our shielding · Your smart contracts, our shielding · Your smart c

shieldify

BeraRoot

SECURITY REVIEW

Date: 3 September 2025

CONTENTS

1. About Shieldify

Positioned as the first hybrid Web3 Security company, Shieldify shakes things up with a unique subscription-based auditing model that entitles the customer to unlimited audits within its duration, as well as top-notch service quality thanks to a disruptive 6-layered security approach. The company works with very well-established researchers in the space and has secured multiple millions in TVL across protocols, also can audit codebases written in Solidity, Rust, Go, Vyper, Move and Cairo.

Learn more about us at shieldify.org.

2. Disclaimer

This security review does not guarantee bulletproof protection against a hack or exploit. Smart contracts are a novel technological feat with many known and unknown risks. The protocol, which this report is intended for, indemnifies Shieldify Security against any responsibility for any misbehavior, bugs, or exploits affecting the audited code during any part of the project's life cycle. It is also pivotal to acknowledge that modifications made to the audited code, including fixes for the issues described in this report, may introduce new problems and necessitate additional auditing.

3. About BeraRoot

Beraroot is a cutting-edge Data Availability (DA) solution tailored for the Proof-of-Liquidity (PoL) consensus mechanism, empowering on-chain traders and fostering a thriving ecosystem within Berachain's infrastructure. By merging Miner Extractable Value (MEV) optimization with DA technologies, we ensure seamless transactions, fortified security, and unparalleled transparency for both Berachain and its Layer2 platforms.

Our innovation lies in addressing the scalability hurdles posed by heightened on-chain activities under PoL, leveraging DAS and KZG Commitments to enhance real-time data broadcast and persistent storage. We minimize transaction costs through gas optimization and introduce a fair MEV algorithm, thereby cultivating an equitable trading environment resistant to unfair practices.

4. Risk Classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

4.1 Impact

- **High** – results in a significant risk for the protocol's overall well-being. Affects all or most users
- **Medium** – results in a non-critical risk for the protocol affects all or only a subset of users, but is still unacceptable
- **Low** – losses will be limited but bearable – and covers vectors similar to griefing attacks that can be easily repaired

4.2 Likelihood

- **High** – almost certain to happen and highly lucrative for execution by malicious actors
- **Medium** – still relatively likely, although only conditionally possible
- **Low** – requires a unique set of circumstances and poses non-lucrative cost-of-execution to rewards ratio for the actor

5. Security Review Summary

The security review lasted 5 days with a total of 80 hours dedicated by 2 researchers from the Shieldify team.

Overall, the code is well-written. The audit report identified one Medium and one Low severity issue. They primarily stem from potential reward loss during periods when zero participants are staking.

The BeraRoot team has done a great job with the development and has been highly responsive to the Shieldify research team's inquiries. The current contract is deprecated, and its issues will be addressed in the next version.

5.1 Protocol Summary

Project Name	**BeraRoot**
Repository	bluebera-contracts
Type of Project	DeFi, Staking
Audit Timeline	5 days
Review Commit Hash	5e2c5f14717718d7b4dc32729853f0c945ec3bb8

5.2 Scope

The following smart contracts were in the scope of the security review:

File	**nSLOC**
contracts/BlueBeraToken.sol	140
contracts/LinearVesting.sol	56
contracts/AirdropDistributor.sol	35
contracts/StakingRewards.sol	91
Total	**322**

6. Findings Summary

The following number of issues have been identified, sorted by their severity:

- **Medium** issues: 1
- **Low** issues: 1

ID	Title	Severity	Status
[M-01]	The Protocol Can Permanently Lose Rewards During Periods When Zero Participants Are Staking	Medium	Acknowledged
[L-01]	Use `Ownable2Step` Version Rather Than `Ownable` Version	Low	Acknowledged

7. Findings

[M-01] Protocol Can Permanently Lose Rewards During Periods When Zero Participants Are Staking

Severity

Medium Risk

Description

The vulnerability occurs in the reward calculation and distribution mechanism when the staking pool experiences periods with zero total supply (`_totalSupply == 0`). During these periods, the `rewardPerToken()` function returns `rewardPerTokenStored` without accounting for the time elapsed, effectively causing rewards to accumulate in a state where they cannot be claimed by any participant.

When `_totalSupply` is zero, the mathematical formula `((lastTimeRewardApplicable() - lastUpdateTime) * rewardRate * 1e18) / _totalSupply` cannot be computed, so the function returns the stored value. However, the `rewardRate` continues to represent active reward distribution, and time continues to pass within the reward period defined by `periodFinish`. This creates a scenario where rewards are theoretically being distributed but cannot be claimed because there are no stakers to receive them.

Location of Affected Code

File: StakingRewards.sol

```
function rewardPerToken() public view returns (uint256) {
    if (_totalSupply == 0) return rewardPerTokenStored;
    return rewardPerTokenStored + ((lastTimeRewardApplicable() -
       lastUpdateTime) * rewardRate * 1e18) / _totalSupply;
}
```

Impact

The primary impact is permanent loss of protocol funds allocated for reward distribution. When reward periods experience extended periods without any stakers, the corresponding portion of rewards becomes permanently inaccessible, representing a direct financial loss to the protocol or reward provider.

Recommendation

Consider implementing a reward reclaim mechanism that allows the contract owner to recover unused rewards from periods with zero participation, enabling reallocation to future reward periods or alternative distribution mechanisms.

Team Response

Acknowledged.

[L-01] Use `Ownable2Step` Version Rather Than `Ownable` Version

Severity

Low Risk

Description

The `Ownable2Step` prevents the contract ownership from mistakenly being transferred to an address that cannot handle it (e.g. due to a typo in the address), by requiring that the recipient of the owner's permissions actively accept via a contract call of its own.

Consider using `Ownable2Step` from OpenZeppelin Contracts to enhance the security of your contract ownership management. This contract prevents the accidental transfer of ownership to an address that cannot handle it, such as due to a typo, by requiring the recipient of owner permissions to actively accept ownership via a contract call.

Location of Affected Code

File: BlueBeraToken.sol
File: LinearVesting.sol
File: AirdropDistributor.sol
File: StakingRewards.sol

Recommendation

Consider using `Ownable2Step` from OpenZeppelin Contracts

Team Response

Acknowledged.

shieldify

Thank you!