 5795323, ~: 5795323)
[PASS] testFuzz_UpgradeTimelock_FailsIfNotOwner(address) (runs: 1024, : 2334611, ~: 2334611)
[PASS] test_AcceleratedProposal() (gas: 976936)
[PASS] test_CancelProposal() (gas: 463168)
[PASS] test_CancelProposalFailsIfCallerNotProposer() (gas: 453725)
[PASS] test_CancelProposalFailsIfProposalActive() (gas: 469939)
[PASS] test_CreateFailedProposal() (gas: 523932)
[PASS] test_CreatePendingProposal() (gas: 471557)
[PASS] test_CreateSucceededProposal() (gas: 526475)
```

```
[PASS] test_ExecuteProposal() (gas: 738105)
[PASS] test_ExecuteProposal_FailsIfProposalNonExistent() (gas: 55312)
[PASS] test_ExecuteProposal_FailsIfProposalNotQueued() (gas: 452179)
[PASS] test_ExecuteProposal_FailsIfSucceededButNotQueued() (gas: 574155)
[PASS] test_GuardianCanCancelProposalWhenReady() (gas: 675824)
[PASS] test_GuardianCanCancelProposalWhenWaiting() (gas: 675293)
[PASS] test_GuardianCancelProposalFailsIfAddressDoesNotHaveRole() (gas: 661040)
[PASS] test_ProposeFailsIfInsufficienVotes() (gas: 65656)
[PASS] test_ProposeFailsWithInvalidDescription() (gas: 56398)
[PASS] test_QueueProposal() (gas: 649555)
[PASS] test_QueueProposalFailsIfProposalCanceled() (gas: 468268)
[PASS] test_QueueProposalFailsIfProposalNonExistent() (gas: 55167)
[PASS] test_QueueProposalFailsIfProposalNotSuccess() (gas: 452109)
[PASS] test_UpgradeGovernanceViaVoting() (gas: 6330371)
[PASS] test_UpgradeGovernance_FailsIfNotOwner() (gas: 5795189)
[PASS] test_UpgradeTimelockViaVoting() (gas: 2850600)
[PASS] test_UpgradeTimelockWithCallerAsTimelock() (gas: 2339925)
[PASS] test_VoteCastingFailsIfMissingVotingPower() (gas: 482365)
Suite result: ok. 27 passed; 0 failed; 0 skipped; finished in 424.73ms (418.34ms CPU time)

Ran 25 tests for test/pol/FeeCollector.t.sol:FeeCollectorTest
[PASS] testFuzz_Donate(uint256) (runs: 1024, : 203024, ~: 203176)
[PASS] testFuzz_Donate_FailsIfAmountLessThanPayoutAmount(uint256) (runs: 1024, : 18146, ~: 17816)
[PASS] testFuzz_Donate_FailsIfNotApproved(uint256) (runs: 1024, : 52976, ~: 52919)
[PASS] testFuzz_Donate_FailsIfPaused(uint256) (runs: 1024, : 154923, ~: 154680)
[PASS] testFuzz_PayoutAmountChangeAfterClaim(uint256) (runs: 1024, : 177697, ~: 177708)
[PASS] testFuzz_PayoutAmountDoesntChangeIfClaimFails(uint256) (runs: 1024, : 103263, ~: 103259)
[PASS] testFuzz_QueuePayoutAmountChange(uint256) (runs: 1024, : 49479, ~: 49476)
[PASS] test_ClaimFees() (gas: 161702)
[PASS] test_ClaimFees_FailsIfNotApproved() (gas: 64597)
[PASS] test_ClaimFees_FailsIfPaused() (gas: 111321)
[PASS] test_Donate() (gas: 202716)
[PASS] test_Donate_FailsIfAmountLessThanPayoutAmount() (gas: 17766)
[PASS] test_Donate_FailsIfNotApproved() (gas: 52613)
[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 840.05ms (829.20ms CPU time)

