PERIMETER

Berachain Proof of Liquidity #1

Defensive Fuzzing Report

Jan. 7, 2025

Draft

Prepared By:
0xScourgedev | Lead Fuzzing Specialist
0xscourgedev@perimetersec.io

Rappie | Lead Fuzzing Specialist
rappie@perimetersec.io

0xharold | Junior Fuzzing Specialist
0xharold@perimetersec.io

Table of Contents

Draft

About Perimeter

Perimeter's mission is to deliver the highest quality fuzzing services to protocols by uniting the world's foremost fuzzing specialists. We possess extensive expertise in fuzzing a diverse range of protocols, from smaller, niche protocols to some of the largest and most complex in DeFi.

In order to deliver on our mission, we have developed the most advanced scaffolding and libraries, enabling us to create highly sophisticated fuzzing suites tailored to meet the unique challenges of each protocol.

Learn more about us at perimetersec.io.

Draft

Risk Classification

The severity of security issues identified during the security review is classified according to the table below.

A. Critical findings are highly likely to be exploited with severe impact on the protocol and require immediate attention.

B. High findings are very likely to occur, easy to exploit, or difficult but highly incentivized, and should be resolved as quickly as possible.

C. Medium findings are possible in certain circumstances or when incentivized, with a moderate likelihood of occurring, and should be addressed.

D. Low findings involve rare circumstances to exploit or offer little to no incentives, though addressing them is still recommended.

E. Informational issues represent improvements that do not impact the project's overall security but are worth considering.

Severity Level	High Impact	Medium Impact	Low Impact
High Likelihood	Critical	High	Medium
Medium Likelihood	High	Medium	Low
Low Likelihood	Medium	Low	Low

Draft

Services Provided

Perimeter has successfully delivered a comprehensive suite of services that include:

- **Fuzzing Suite Development:** Design and implement a stateful fuzzing suite using Echidna and Medusa. This suite will be tailor-made for the protocol and contracts in scope. The completed fuzzing suite can later be integrated into the testing suite to serve long-term security needs.
- **Findings Reporting:** We provided thorough documentation and reporting of all findings identified throughout the engagement.
- **Invariant Testing Assurance:** Guarantee that each invariant implemented will be tested no fewer than 50,000,000 instances, ensuring thorough validation and reliability.
- **Proof-of-Concept Development:** Develop a corresponding Proof-of-Concept (PoC) for each finding and assertion/property counterexample identified, to demonstrate potential vulnerabilities and their implications.
- **Comprehensive Final Report:** Create a detailed final report that will include all findings, along with their corresponding PoCs. This report will also detail the invariants tested, their run status, and the number of runs, providing a comprehensive overview of the engagement's outcomes.

Draft

Files in Scope

The engagement will be focused on the files listed below, acquired from commit b60450ec5129ecfc12ddf4cb6012dad056ceb7c3.

```
src
├── base
│   ├── FactoryOwnable.sol
│   ├── IStakingRewardsErrors.sol
│   ├── IStakingRewards.sol
│   └── StakingRewards.sol
├── libraries
│   └── Utils.sol
├── pol
│   ├── BGT.sol
│   ├── BGTStaker.sol
│   ├── FeeCollector.sol
│   ├── interfaces
│   │   └── *.sol (11 files)
│   ├── rewards
│   │   ├── BerachainRewardsVaultFactory.sol
│   │   ├── BerachainRewardsVault.sol
│   │   ├── BeraChef.sol
│   │   ├── BlockRewardController.sol
│   │   └── Distributor.sol
│   └── RootHelper.sol
└── WBERA.sol
```

Files Out of Scope

Files outside the scope were not directly considered in achieving the target. However, since many of these files are utilized by those within the scope, a significant portion was indirectly covered.

Methodology

The primary goal of this engagement was to transform Rappie's offensive fuzzing suite into a defensive fuzzing suite. This involved migrating invariants into a sophisticated scaffolding framework, making them easier to work with, directly integrable into the test suite, and extensible for future development.

To thoroughly verify the correctness of the protocol, a large number of invariants were identified and implemented, and the coverage was significantly increased to include the following files:

- `BeraChef.sol`
- `BGTStaker.sol`
- `FeeCollector.sol`

In addition, support was added for multiple validators, tokens with varying decimals, and multiple instances of `BerachainRewardsVault`. The main threats under investigation were potential denial-of-service vulnerabilities in reward distribution, rounding issues in the `BerachainRewardsVault`, and other unexpected behavior.

