alethea.ai

Staking Smart Contracts

Audit Report

August 29, 2024

Hiroto Iwaki

Table of Contents

Overview

This document describes the audit results for smart contract of ERC1363StakingTrackerV1 and NFTStaking Contract in Alethea AI protocols.
The ERC1363StakingTrackerV1 contract allows users to lock and unlock ERC20 tokens for a specified duration. It supports multiple staking/deposit tokens and allows administrators to configure the lock-up duration for each token. It ensures that the tokens remain locked for the duration specified by the admin, with the option to disable the lock-up period altogether.

The NFTStakingV2 contract enables staking and unstaking of NFTs while tracking relevant staking data such as owner, stake time, unlock time, and unstake time. It includes administrative functions for setting a locking period, rescuing tokens, and managing staking features.

Project	Alethea AI
Repository	AI Protocol Contracts
Commit Hash	306132b27dca9353b0da638a118e1e1176059cf7
Network	Ethereum Mainnet, Base
Type of Project	Staking
Audit Period	August 20 to August 29

Risk Classification

High Assets can be stolen/lost/compromised directly (or indirectly if there is a valid attack path that does not have hand-wavy hypotheticals).

Medium Assets not at direct risk, but the function of the protocol or its availability could be impacted, or leak value with a hypothetical attack path with stated assumptions, but external requirements.

Low Assets are not at risk: state handling, function incorrect as to spec, issues with comments.

Info Code style, clarity, syntax, versioning, off-chain monitoring events, etc.

Gas Re-writing Solidity code to accomplish the same business logic while consuming fewer gas.

Summary of Findings

High	0
Medium	0
Low	1
Info	0
Gas	1
Total	2

Finding Details

Low

[L-01] Lack of Input Validation in deposit function

Severity

Low

Affected Lines

ERC1363StakingTrackerV1.sol# L190, L190

Finding Description

The function __createDeposit casts the deposit amount to uint160, and there is a check to ensure that the value does not exceed type(uint160).max. However, in a malicious situation where extremely large values are sent, there could be a silent overflow or unintended behavior.

```
function __createDeposit(address depositToken, address depositOwner, uint160
depositAmount) private  {
 // verify the inputs
```

Recommended Mitigation

Keep the checks in place to ensure the value does not exceed `type(uint160).max` and consider further testing for scenarios where edge cases might arise.

Gas Optimization

[G-01] Use `immutable` and `constant` for Fixed State Variables

Affected Lines

ERC20BondingCurveFactoryV1.sol#L70, L79

NFTFactoryV2.sol#L67, L74, L81, L89, L97

Descriptions

State variables like FEATURE_LOCKING_DISABLED and ROLE_DEPOSIT_SETTINGS_MANAGER are marked as constant, which is good practice.

```
@> uint32 public constant FEATURE_LOCKING_DISABLED = 0x0000_0001;

  /**
   * @notice Deposit settings manager is responsible for deposit settings setup
   *
   * @dev Role ROLE_DEPOSIT_SETTINGS_MANAGER allows
   *      updating global deposit settings via updateDepositSettings(), and
   *      deleting these settings via deleteDepositSettings()
   */
@> uint32 public constant ROLE_DEPOSIT_SETTINGS_MANAGER = 0x0002_0000;
```

Recommended Mitigation

Also, consider marking other state variables that do not change after initialization with `immutable` to save gas.

Conclusion

The ERC1363StakingTrackerV1.sol, NFTStakingV2 contracts are generally well-structured, with clear functionality for staking, locking and some other functionalities. Additionally, implementing gas optimizations and input checks will help improve the contract's overall security and efficiency.