Ran 32 tests for test/honey/CollateralVault.t.sol:CollateralVaultTest
[PASS] testFuzz_depositIntoUSTVault(uint256) (runs: 1024, : 129257, ~: 130364)
[PASS] testFuzz_deposit_succeedsWithCorrectSender(uint256) (runs: 1024, : 129409, ~: 130063)
[PASS] testFuzz_mintFromUSTVault(uint256) (runs: 1024, : 131404, ~: 132172)
[PASS] testFuzz_mint_succeedsWithCorrectSender(uint256) (runs: 1024, : 146272, ~: 146875)
[PASS] testFuzz_redeemFromUSTVault(uint256) (runs: 1024, : 163524, ~: 163858)
[PASS] testFuzz_redeem_failWithInsufficientAllowance(uint256) (runs: 1024, : 131978, ~: 131937)
[PASS] testFuzz_redeem_failsWithInsufficientBalance(uint256) (runs: 1024, : 29622, ~: 29622)
[PASS] testFuzz_redeem_succeedsWithCorrectSender(uint256) (runs: 1024, : 162512, ~: 162868)
[PASS] testFuzz_withdrawFromUSTVault(uint256) (runs: 1024, : 165245, ~: 165607)
[PASS] testFuzz_withdraw_failsWithIncorrectSender(uint128) (runs: 1024, : 22666, ~: 22666)
[PASS] testFuzz_withdraw_failsWithInsufficientAllowance(uint256) (runs: 1024, : 133285, ~: 133243)
[PASS] testFuzz_withdraw_failsWithInsufficientBalance(uint256) (runs: 1024, : 37453, ~: 37453)
[PASS] testFuzz_withdraw_succeedsWithCorrectSender(uint256) (runs: 1024, : 164298, ~: 164478)
[PASS] test__codesize() (gas: 60075)
[PASS] test_deposit() (gas: 129856)
[PASS] test_deposit_whileItsPaused() (gas: 48312)
[PASS] test_deposit_withOutOwner() (gas: 20343)
[PASS] test_initializeVault_failsIfZeroAsset() (gas: 142408)
[PASS] test_initializeVault_failsIfZeroFactory() (gas: 119887)
[PASS] test_mint() (gas: 131565)
[PASS] test_mint_failsWithIncorrectSender() (gas: 22473)
[PASS] test_mint_whileItsPaused() (gas: 48356)
[PASS] test_pauseVault_succeedsWithCorrectSender() (gas: 47270)
[PASS] test_pausingVault_failsIfNotFactory() (gas: 20135)
```

```
[PASS] test_redeem() (gas: 157724)
[PASS] test_redeem_whileItsPaused() (gas: 48378)
[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: 158956)
[PASS] test_withdraw_whileItsPaused() (gas: 48443)
Suite result: ok. 32 passed; 0 failed; 0 skipped; finished in 1.10s (1.10s CPU time)

Ran 24 tests for test/pol/BlockRewardController.t.sol:BlockRewardControllerTest
[PASS] testFuzz_ComputeRewards(uint256,uint256,uint256,int256) (runs: 1024, : 18508, ~: 18647)
[PASS] testFuzz_ProcessRewards(uint256,uint256,uint256,uint256,uint256,uint256) (runs: 1024, : 772558, ~: 773368)
[PASS] testFuzz_SetBoostMultiplier(uint256) (runs: 1024, : 44555, ~: 44542)
[PASS] testFuzz_SetMinBoostedRewardRate(uint256) (runs: 1024, : 44043, ~: 44335)
[PASS] testFuzz_SetRewardConvexity(uint256) (runs: 1024, : 44576, ~: 44543)
[PASS] testFuzz_SetRewardRate(uint256) (runs: 1024, : 44080, ~: 44314)
[PASS] test_ComputeRewards() (gas: 26325)
[PASS] test_FailIfInitializeAgain() (gas: 24472)
[PASS] test_FailIfNotDistributor() (gas: 22475)
[PASS] test_FailIfNotOwner() (gas: 1767050)
[PASS] test_OwnerIsGovernance() (gas: 17516)
[PASS] test_ProcessRewards() (gas: 339200)
[PASS] test_ProcessRewardsMax() (gas: 551311)
[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: 1752038)
Suite result: ok. 24 passed; 0 failed; 0 skipped; finished in 749.64ms (747.59ms CPU time)

Ran 32 tests for test/honey/Honey.t.sol:HoneyTest
[PASS] testFuzz_Approve(address,uint256) (runs: 1024, : 40836, ~: 41031)
[PASS] testFuzz_Burn(uint256,uint256) (runs: 1024, : 77969, ~: 78238)
[PASS] testFuzz_Burn_FailIfInsufficientBalance(uint256,uint256) (runs: 1024, : 70674, ~: 71596)
[PASS] testFuzz_Mint(uint256) (runs: 1024, : 68044, ~: 68589)
[PASS] testFuzz_Mint_TotalSupplyOverflow(uint256,uint256) (runs: 1024, : 66449, ~: 66462)
[PASS] testFuzz_Transfer(address,uint256) (runs: 1024, : 80661, ~: 81050)
[PASS] testFuzz_TransferFrom(address,uint256) (runs: 1024, : 98502, ~: 99066)
[PASS] testFuzz_TransferFrom_FailsIfInsufficientAllowance(address,uint256) (runs: 1024, : 98853, ~: 98951)
[PASS] testFuzz_TransferFrom_FailsIfInsufficientBalance(address,uint256) (runs: 1024, : 98798, ~: 99227)
[PASS] testFuzz_Transfer_FailsIfInsufficientBalance(address,uint256) (runs: 1024, : 70752, ~: 70738)
[PASS] testFuzz_UpgradeTo_FailsIfNotOwner(address) (runs: 1024, : 1485030, ~: 1485030)
[PASS] test_Approve() (gas: 43032)
[PASS] test_Burn() (gas: 78136)
[PASS] test_Burn_FailIfInsufficientBalance() (gas: 31682)
[PASS] test_Burn_FailIfNotFactory() (gas: 15476)
[PASS] test_Initialize_FailsIfZeroAddresses() (gas: 1587478)
[PASS] test_MetaData() (gas: 14078)
[PASS] test_Mint() (gas: 68496)
[PASS] test_Mint_FailIfNotFactory() (gas: 15466)
[PASS] test_Permit() (gas: 95528)
[PASS] test_Permit_BadDeadlineReverts() (gas: 46146)
[PASS] test_Permit_BadNonceReverts() (gas: 46647)
[PASS] test_Permit_PastDeadlineReverts() (gas: 40700)
[PASS] test_Permit_ReplayReverts() (gas: 97964)
[PASS] test_Transfer() (gas: 82997)
[PASS] test_TransferFrom() (gas: 100756)
[PASS] test_TransferFrom_FailsIfInsufficientAllowance() (gas: 100917)
[PASS] test_TransferFrom_FailsIfInsufficientBalance() (gas: 101196)
[PASS] test_Transfer_FailsIfInsufficientBalance() (gas: 72666)
[PASS] test_UpgradeTo() (gas: 1132896)
[PASS] test_UpgradeTo_FailIfNotOwner() (gas: 1484895)
[PASS] test__codesize() (gas: 65479)
Suite result: ok. 32 passed; 0 failed; 0 skipped; finished in 605.71ms (604.69ms CPU time)
```

