 

Staking Smart Contracts

Audit Report

August 29, 2024

Mykyta Mesentsev

# Table of Contents

[Table of Contents 1](#_Toc175869883)

[Overview 2](#_Toc175869884)

[Risk Classification 3](#_Toc175869885)

[Summary of Findings 4](#_Toc175869886)

[Finding Details 5](#_Toc175869887)

[Low 5](#_Toc175869888)

[[L-01] Lack of Input Validation in deposit function 5](#_Toc175869889)

[Gas Optimization 6](#_Toc175869890)

[[G-01] Use immutable and constant for Fixed State Variables 6](#_Toc175869891)

[Conclusion 7](#_Toc175869892)

# 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](https://github.com/AI-Protocol-Official/ai-protocol-contracts) |
| Commit Hash | [306132b27dca9353b0da638a118e1e1176059cf7](https://github.com/AI-Protocol-Official/ai-protocol-contracts/pull/281/files) |
| 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.