Optimization mode with Medusa was used to determine the bounds of reward distribution and the maximum gas usage of the `distributeFor` function. However, due to planned refactors of the Proof of Liquidity codebase, these results became less relevant to the Berachain team. As a result, developments for them were discontinued and thus are considered preliminary rough estimates rather than precise bounds.

The time initially allocated for these optimizations was instead redirected toward creating additional invariants.

Draft

Invariants

We created many tests to verify the correctness of **142** invariants described in the table below. During the execution phase, these invariants were assessed for a total of **1,210,000,000+** calls.

The table below lists all invariants that are part of this engagement.

Invariant	Description	Tested	Passed	# Runs
STAKE-01	Staking for a reward vault increases stake balance for the caller by staked amount	✅	✅	1.21B+
STAKE-02	Staking for a reward vault decreases token balance for the caller by staked amount	✅	✅	1.21B+
STAKE-03	Staking for a reward vault does not change the amount of earned rewards	✅	✅	1.21B+
STAKE-04	Staking increases reward vault's stake token balance by staked amount	✅	✅	1.21B+
STAKE-05	Total supply of staked token in reward vault increases by staked amount	✅	✅	1.21B+
STAKE-06	Staking does not unexpectedly revert	✅	✅	1.21B+
STAKE-07	getReward for BGTStaker never reverts	✅	✅	1.21B+
DSTAKE-01	Delegate staking increases delegatee's total delegate staked by inputted amount	✅	✅	1.12B+
DSTAKE-02	Delegate staking decreases actor's token balance by inputted amount	✅	✅	1.12B+
DSTAKE-03	Delegate staking increases delegatee's stake balance by inputted amount	✅	✅	1.12B+
DSTAKE-04	Delegate staking increases reward vault's balance by inputted amount	✅	✅	1.12B+
DSTAKE-05	Total supply of staked token in reward vault increases by inputted amount	✅	✅	1.12B+
DSTAKE-06	Delegate staking increases delegatee's stakedByDelegate for a given actor by inputted amount	✅	✅	1.12B+

Invariant	Description	Tested	Passed	# Runs
DSTAKE-07	delegateStake does not unexpectedly revert	✅	✅	1.21B+
WDRAW-01	Withdrawing increases token balance by withdrawn amount	✅	✅	1.21B+
WDRAW-02	Withdrawing decreases stake balance by withdrawn amount	✅	✅	1.21B+
WDRAW-03	Withdrawing does not change the amount of earned rewards	✅	✅	1.21B+
WDRAW-04	Withdrawing decreases reward vault's balance by staked amount	✅	✅	1.21B+
WDRAW-05	Total supply of staked token in reward vault decreases by staked amount	✅	✅	1.21B+
WDRAW-06	Withdrawing does not unexpectedly revert	✅	✅	1.21B+
DWDRAW-01	Delegate withdrawal decreases delegatee's total delegate staked by inputted amount	✅	✅	1.12B+
DWDRAW-02	Delegate withdrawal increases actor's token balance by inputted amount	✅	✅	1.12B+
DWDRAW-03	Delegate withdrawal decreases delegatee's stake balance by inputted amount	✅	✅	1.12B+
DWDRAW-04	Delegate withdrawal decreases reward vault's balance by inputted amount	✅	✅	1.12B+
DWDRAW-05	Total supply of staked token in reward vault decreases by inputted amount	✅	✅	1.12B+
DWDRAW-06	Delegate withdrawal decreases delegatee's stakedByDelegate for a given actor by inputted amount	✅	✅	1.12B+
DWDRAW-07	Delegate withdrawal does not unexpectedly revert	✅	✅	1.21B+
EXIT-01	After exiting, the user's vault balance is equal to the user's delegateTotalStaked	✅	✅	1.21B+

Invariant	Description	Tested	Passed	# Runs
EXIT-02	Exiting sets the user's earned to zero	✅	✅	1.21B+
EXIT-03	After each user exits a vault, the sum of each user's delegateTotalStaked is always equal to the total supply of the reward vault	✅	✅	1.21B+
EXIT-04	Exiting increases actor's token balance by the difference between actor's stakeBalance and total delegate staked	✅	✅	1.12B+
EXIT-05	Exiting decreases stake balance by the difference between actor's stakeBalance and total delegate staked	✅	✅	1.12B+
EXIT-06	Exiting decreases reward vault's balance by the difference between actor's stakeBalance and total delegate staked	✅	✅	1.12B+
EXIT-07	After exiting, total supply of staked token in reward vault decreases by the difference between actor's stakeBalance and total delegate staked	✅	✅	1.12B+
EXIT-08	Exiting does not unexpectedly revert	✅	✅	1.21B+
RWD-01	Getting reward for a reward vault increases bgt balance of recipient by earned amount of selected account	✅	✅	1.21B+
RWD-02	Getting reward for a reward vault sets earned of the selected account to zero	✅	✅	1.21B+
RWD-12	Setting rewards duration for BGTStaker does not unexpectedly revert	✅	✅	1.21B+
RWD-13	Getting reward for a reward vault does not unexpectedly revert	✅	✅	1.21B+
INCENT-01	After a successful addIncentive call, the remaining amount of incentive token increases by the inputted amount	✅	✅	1.21B+
INCENT-02	After a successful addIncentive call, the minimum incentive rate of the incentive token does not change	✅	✅	1.21B+