```
Ran 16 tests for test/pol/Distributor.t.sol:DistributorTest
[PASS] testFuzz_DistributeDoesNotLeaveDust(uint256) (runs: 1024, : 1877097, ~: 1877288)
[PASS] testFuzz_DistributeDoesNotLeaveDust(uint256,uint256,uint256,uint256,uint256) (runs: 1024, : 1935009, ~: 1939955)
[PASS] test_Distribute() (gas: 501996)
[PASS] test_DistributeAndActivateQueuedRewardAllocation() (gas: 590103)
[PASS] test_DistributeForNonReentrant() (gas: 3469151)
[PASS] test_DistributeMulticall() (gas: 658251)
[PASS] test_FailIfInitializeAgain() (gas: 24563)
[PASS] test_FailIfNotManager() (gas: 32216)
[PASS] test_FailIfNotOwner() (gas: 2115763)
[PASS] test_IsTimestampActionable_OutOfBuffer() (gas: 19266)
[PASS] test_IsTimestampActionable_Processing() (gas: 488502)
[PASS] test_OwnerIsGovernance() (gas: 16975)
[PASS] test_ProcessTimestamp_OutOfBuffer() (gas: 173923)
[PASS] test_ProcessTimestamp_Processing() (gas: 594395)
[PASS] test_UpgradeTo() (gas: 2096539)
[PASS] test_ZeroRewards() (gas: 296549)
Suite result: ok. 16 passed; 0 failed; 0 skipped; finished in 1.43s (1.43s CPU time)

Ran 1 test for test/pol/POLDeployer.t.sol:POLDeployerTest
[PASS] test_DeployPOL() (gas: 15645326)
Logs:
  POLDeployer init code size 52466
  BeraChef implementation address 0xF59ed392C804898DDfCfEFBB0a32aa360765AebF
  BeraChef init code hash
  0x677948307bd0b407e503f3b99df75ac6472782b912c4863f9ac3a1acc3baa94f
  BeraChef address 0x6aE0D2beA59Af03988091a3876e5C1924b5dd726
  BlockRewardController implementation address 0x8DA6E460844a35D96Ea54132d40563153448bC1E
  BlockRewardController init code hash
  0xb811c3e81f7c21f2e1c4dbcd9f72933e36ce9c151bb0be2f5cb75d26acc89977
  BlockRewardController address 0xa7F576464b63c0631d5d3f9BdB043Fc851Fe1baF
  Distributor implementation address 0xa48f9cC7c8276222279BF99E620Ac79EA119ad4d
  Distributor init code hash
  0x7c8d5981e92f0b207ac2a4d85c61f832d195a2cdb88b70da75abaa02bb2c29bf
  Distributor address 0x65dD9648dc7bBd3F8AbaF58eFE6fF54d75c9BA8C

Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 2.60ms (1.93ms CPU time)

Ran 2 tests for test/pol/e2e/POLGasSimulationMax.t.sol:POLGasSimulationMax
[PASS] testGasPOLDistribution() (gas: 1381308)
[PASS] testGasPOLDistributionCatchUp() (gas: 3099340)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 26.23ms (7.13ms CPU time)

Ran 24 tests for test/pol/BeaconDeposit.t.sol:BeaconDepositTest
[PASS] testFuzz_AcceptOperatorChange_FailsIfNotEnoughTime(uint256) (runs: 1024, : 168004, ~: 168272)
[PASS] testFuzz_DepositCount(uint256) (runs: 1024, : 1567653, ~: 1580583)
[PASS] testFuzz_DepositNotDivisibleByGwei(uint256) (runs: 1024, : 58603, ~: 58656)
[PASS] testFuzz_DepositWrongCredentials(bytes) (runs: 1024, : 21992, ~: 21993)
[PASS] testFuzz_DepositWrongMinAmount(uint256) (runs: 1024, : 90338, ~: 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 2.23s (2.37s 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 7.16ms (941.92µs CPU time)
```

