gkrastenov

Smart Contract Security Review

LO-FI PEPE

June 21, 2024

gkrastenov.com gkrastenov gkrastenov

Table of Contents

1 About gkrastenov

Georgi Krastenov, known as gkrastenov, is an independent smart contract security researcher and former smart contract engineer at Nexo. Having conducted over 15 solo smart contract security reviews and discovered numerous vulnerabilities in various protocols, he does his best to contribute to the blockchain ecosystem and its protocols by dedicating time and effort to security research and reviews. Check his previous work here or reach out on Twitter/X or Telegram @gkrastenov.

2 Disclaimer

Audits are a time, resource, and expertise bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to identify as many vulnerabilities as possible. Audits can show the presence of vulnerabilities **but not their absence**.

3 About LO-FI

LO-FI PEPE is a meme NFT collection. The creator of the collection has created a staking contract where every owner of an NFT from this collection can stake their NFT and participate in rewarding. As a reward, the staker can claim PEPE tokens anytime they want.

4 Description

The report describes all changes that were made during development and gas optimizations. New functionality for unstaking all NFTs is added. Additionally, a new requirement for stakers to have at least 10 NFTs is implemented. The staking contract was tested twice on Sepoila Testnet and Mainnet.

5 Executive summary

Overview

Project Name	LO-FI PEPE NFT
Repository	https://github.com/0xVins/lofipepestaking
Commit hash	4f04d22bc3eab218ed4dbbf3579291cba26a07e5
Review Commit hash	3c517eb67fdf8c4910fafcf62a0b12b2b582b5b1
Documentation	https://lofipepe.com/
Methods	Manual review

Scope

contract/Staking.sol

Timeline

May 29, 2024	Audit kick-off
June 21, 2024	Preliminary report
June 21, 2024	Mitigation review

Issues Found

Severity	**Count**
High	0
Medium	0
Low	0
Development	4
Gas	6
Total	**10**

6 Findings & Changes

6.1 Development

6.1.1 Add the unStakeAll function

Severity: *Code Improvement*

Description: A new function unStakeAll is added. When the staking period has passed, the admin has the right to unstake all NFTs. Every staked NFT and the reward amount are transferred to the owner of the NFT (the original staker's address).

Resolution: Added at `d38a08d4c39ae6768ec4a623b2a51de53fa20e87` commit.

6.1.2 setEnd function is removed

Severity: *Code Improvement*

Context: staking.sol#L534

Description: The `setEnd` function is removed because the end staking period is defined as a constant.

Resolution: Added at `d38a08d4c39ae6768ec4a623b2a51de53fa20e87` commit.

6.1.3 Staking status is removed

Severity: *Code Improvement*

Context: staking.sol#L297

Description: The staking status is removed, and now a boolean variable `isCanceled` is used to track if the NFT is available to be claimed. When the NFT is unstaked, `isCanceled` is set to true.

Resolution: Added at `d38a08d4c39ae6768ec4a623b2a51de53fa20e87` commit.

6.1.4 Add a requirement to have at least 10 NFTs

Severity: *Code Improvement*

Description: New functionality is added to the code. Currently, a new potential staker should have at least 10 NFTs in their balance (staked or unstaked).

Resolution: Added at `d38a08d4c39ae6768ec4a623b2a51de53fa20e87` commit.

6.2 Gas

6.2.1 Packing of storage variables

Severity: *Gas*

Context: staking.sol#L278-L289

Description: Solidity contracts have contiguous 32 byte (256 bit) slots used for storage. When we arrange variables so multiple fit in a single slot, it is called variable packing. All storage variables were packed to optimize gas usage.

Resolution and Client comment: Resolved. Fixed at `c6f34998a5fc87a240c7ccf2606d7823c565b2f7` commit.

6.2.2 Cache array length outside of loop

Severity: *Gas*

Context: staking.sol#L369

Description: If not cached, the solidity compiler will always read the length of the array during each iteration. That is, if it is a storage array, this is an extra sload operation (100 additional extra gas for each iteration except for the first) and if it is a memory array, this is an extra mload operation (3 additional gas for each iteration except for the first).

Resolution and Client comment: Resolved. Fixed at `c6f34998a5fc87a240c7ccf2606d7823c565b2f7` commit.

6.2.3 For operations that will not overflow, you could use unchecked

Severity: *Gas*

Context: staking.sol#L392

Description: For operations that will not overflow, you could use unchecked.

Resolution and Client comment: Resolved. Fixed at `c6f34998a5fc87a240c7ccf2606d7823c565b2f7` commit.

6.2.4 Use custom errors instead of revert strings to save Gas

Severity: *Gas*

Context: staking.sol#L392-L360

Description: Custom errors are available from solidity version 0.8.4. Custom errors save ~50 gas each time they're hit by avoiding having to allocate and store the revert string.

Additionally, custom errors can be used inside and outside of contracts (including interfaces and libraries).

Resolution and Client comment: Resolved. Fixed at `c6f34998a5fc87a240c7ccf2606d7823c565b2f7` commit.

6.2.5 Use Constant and Immutable variables for variable that don't change

Severity: *Gas*

Context: staking.sol#L369

Description: Using the constant and the immutable keywords for variables that do not change helps to save on gas used. The reason been that constant and immutable variables do not occupy a storage slot when compiled. They are saved inside the contract byte code.

Resolution and Client comment: Resolved. Fixed at `c6f34998a5fc87a240c7ccf2606d7823c565b2f7` commit.

6.2.6 Indexed token from all events is removed

Severity: *Gas*

Context: staking.sol#L320-L344

Description: The indexed token which refers to the NFT collection is removed from every event. The address of the collection is hardcoded and there is no need to include it in the event logging.

Resolution and Client comment: Resolved. Fixed at `c6f34998a5fc87a240c7ccf2606d7823c565b2f7` commit.