Invariant	Description	Tested	Passed	# Runs
INCENT-03	After a successful addIncentive call, the incentive rate of the incentive token is less than or equal to the incentive rate ceiling	✅	✅	1.21B+
INCENT-05	addIncentive does not unexpectedly revert	✅	✅	1.21B+
INCENT-06	removeIncentiveToken does not unexpectedly revert	✅	✅	1.21B+
INCENT-07	setMaxIncentiveTokensCount does not unexpectedly revert	✅	✅	1.21B+
INCENT-08	Whitelisting incentive token does not unexpectedly revert	✅	✅	1.21B+
DIST-01	After a successful distributeFor call, the stored reward per token increases or remains the same for all vaults	✅	✅	1.12B+
DIST-02	After a successful distributeFor call, the total supply for all vaults remain the same	✅	✅	1.12B+
DIST-03	After a successful distributeFor call, if the MIN_BASE_RATE or the MIN_REWARD_RATE are greater than 0, then BGT total supply strictly increases	✅	✅	1.12B+
DIST-04	After a successful distributeFor call, the total earned rewards for each vault remains the same	✅	✅	1.12B+
DIST-05	Distributing rewards never reverts	✅	✅	1.21B+
BOOST-01	User's BGT balance does not change after queuing boost	✅	✅	1.21B+
BOOST-02	User's unboosted BGT decreases by queued amount after queuing boost	✅	✅	1.21B+
BOOST-03	User's queued BGT increases by queued amount after queuing boost	✅	✅	1.21B+
BOOST-04	User's boosted BGT does not change after queuing boost	✅	✅	1.21B+

Invariant	Description	Tested	Passed	# Runs
BOOST-05	Queuing boost does not unexpectedly revert	✅	✅	1.21B+
BOOST-06	Validator's boostees does not change after queuing boost	✅	✅	1.21B+
BOOST-07	User's BGT balance in BGTStaker does not change after queuing boost	✅	✅	1.21B+
BOOST-08	User's boostedQueue on a validator increases by amount after queuing boost	✅	✅	1.21B+
BOOST-09	User's dropBoostQueue on a validator does not change after queuing boost	✅	✅	1.21B+
BOOST-10	User's dropBoostQueue on a validator does not change after successfully activating boost	✅	✅	1.21B+
BOOST-11	User's BGT balance does not change after successfully activating boost	✅	✅	1.21B+
BOOST-12	User's unboosted BGT does not change after successfully activating boost	✅	✅	1.21B+
BOOST-13	User's queued BGT decreases after successfully activating boost	✅	✅	1.21B+
BOOST-14	User's boosted BGT increases after successfully activating boost	✅	✅	1.21B+
BOOST-15	Activating boost does not unexpectedly revert	✅	✅	1.21B+
BOOST-16	Validator's boostees increases after successfully activating boost	✅	✅	1.21B+
BOOST-17	User's BGT balance in BGTStaker increases after successfully activating boost	✅	✅	1.21B+
BOOST-18	User's boostedQueue on a validator is 0 after successfully activating boost	✅	✅	1.21B+
BOOST-19	User's BGT balance in BGTStaker increases after successfully activating boost	✅	✅	1.21B+
BOOST-20	User's dropBoostQueue on a validator increases by amount after queuing drop boost	✅	✅	1.21B+