```
Ran 1 test for test/pol/e2e/POLGasSimulationAdvance.t.sol:POLGasSimulationAdvance
[PASS] testGasPOLDistribution() (gas: 941340)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 13.08ms (582.96µs 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 101.38µs (37.58µs CPU time)

Ran 73 tests for test/honey/HoneyFactory.t.sol:HoneyFactoryTest
[PASS] testFuzz_CollectedFeesWithDifferentFeeRate(uint256,uint256) (runs: 1024, : 278560, ~: 287218)
[PASS] testFuzz_InflationAttack(uint256) (runs: 1024, : 327251, ~: 328405)
[PASS] testFuzz_PreviewHoneyToRedeem(uint64) (runs: 1024, : 45168, ~: 45168)
[PASS] testFuzz_PreviewRequiredCollateral(uint128) (runs: 1024, : 40267, ~: 40267)
[PASS] testFuzz_mint(uint256) (runs: 1024, : 286579, ~: 288620)
[PASS] testFuzz_mintWithHigherDecimalAsset(uint256) (runs: 1024, : 277509, ~: 289487)
[PASS] testFuzz_mintWithLowerDecimalAsset(uint256) (runs: 1024, : 287754, ~: 289671)
[PASS] testFuzz_mint_failWithPausedFactory(uint128) (runs: 1024, : 51450, ~: 51450)
[PASS] testFuzz_mint_failsWithInsufficientAllowance(uint256) (runs: 1024, : 32215, ~: 32193)
[PASS] testFuzz_mint_failsWithUnregisteredAsset(uint32) (runs: 1024, : 533434, ~: 533434)
[PASS] testFuzz_redeem(uint256) (runs: 1024, : 325017, ~: 325171)
[PASS] testFuzz_redeemWithHigherDecimalAsset(uint256) (runs: 1024, : 326310, ~: 326352)
[PASS] testFuzz_redeemWithLowerDecimalAsset(uint256) (runs: 1024, : 326251, ~: 326254)
[PASS] testFuzz_redeem_failWithPausedFactory(uint128) (runs: 1024, : 49229, ~: 49229)
[PASS] testFuzz_redeem_failsWithInsufficientHoneys(uint256) (runs: 1024, : 41965, ~: 41939)
[PASS] testFuzz_redeem_failsWithInsufficientShares(uint256) (runs: 1024, : 282459, ~: 282476)
[PASS] testFuzz_redeem_failsWithUnregisteredAsset(uint128) (runs: 1024, : 530983, ~: 530983)
[PASS] testFuzz_setFeeReceiver(address) (runs: 1024, : 27130, ~: 27130)
[PASS] testFuzz_setMintRate(uint256) (runs: 1024, : 30240, ~: 30239)
[PASS] testFuzz_setMintRate_failsWithOverOneHundredPercentRate(uint256) (runs: 1024, : 20876, ~: 20951)
[PASS] testFuzz_setMintRate_failsWithUnderNinetyEightPercentRate(uint256) (runs: 1024, : 20907, ~: 20829)
[PASS] testFuzz_setPOLFeeCollector(address) (runs: 1024, : 27088, ~: 27088)
[PASS] testFuzz_setPOLFeeCollectorFeeRate(uint256) (runs: 1024, : 26554, ~: 26516)
[PASS] testFuzz_setRedeemRate(uint256) (runs: 1024, : 30141, ~: 30175)
[PASS] testFuzz_setRedeemRate_failsWithOverOneHundredPercentRate(uint256) (runs: 1024, : 20857, ~: 20934)
[PASS] testFuzz_setRedeemRate_failsWithUnderNinetyEightPercentRate(uint256) (runs: 1024, : 20937, ~: 20851)
[PASS] testFuzz_withdrawAllFee(uint256) (runs: 1024, : 305348, ~: 307686)
[PASS] testFuzz_withdrawFee(uint256) (runs: 1024, : 292185, ~: 294902)
[PASS] testFuzz_withdrawFee_failsWithAssetNotRegistered() (gas: 531297)
[PASS] test_CollectedFees() (gas: 286970)
[PASS] test_CreateVault() (gas: 786429)
[PASS] test_InitializeFactoryWithZeroAddresses() (gas: 5234973)
[PASS] test_Initialize_ParamsSet() (gas: 35300)
[PASS] test_IntegrationTest() (gas: 592764)
[PASS] test_TransferOwnershipOfBeacon() (gas: 29458)
[PASS] test_TransferOwnershipOfBeaconFailsIfNotOwner() (gas: 21985)
[PASS] test_TransferOwnershipOfBeaconFailsIfZeroAddress() (gas: 23122)
[PASS] test_UpgradeAndDowngradeOfBeaconProxy() (gas: 1923177)
[PASS] test_UpgradeBeaconProxy() (gas: 331462)
[PASS] test_UpgradeBeaconProxyImplFailsIfNotOwner() (gas: 307123)
[PASS] test_UpgradeBeaconProxyToFaultyVault() (gas: 1911350)
[PASS] test_WithdrawFee_TestEvent() (gas: 318317)
[PASS] test__codesize() (gas: 100224)
[PASS] test_createAlreadyRegisteredVault() (gas: 22790)
[PASS] test_factoryPause() (gas: 42165)
[PASS] test_factoryPause_failsWhenAlreadyPaused() (gas: 43059)
[PASS] test_factoryPause_failsWithoutManager() (gas: 15838)
[PASS] test_factoryUnPause_failsWhenAlreadyUnpaused() (gas: 20054)
[PASS] test_factoryUnPause_failsWithoutManager() (gas: 15860)
[PASS] test_factoryUnpause() (gas: 31968)
[PASS] test_mint() (gas: 266851)
[PASS] test_mint_failsWithBadCollateralAsset() (gas: 54015)
[PASS] test_pauseVault() (gas: 64051)
[PASS] test_pauseVault_failsWithoutManager() (gas: 18044)
[PASS] test_redeem() (gas: 282817)
[PASS] test_setCollateralAssetStatus() (gas: 49329)
[PASS] test_setCollateralAssetStatus_failsWithSameState() (gas: 25060)
[PASS] test_setCollateralAssetStatus_failsWithUnregisteredAsset() (gas: 533430)
[PASS] test_setCollateralAssetStatus_failsWithoutManager() (gas: 18122)
[PASS] test_setFeeReceiver() (gas: 28333)
[PASS] test_setFeeReceiver_failsWithZeroAddress() (gas: 18025)
[PASS] test_setFeeReceiver_failsWithoutAdmin() (gas: 18107)
```

