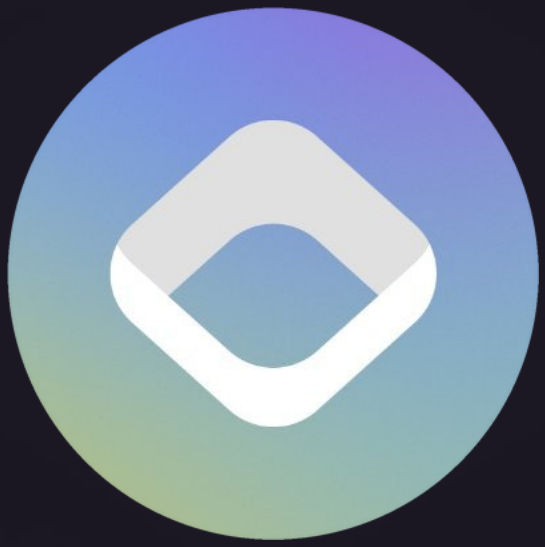
SHERLOCK

Security Review For RootsFi



Collaborative Audit Prepared For: **RootsFi**
Lead Security Expert(s): **eeyore**
KupiaSec

Date Audited: **May 1 - May 3, 2025**
Final Commit: **c3c587c**

Introduction

This review focuses on the validator boosting for users and the issuance of the non-soulbound token rBGT

Scope

Repository: roots-fi/roots-core

Audited Commit: ff764aa7a63d4158535a859616c0e9eea755c547

Final Commit: c3c587c681fff39c2a021a7d37fee67cf7ca0d79

Files:

- src/core/BGTHandler.sol
- src/core/RootsBGT.sol
- src/core/Staker.sol

Final Commit Hash

c3c587c681fff39c2a021a7d37fee67cf7ca0d79

Findings

Each issue has an assigned severity:

- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- High issues are directly exploitable security vulnerabilities that need to be fixed.
- Low/Info issues are non-exploitable, informational findings that do not pose a security risk or impact the system's integrity. These issues are typically cosmetic or related to compliance requirements, and are not considered a priority for remediation.

Issues Found

High	Medium	Low/Info
0	0	8

Issues Not Fixed and Not Acknowledged

High	Medium	Low/Info
0	0	0

Issue L-1: Missing bgtHandler validation when setting gauge

Source: https://github.com/sherlock-audit/2025-04-rootsfi/issues/5

This issue has been acknowledged by the team but won't be fixed at this time.

Summary

The `bgtHandler` must not be left unset (`address(0)`) when a valid `gauge` address is assigned in `Staker` contract.

If the `gauge` is configured without setting the `bgtHandler`, the `_fetchRewards()` function will revert during execution due to the missing handler address.

This issue also applies to the `setGauge()` function, which does not currently validate that a non-zero `bgtHandler` is in place before allowing a gauge to be set.

Recommendation

In both the `initialize()` and `setGauge()` functions, include a validation check to ensure that `bgtHandler` is not set to `address(0)` when assigning a non-zero `gauge` address.

Issue L-2: BGTHandler implementation contract can be initialized

Source: https://github.com/sherlock-audit/2025-04-rootsfi/issues/6

This issue has been acknowledged by the team but won't be fixed at this time.

Summary

The `BGTHandler` contract is an UUPSUpgradeable contract intended to be used through a proxy. Since implementation contracts should not be used directly, it is standard practice to disallow their initialization.

Recommendation

Consider adding a constructor that calls `_disableInitializers()` in `BGTHandler` to prevent the contract from being initialized outside of the proxy context.

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

Issue L-3: Missing check for IGauge(_newGauge).paused() before stake() call

Source: https://github.com/sherlock-audit/2025-04-rootsfi/issues/7

Summary

In the `setGauge` function, the contract calls `IGauge(_newGauge).stake(assetBalance)` without checking if the new gauge is paused. As with the `onDeposit` function, if the gauge is paused, the assets should not be staked but should instead be added to the `stakeOnHold` variable. Otherwise, the function call will revert.

Recommendation

Before calling `stake()` on the new gauge, ensure that the gauge is not paused. If it is paused, the current balance should be added to the `stakeOnHold` variable instead of attempting to stake it.

Issue L-4: Redundant unlimited asset approval to _troveManager

Source: https://github.com/sherlock-audit/2025-04-rootsfi/issues/8

Summary

The `initialize` function in `Staker` contract includes an unlimited approval of the `asset` to the `_troveManager` address. However, the `TroveManager` contract does not contain any function that utilizes `transferFrom()` to move tokens from this contract. As a result, this approval appears to be redundant.

Recommendation

Consider removing the line:

```
IERC20(_asset).approve(_troveManager, type(uint256).max);
```

unless there is a concrete and verified use case for the `TroveManager` to call `transferFrom()` on the approved asset.

Issue L-5: Unused DECIMAL_PRECISION constant

Source: https://github.com/sherlock-audit/2025-04-rootsfi/issues/9

Summary

The `Staker` contract defines a constant `DECIMAL_PRECISION = 1e18`, which is not used anywhere in the code. Instead, multiple instances of the hardcoded value `1e18` are used directly in reward calculations. This practice reduces maintainability and introduces the risk of inconsistencies if the precision ever needs to be updated in the future.

Recommendation

Replace hardcoded instances of `1e18` with the `DECIMAL_PRECISION` constant throughout the contract.

Issue L-6: Unnecessary update of lastUpdate in the _fetchRewards() function

Source: https://github.com/sherlock-audit/2025-04-rootsfi/issues/10

Summary

In the `_fetchRewards()` function, the variable `lastUpdate` is set to the current `block.timestamp`. However, prior to invoking this function, `lastUpdate` has already been updated to `block.timestamp`. Therefore, it is unnecessary to update the variable again within the `_fetchRewards()` function.

Recommendation

Remove the instruction to update the variable `lastUpdate` in the `_fetchRewards()` function.

Issue L-7: Use safeTransfer() instead of transfer()

Source: https://github.com/sherlock-audit/2025-04-rootsfi/issues/11

This issue has been acknowledged by the team but won't be fixed at this time.

Summary

When transferring asset, the method `ERC20.transfer()` is currently used. To ensure compatibility with all tokens, it would be better to utilize the `safeTransfer` function.

Recommendation

Use the `SafeERC20.safeTransfer()` function instead of `transfer()` function.

Issue L-8: Stake all balance of the contract instead of stakeOnHold

Source: https://github.com/sherlock-audit/2025-04-rootsfi/issues/12

Summary

When the `staker` contract holds any amount of `asset` and the `gauge` is not paused, it stakes them to the gauge in the `_updateRewardIntegral()` function. The variable `stakeOnHold` attempts to track the balance of `asset` in the contract, but it could be different from the actual balance since `stakeOnHold` reflects the balance recorded in the `setGauge()` function.

Recommendation

Use `asset.balanceOf(address(this))` instead of `stakeOnHold` when staking `asset` in the `_updateRewardIntegral()` function.

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users' sole risk and is the users' responsibility.