Invariant	Description	Tested	Passed	# Runs
BOOST-21	User's BGT balance does not change after queuing drop boost	✅	✅	1.21B+
BOOST-22	User's unboosted BGT does not change after queuing drop boost	✅	✅	1.21B+
BOOST-23	User's queued BGT does not change after queuing drop boost	✅	✅	1.21B+
BOOST-24	User's boosted BGT does not change after queuing drop boost	✅	✅	1.21B+
BOOST-25	Queuing drop boost does not unexpectedly revert	✅	✅	1.21B+
BOOST-26	Validator's boostees does not change after queuing drop boost	✅	✅	1.21B+
BOOST-27	User's BGT balance in BGTStaker does not change after queuing drop boost	✅	✅	1.21B+
BOOST-28	Unboosted BGT balance of the user increases by amount after cancelling boost	✅	✅	1.21B+
BOOST-29	User's boostedQueue on a validator decreases by amount after cancelling boost	✅	✅	1.21B+
BOOST-30	User's queued BGT decreases by amount after cancelling boost	✅	✅	1.21B+
BOOST-31	User's BGT balance does not change after successfully dropping boost	✅	✅	1.21B+
BOOST-32	User's unboosted BGT increases after successfully dropping boost	✅	✅	1.21B+
BOOST-33	User's queued BGT does not change after successfully dropping boost	✅	✅	1.21B+
BOOST-34	User's boosted BGT decreases after successfully dropping boost	✅	✅	1.21B+
BOOST-35	Dropping boost does not unexpectedly revert	✅	✅	1.21B+

Invariant	Description	Tested	Passed	# Runs
BOOST-36	Validator's boostees decreases after successfully dropping boost	✅	✅	1.21B+
BOOST-37	User's BGT balance in BGTStaker decreases after successfully dropping boost	✅	✅	1.21B+
BOOST-38	User's boostedQueue on a validator does not change after successfully dropping boost	✅	✅	1.21B+
BOOST-39	User's dropBoostQueue on a validator is 0 after successfully dropping boost	✅	✅	1.21B+
BOOST-40	Cancelling boost does not unexpectedly revert	✅	✅	1.21B+
BOOST-41	User's dropBoostQueue on a validator decreases by amount after cancelling drop boost	✅	✅	1.21B+
BOOST-42	Cancelling drop boost does not unexpectedly revert	✅	✅	1.21B+
BOOST-43	totalBoosts increases by exactly the queued boosts of for the user of the selected pubkey after a successful activateBoost call	✅	✅	1.21B+
BOOST-44	totalBoosts decreases by exactly the queued drop boosts of for the user of the selected pubkey after a successful dropBoost call	✅	✅	1.21B+
BOOST-45	totalBoosts is equal to the sum of all users' boosts	✅	✅	1.21B+
BOOST-46	The unboosted BGT balance of a user is always less than or equal to the user's BGT balance	✅	✅	1.12B+
BOOST-47	The queued BGT balance of a user is always less than or equal to the user's BGT balance	✅	✅	1.12B+
BOOST-48	The boosted BGT balance of a user is always less than or equal to the user's BGT balance	✅	✅	1.12B+
BOOST-49	The sum of the unboosted, queued, and boosted BGT balances of a user is always equal to the user's BGT balance	✅	✅	1.12B+

Invariant	Description	Tested	Passed	# Runs
BRCREV-01	setBaseRate does not unexpectedly revert	✅	✅	1.21B+
BRCREV-02	setBoostMultiplier does not unexpectedly revert	✅	✅	1.21B+
BRCREV-03	setMinBoostedRewardRate does not unexpectedly revert	✅	✅	1.21B+
BRCREV-04	setRewardConvexity does not unexpectedly revert	✅	✅	1.21B+
BRCREV-05	setRewardRate does not unexpectedly revert	✅	✅	1.21B+
CHFREV-01	queueNewRewardAllocation does not unexpectedly revert	✅	✅	1.21B+
CHFREV-02	setRewardAllocationBlockDelay does not unexpectedly revert	✅	✅	1.21B+
CHFREV-03	setDefaultRewardAllocation does not unexpectedly revert	✅	✅	1.21B+
CHFREV-04	setMaxNumWeightsPerRewardAllocation does not unexpectedly revert	✅	✅	1.21B+
CHFREV-05	setVaultWhitelistedStatus does not unexpectedly revert	✅	✅	1.21B+
CHFREV-06	If the length of the weights passed to queueNewRewardAllocation is 0, then it should always revert	✅	✅	1.21B+
VLTREV-01	setOperator does not unexpectedly revert	✅	✅	1.21B+
VLTREV-02	setRewardsDuration does not unexpectedly revert	✅	✅	1.21B+
COMMIS-01	Current commission rate does not change when the new commission rate is queued	✅	✅	1.21B+
COMMIS-02	Queued commission rate is equal to the new commission rate proposed	✅	✅	1.21B+
COMMIS-03	Queued commission rate is 0 after cancelling the queued commission rate	✅	✅	1.21B+