```
[PASS] test_setMintRate_failsWithoutManager() (gas: 18102)
[PASS] test_setPOLFeeCollector() (gas: 28313)
[PASS] test_setPOLFeeCollectorFeeRate() (gas: 26474)
[PASS] test_setPOLFeeCollectorFeeRate_failsWithOverOneHundredPercentRate(uint256) (runs: 1024, : 18594, ~: 18671)
[PASS] test_setPOLFeeCollectorFeeRate_failsWithoutManager() (gas: 15834)
[PASS] test_setPOLFeeCollector_failsWithZeroAddress() (gas: 17981)
[PASS] test_setPOLFeeCollector_failsWithoutAdmin() (gas: 18037)
[PASS] test_setRedeemRate_failsWithoutManager() (gas: 18125)
[PASS] test_unpauseVault() (gas: 51139)
[PASS] test_unpauseVault_failsWithoutManager() (gas: 18067)
[PASS] test_withdrawFee_WithZeroCollectedFee() (gas: 26748)
Suite result: ok. 73 passed; 0 failed; 0 skipped; finished in 2.54s (3.32s CPU time)

Ran 14 tests for test/base/Utils.t.sol:UtilsTest
[PASS] testFuzz_Allowance(address,address,uint256) (runs: 1024, : 36614, ~: 36828)
[PASS] testFuzz_SafeIncreaseAllowance(address,uint256) (runs: 1024, : 40362, ~: 40338)
[PASS] testFuzz_SafeIncreaseAllowance_Overflow(uint256,uint256) (runs: 1024, : 41293, ~: 41358)
[PASS] testFuzz_TrySafeTransferDoesNotExeedGasLimit(address) (runs: 1024, : 375277, ~: 375277)
[PASS] testFuzz_TrySafeTransferERC20(address,uint256) (runs: 1024, : 65330, ~: 65486)
[PASS] testFuzz_TrySafeTransferExeedGasLimit(address) (runs: 1024, : 2007925, ~: 2007925)
[PASS] testFuzz_TrySafeTransferMissingReturnToken(address,uint256) (runs: 1024, : 41900, ~: 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: 5713)
[PASS] test_SafeIncreaseAllowance() (gas: 42327)
[PASS] test_SafeIncreaseAllowance_IfTokenNotExists() (gas: 9541)
Suite result: ok. 14 passed; 0 failed; 0 skipped; finished in 3.74s (3.91s 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 212.40ms (213.20ms 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, : 214336, ~: 214390)
[PASS] testFuzz_PartialWithdraw(address,uint256,uint256) (runs: 1024, : 99445, ~: 99695)
[PASS] testFuzz_RecoverERC20(uint256) (runs: 1024, : 1036542, ~: 1036214)
[PASS] testFuzz_RecoverERC20FailsIfNotOwner(address) (runs: 1024, : 18794, ~: 18794)
[PASS] testFuzz_SetRewardDuration(uint256) (runs: 1024, : 28832, ~: 28828)
[PASS] testFuzz_SetRewardDurationFailsIfNotOwner(address) (runs: 1024, : 18685, ~: 18685)
[PASS] testFuzz_SetRewardsDurationDuringCycleEarned(uint256,uint256) (runs: 1024, : 431716, ~: 431773)
[PASS] testFuzz_Stake(address,uint256) (runs: 1024, : 85907, ~: 85948)
[PASS] testFuzz_StakeFailsIfNotBGT(address) (runs: 1024, : 18466, ~: 18466)
[PASS] testFuzz_Withdraw(address,uint256) (runs: 1024, : 75512, ~: 75528)
[PASS] testFuzz_WithdrawFailsIfInsufficientStake(address,uint256,uint256) (runs: 1024, : 88155, ~: 88319)
[PASS] testFuzz_WithdrawFailsIfNotBGT(address) (runs: 1024, : 18398, ~: 18398)
[PASS] testFuzz_getRewardWithLowTotalSupply(uint256) (runs: 1024, : 567039, ~: 567025)
[PASS] test_GetReward_AfterDuration(uint256,uint256) (runs: 1024, : 403828, ~: 403608)
[PASS] test_GetReward_BeforeDuration(uint256,uint256) (runs: 1024, : 403702, ~: 403699)
[PASS] test_GetReward_Notified_Twice(uint256,uint256) (runs: 1024, : 482809, ~: 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, : 93269, ~: 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 4.44s (4.49s CPU time)

