The Accelerated Curve

Nirlin Solo Audit

Load Network Audit Report

April 21, 2025

This audit report presents a focused security assessment of the Load Network distribution smart contracts, highlighting potential vulnerabilities and areas for improvement.

Scope and Summary of Findings

Scope of Audit

Contract Files Audited
DaiBalanceChecker.sol
DaiDistribution.sol
DaiDistributionPlus.sol
Refunder.sol
StEthDistribution.sol
LinearDistributionIntervalDecrease.sol
DaiDistributionInterface.sol
DaiDistributionPlusInterface.sol
DaiDsrManagerInterface.sol
IDaiDistribution.sol
StEthDistributionInterface.sol

Summary of Findings

Title	Severity
Users Lose Rewards if Withdrawing After Distribution Period Ends	High

Audit Report: DaiDistributionPlus: Users Lose Rewards if Withdrawing After Distribution Period Ends

Summary

DaiDistributionPlus contracts for stETH and DAI users lose accumulated rewards if they withdraw after the distribution period ends and there was low staking and withdrawing activity before distribution period ends.

Description

The vulnerability stems from a premature check in the `getPeriodReward` function:

```
if (block.number - deploymentBlock > DISTRIBUTION_PERIOD_BLOCKS) { return 0; }
```

This check zeroes out reward calculations after the distribution period ends, causing users to permanently lose their earned rewards.

Proof of Concept

A test case demonstrates the issue:

```
function test_multipleUsersStakingAndRewards() public {
    // [Test setup code]

    // First staker stakes
    vm.startPrank(firstStaker);
    DAI.approve(address(distribution), tenDAI);
    distribution.stake(0, tenDAI, "0");
    vm.stopPrank();

    // Move time forward 2 weeks
    vm.warp(block.timestamp + 2 weeks);
    vm.roll(block.number + 100_000);

    // Three more users stake
    // [Users 1-3 staking code]

    // Move time to end of distribution period
    vm.roll(block.number + distribution.DISTRIBUTION_PERIOD_BLOCKS());
    vm.warp(block.timestamp + 40 weeks);

    // Users withdraw and check rewards
    // [Withdrawal code]
}
```

Expected Results with Vulnerable Code

```
First Staker:
  Initial Balance:  990 DAI
  Final Balance:    1000 DAI
  Rewards Earned:   48330 DAI
  Remaining Rewards:48330 DAI

User 1:
```

```
  Initial Balance:  0 DAI
  Final Balance:    10 DAI
  Rewards Earned:   0 DAI
  Remaining Rewards:0 DAI
```

Expected Results with Fix Applied

```
First Staker:
  Initial Balance:  990 DAI
  Final Balance:    1000 DAI
  Rewards Earned:   283882 DAI
  Remaining Rewards:283882 DAI

User 1:
  Initial Balance:  0 DAI
  Final Balance:    10 DAI
  Rewards Earned:   235552 DAI
  Remaining Rewards:235552 DAI
```

POC Test Code

```
function test_multipleUsersStakingAndRewards() public {
    // [Test setup code]

    // First staker stakes
    vm.startPrank(firstStaker);
    DAI.approve(address(distribution), tenDAI);
    distribution.stake(0, tenDAI, "0");
    vm.stopPrank();

    // Move time forward 2 weeks
    vm.warp(block.timestamp + 2 weeks);
    vm.roll(block.number + 100_000);

    // Three more users stake
    // [Users 1-3 staking code]

    // Move time to end of distribution period without any further stake or
    withdrawal
    vm.roll(block.number + distribution.DISTRIBUTION_PERIOD_BLOCKS());
    vm.warp(block.timestamp + 40 weeks);

    // Users withdraw and check rewards
    // [Withdrawal code]
}
```

Recommendation

Refactor the logic in `getPeriodReward` to calculate rewards **based on the user's stake and time during the distribution period**, even if the withdrawal happens later.

A correct fix might include:

- Storing the user's reward entitlement separately at the end of the distribution period.

- Decoupling reward accrual from withdrawal logic, so users can claim even after the period ends.
- Allowing post-period withdrawals to read from a finalized reward snapshot.

Avoid abruptly cutting off reward calculation with a single block check — instead, finalize reward states at period end and preserve them.