Invariant	Description	Tested	Passed	# Runs
COMMIS-04	Current commission rate gets updated to the queued commission rate	✅	✅	1.21B+
COMMIS-05	Queued commission rate is 0 after activating the queued commission rate	✅	✅	1.21B+
COMMIS-06	Current commission rate does not change when the queued commission rate is cancelled	✅	✅	1.21B+
COMMIS-07	Activating commission does not unexpectedly revert	✅	✅	1.21B+
COMMIS-08	Cancelling commission change does not unexpectedly revert	✅	✅	1.21B+
COMMIS-09	Queueing commission rate does not unexpectedly revert	✅	✅	1.21B+
FEE-01	queuePayoutAmountChange for FeeCollector never reverts	✅	✅	1.21B+
FEE-02	The WBERA balance of the BGTStaker contract increases by exactly the payout amount after claiming fees	✅	✅	1.21B+
FEE-03	The WBERA balance of the BGTStaker contract increases by exactly the amount after donating	✅	✅	1.21B+
FEE-04	The WBERA balance of the user decreases by exactly the amount after donating	✅	✅	1.21B+
FEE-05	The WBERA balance of the user decreases by exactly the payout amount after claiming fees	✅	✅	1.21B+
FEE-06	The feeToken balance of the FeeCollector contract is zero after claiming fees	✅	✅	1.21B+
FEE-07	After claiming fees, the feeToken balance of the recipient increases by exactly the balance of the FeeCollector had before	✅	✅	1.21B+
FEE-08	claimFees for FeeCollector never reverts	✅	✅	1.21B+

Invariant	Description	Tested	Passed	# Runs
FEE-09	donate for FeeCollector never reverts	✅	✅	1.21B+
SPLY-01	totalSupply of BGTStaker is equal to the sum of all user's balances	✅	✅	1.21B+
SPLY-02	totalSupply of reward vaults is equal to the sum of all user's balances	✅	✅	1.21B+
SPLY-03	totalSupply of reward vaults is always greater or equal to the sum of each user's delegateTotalStaked	✅	✅	1.21B+
SPLY-04	The native token balance of the BGT contract is always greater or equal to the total supply of BGT	✅	✅	1.21B+
SPLY-05	The total supply of BGT is always equal to the sum of all user's BGT balances plus the BGT balance of the distributor	✅	✅	1.21B+
SPLY-06	The total supply of BGT is always equal to the sum of all users' unboosted balances, all users' queued balances, all users' boosted balances, and the BGT balance of the distributor	✅	✅	1.21B+
REDEEM-01	Sender's BGT balance decreases by redeemed amount after redeeming	✅	✅	1.21B+
REDEEM-02	Receiver's BERA balance increases by redeemed amount after redeeming	✅	✅	1.21B+
REDEEM-03	If the sender is not receiver, sender's BERA balance does not change after redeeming	✅	✅	1.21B+
REDEEM-04	If the receiver is not sender, receiver's BGT balance does not change after redeeming	✅	✅	1.21B+
REDEEM-05	Redeeming of BGT does not unexpectedly revert	✅	✅	1.21B+

Optimizations

Medusa optimization mode was run for over **143,000,000+** iterations to determine the bounds for BGT emissions and maximum gas consumed by the `distributeFor` function with different numbers of max weights.

Invariant	# Runs	Optimized Value
The minimum amount of BGT distributed for a single distribution	143M+	1
The maximum amount of BGT distributed for a single distribution	143M+	~110e18
The maximum amount of gas consumed by the `distributeFor` function with max weights set to **5**	143M+	1,087,757
The maximum amount of gas consumed by the `distributeFor` function with max weights set to **10**	143M+	1,531,623
The maximum amount of gas consumed by the `distributeFor` function with max weights set to **11**	143M+	1,528,768
The maximum amount of gas consumed by the `distributeFor` function with max weights set to **12**	143M+	1,530,229
The maximum amount of gas consumed by the `distributeFor` function with max weights set to **13**	143M+	1,624,446
The maximum amount of gas consumed by the `distributeFor` function with max weights set to **14**	143M+	1,742,765
The maximum amount of gas consumed by the `distributeFor` function with max weights set to 1**5**	143M+	1,708,447

Disclaimer

All activities conducted by Perimeter in connection with this project were carried out in accordance with the terms outlined in a Statement of Work and an agreed-upon project plan, as set forth in a proposal document delivered prior to the commencement of the project.

Security assessment projects are subject to time limitations, and as such, the findings presented in this report should not be interpreted as an exhaustive or comprehensive identification of all security issues, vulnerabilities, or defects within the target codebase. Perimeter makes no representations or warranties that the target codebase is free from defects.

Furthermore, this report is not intended to be, and should not be construed as, investment advice or a recommendation to participate in any financial transactions. The content herein does not constitute endorsements or recommendations for any financial decisions, securities, or investment strategies.

Draft