Ran 96 tests for test/pol/BGT.t.sol:BGTTest
[PASS] testFuzz_ActivateBoost(address,address,uint256) (runs: 1024, : 349175, ~: 348918)
[PASS] testFuzz_ActivateBoost_ReturnsFalseIfNotEnoughTimePassed(uint256) (runs: 1024, : 218007, ~: 218128)
[PASS] testFuzz_ActivateCommissionChange(uint256) (runs: 1024, : 88171, ~: 88568)
[PASS] testFuzz_ActivateCommissionChange_FailsIfHistoryBufferNotPassed(uint256) (runs: 1024, : 78530, ~: 78650)
[PASS] testFuzz_Approve(address,address,uint256) (runs: 1024, : 66630, ~: 67019)
[PASS] testFuzz_CancelBoost(address,uint256,uint256) (runs: 1024, : 221153, ~: 222776)
[PASS] testFuzz_CancelBoost_FailsIfCancelledMoreThanQueuedBoost(uint256,uint256) (runs: 1024, : 201538, ~: 201901)
[PASS] testFuzz_CancelCommissionChange(uint256) (runs: 1024, : 52161, ~: 52212)
[PASS] testFuzz_CancelCommissionChange_FailsIfNotOperator(address) (runs: 1024, : 63767, ~: 63767)
[PASS] testFuzz_CancelDropBoost(address,uint256,uint256) (runs: 1024, : 408894, ~: 408942)
[PASS] testFuzz_CancelDropBoost_FailsIfCancelledMoreThanQueuedDropBoost(uint256,uint256) (runs: 1024, : 399641, ~:
↪  399574)
[PASS] testFuzz_DropBoost(address,uint256,uint256) (runs: 1024, : 408771, ~: 410918)
[PASS] testFuzz_DropBoost_ReturnsFalseIfNotEnoughTimePassed(uint256) (runs: 1024, : 396938, ~: 397115)
[PASS] testFuzz_Mint(address,uint256) (runs: 1024, : 111623, ~: 112759)
[PASS] testFuzz_MultiCall_CancelBoostInBatch(address,uint256,uint256) (runs: 1024, : 271651, ~: 271760)
[PASS] testFuzz_MultiCall_QueueBoostInBatch(address,uint256,uint256) (runs: 1024, : 252238, ~: 252344)
[PASS] testFuzz_MultiCall_QueueDropBoostInBatch(uint256,uint256,uint256) (runs: 1024, : 389284, ~: 389587)
[PASS] testFuzz_QueueBoost(address,uint256) (runs: 1024, : 194365, ~: 195132)
[PASS] testFuzz_QueueCommissionChange(uint256) (runs: 1024, : 59222, ~: 59290)
[PASS] testFuzz_QueueCommissionChange_FailsIfNotOperator(address) (runs: 1024, : 26941, ~: 26941)
[PASS] testFuzz_QueueCommissionChange_FailsIfOverTenPercent(uint256) (runs: 1024, : 26584, ~: 26448)
[PASS] testFuzz_QueueDropBoost(address,uint256,uint256) (runs: 1024, : 393356, ~: 393394)
[PASS] testFuzz_QueueDropBoost_FailsIfDroppedMoreThanBoost(uint256,uint256) (runs: 1024, : 357639, ~: 357601)
[PASS] testFuzz_Redeem(uint256) (runs: 1024, : 168741, ~: 168988)
[PASS] testFuzz_SetActivateBoostDelay(uint256) (runs: 1024, : 21008, ~: 21203)
[PASS] testFuzz_SetActivateCommissionChangeDelay(uint256) (runs: 1024, : 20966, ~: 21169)
[PASS] testFuzz_SetDropBoostDelay(uint256) (runs: 1024, : 21007, ~: 21208)
[PASS] testFuzz_SetMinter(address) (runs: 1024, : 25329, ~: 25329)
[PASS] testFuzz_Transfer(address,address,uint256) (runs: 1024, : 155181, ~: 155891)
[PASS] testFuzz_TransferFrom(address,address,uint256,uint256) (runs: 1024, : 156781, ~: 158303)
[PASS] testFuzz_WhitelistSender(address,bool) (runs: 1024, : 30533, ~: 40037)
[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: 450246)
[PASS] test_MultiCall_ActivateBoostInBatch_DoesNotFailIfNotEnoughTimePassed() (gas: 381059)
[PASS] test_MultiCall_CancelBoostInBatch() (gas: 272959)
[PASS] test_MultiCall_QueueBoostInBatch() (gas: 253612)
[PASS] test_MultiCall_QueueBoostInBatch_FailsIfNotEnoughUnboostedBalance() (gas: 25208)
[PASS] test_MultiCall_QueueDropBoostInBatch() (gas: 388134)
[PASS] test_MultiCall_SetPramsInBatch() (gas: 33022)
[PASS] test_MultiCall_SetPramsInBatch_FailsIfNotOwner() (gas: 14978)
[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 5.99s (6.70s CPU time)

Ran 119 tests for test/pol/RewardVault.t.sol:RewardVaultTest
[PASS] testFuzz_AddIncentive_FailsIfAmountLessThanMinRate(uint256) (runs: 1024, : 298116, ~: 297788)
[PASS] testFuzz_AddIncentive_FailsIfRateMoreThanMaxIncentiveRate(uint256) (runs: 1024, : 297162, ~: 297433)
[PASS] testFuzz_AddIncentive_IncentiveRateNotChanged(uint256) (runs: 1024, : 566571, ~: 566589)
[PASS] testFuzz_AddIncentive_IncreaseIncentiveRate(uint256) (runs: 1024, : 567074, ~: 567171)
[PASS] testFuzz_AddIncentive_UpdatesIncentiveRate(uint256,uint256) (runs: 1024, : 556343, ~: 556590)
[PASS] testFuzz_ChangeInFactoryOwner(address) (runs: 1024, : 66443, ~: 66443)
[PASS] testFuzz_DelegateStake(address,address,uint256) (runs: 1024, : 216406, ~: 216272)
[PASS] testFuzz_DelegateWithdraw(address,address,uint256,uint256) (runs: 1024, : 258589, ~: 258595)
[PASS] testFuzz_DelegateWithdrawFailsIfNotEnoughStakedByDelegate(uint256,uint256,uint256) (runs: 1024, : 405928, ~:
↪  405860)
[PASS] testFuzz_DistributeDoesNotLeaveDust(uint256) (runs: 1024, : 1877140, ~: 1877329)
[PASS] testFuzz_DistributeDoesNotLeaveDust(uint256,uint256,uint256,uint256,uint256) (runs: 1024, : 1934959, ~: 1939411)
[PASS] testFuzz_Exit(uint256,uint256) (runs: 1024, : 979690, ~: 948016)
[PASS] testFuzz_GetRewardToRecipient(address) (runs: 1024, : 886536, ~: 886536)
[PASS] testFuzz_NotifyRewardsFailsIfRewardInsolvent(uint256,uint256) (runs: 1024, : 220467, ~: 220544)
[PASS] testFuzz_PartialWithdraw(address,uint256,uint256) (runs: 1024, : 326968, ~: 326905)
[PASS] testFuzz_ProcessIncentives(uint256) (runs: 1024, : 806533, ~: 844729)
[PASS] testFuzz_RemoveIncentiveToken(address) (runs: 1024, : 134379, ~: 134365)
[PASS] testFuzz_RemoveIncentiveToken_Multiple(uint8,uint256) (runs: 1024, : 3800703, ~: 3341618)
[PASS] testFuzz_SetRewardDuration(uint256) (runs: 1024, : 38136, ~: 38132)
[PASS] testFuzz_SetRewardsDurationDuringCycleEarned(uint256,uint256) (runs: 1024, : 705298, ~: 705345)
[PASS] testFuzz_Stake(address,uint256) (runs: 1024, : 295238, ~: 295393)
```

```
[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, : 35641, ~: 35911)
[PASS] testFuzz_Withdraw(address,uint256) (runs: 1024, : 326897, ~: 326979)
[PASS] testFuzz_Withdraw_FailsIfInsufficientSelfStake(uint256,uint256,uint256) (runs: 1024, : 409180, ~: 409157)
[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 115792089237316195423570985008687907853269084665640564039457584007913129639935

[PASS] test_ChangeInFactoryOwner() (gas: 67734)
[PASS] test_DelegateStake() (gas: 220062)
[PASS] test_DelegateStakeFailsIfPused() (gas: 77735)
[PASS] test_DelegateStakeWithSelfStake() (gas: 481472)
[PASS] test_DelegateWithdraw() (gas: 233742)
[PASS] test_DelegateWithdrawFailsIfNotDelegate() (gas: 33388)
[PASS] test_DelegateWithdrawFailsIfNotEnoughStakedByDelegate() (gas: 215760)
[PASS] test_Distribute() (gas: 502087)
[PASS] test_DistributeAndActivateQueuedRewardAllocation() (gas: 590211)
[PASS] test_DistributeForNonReentrant() (gas: 3469304)
[PASS] test_DistributeMulticall() (gas: 658307)
[PASS] test_DoNotUpdateIncentiveRateWhenAmountIsInsufficient() (gas: 563189)
[PASS] test_Exit() (gas: 1019284)
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, : 830233, ~: 832963)
[PASS] test_GetRewardNotOperatorOrUser() (gas: 40188)
[PASS] test_GetRewardWithDelegateStake() (gas: 844665)
[PASS] test_GetReward_AfterDuration(uint256,uint256) (runs: 1024, : 663473, ~: 663336)
[PASS] test_GetReward_BeforeDuration(uint256,uint256) (runs: 1024, : 663337, ~: 663345)
[PASS] test_GetReward_Notified_Twice(uint256,uint256) (runs: 1024, : 755072, ~: 754912)
[PASS] test_GetRewards() (gas: 825597)
[PASS] test_GetWhitelistedTokens() (gas: 352356)
[PASS] test_IncreaseIncentiveRate() (gas: 563228)
[PASS] test_InitialState() (gas: 33029)
[PASS] test_IsTimestampActionable_OutOfBuffer() (gas: 19355)
[PASS] test_IsTimestampActionable_Processing() (gas: 488571)
[PASS] test_NotifyRewardsDoesNotSetRewardRate() (gas: 418601)
[PASS] test_NotifyRewardsFailsIfRewardInsolvent() (gas: 220286)
[PASS] test_NotifyRewardsSetRewardRate() (gas: 741124)
[PASS] test_OperatorWorks() (gas: 913922)
[PASS] test_OwnerIsGovernance() (gas: 24206)
[PASS] test_PanprogL1() (gas: 593654)
[PASS] test_Pause() (gas: 51659)
[PASS] test_Pause_FailIfNotVaultManager() (gas: 25152)
[PASS] test_ProcessIncentives() (gas: 685883)
[PASS] test_ProcessIncentives_NotFailWithMaliciusIncentive() (gas: 991150)
[PASS] test_ProcessTimestamp_OutOfBuffer() (gas: 173901)
[PASS] test_ProcessTimestamp_Processing() (gas: 594374)
[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: 1126785)
[PASS] test_SelfStake() (gas: 304159)
[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: 704899)
[PASS] test_SetRewardDuration_FailIfNotOwner() (gas: 25221)
[PASS] test_SetRewardDuration_FailsIfZero() (gas: 27354)
[PASS] test_SetRewardRateAfterFirstStake() (gas: 579444)
[PASS] test_SetRewardsDurationDuringCycle() (gas: 713263)
[PASS] test_SetRewardsDurationDuringCycleMultipleUsers() (gas: 934526)
[PASS] test_Stake() (gas: 295351)
[PASS] test_StakeFailsIfPaused() (gas: 75568)
[PASS] test_Stake_FailsIfOverflow(address) (runs: 1024, : 302939, ~: 303080)
[PASS] test_Stake_FailsIfZeroAmount() (gas: 35768)
[PASS] test_TotalSupply() (gas: 509417)
[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: 2096606)
[PASS] test_WhitelistIncentiveToken() (gas: 269421)
[PASS] test_WhitelistIncentiveToken_FailsIfAlreadyWhitelisted() (gas: 274936)
[PASS] test_WhitelistIncentiveToken_FailsIfCountEqualToMax() (gas: 798913)
[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: 408974)
Logs:
  Bound result 1000000000000000000
  Bound result 1000000000000000000
  Bound result 1000000000000000000

[PASS] test_Withdraw_FailsIfZeroAmount() (gas: 38000)
[PASS] test_ZeroRewards() (gas: 296594)
Suite result: ok. 119 passed; 0 failed; 0 skipped; finished in 4.96s (14.00s CPU time)

Ran 2 tests for test/pol/e2e/POLE2EFuzz.t.sol:POLE2EFuzz
[PASS] testFuzzDistribution(uint96[],uint256,uint256) (runs: 1024, : 16360466, ~: 15900183)
[PASS] testGasPOLDistribution() (gas: 293999)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 7.60s (7.59s CPU time)

Ran 29 test suites in 7.60s (37.38s CPU time): 627 tests passed, 0 failed, 0 skipped (627 total tests)
```

8.2 Automated Tools

8.2.1 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.