# IQ NFT Staking Security Analysis

## by Pessimistic

This report is public

April 25, 2024

# Abstract

In this report, we consider the security of smart contracts of IQ NFT Staking project. Our task is to find and describe security issues in the smart contracts of the platform.

# Disclaimer

The audit does not give any warranties on the security of the code. A single audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, a security audit is not investment advice.

# Summary

In this report, we considered the security of IQ NFT Staking smart contracts. We described the audit process in the section below.

The initial audit showed one critical issue: The stakers are not able to claim their rewards. The audit also revealed two issues of medium severity: Unchecked returned value, Project roles. Moreover, several low-severity issues were found.

After the audit, the developers provided a new version of the code that includes fixes for most of the findings and provides comments on the M02 issue.

The overall code quality of the project is good.

# General recommendations

We do not have any additional recommendations.

# Project overview

## Project description

For the audit, we were provided with IQ NFT Staking project on a private GitHub repository, commit 8220867e2b07df7e850f9016915a8c9088eefa90.

The scope of the audit included:

- **contracts/staking/IIQNftStaking.sol**;
- **contracts/staking/IQNftStaking.sol**.

The documentation for the project included **IQNftStaking documentation.pdf** (sha1sum 7c03589173d63a6ef434a4152589bf9303c10fa6).

All 27 tests pass successfully. The code coverage is 77.08%.

The total LOC of audited sources is 245.

## Codebase update

After the initial audit, the codebase was updated. For the recheck, we were provided with commit d79b2d9c27bba01491e7acaace1d7dc3e678c399. This update included the fixes for all issues except for the M02 issue, comments on the M02 issue. In addition to this, the developers implemented additional tests.

All 32 tests pass successfully. The code coverage increased to 100%.

# Audit process

We started the audit on April 12, 2024 and finished on April 15, 2024.

We inspected the materials provided for the audit. Then, we contacted the developers for an introduction to the project. After a discussion, we performed preliminary research and started the review.

During the work, we stayed in touch with the developers and discussed confusing or suspicious parts of the code.

We manually analyzed all the contracts within the scope of the audit and checked their logic. Among other, we verified the following properties of the contracts:

- Users should be able to unstake their NFTs at any point in time;
- Users should be able to claim their reward tokens (see M02);
- NFT tokens cannot be withdrawn by the malicious actor;
- The signatures should not be replayable across the stakers;
- An owner should have access to the remaining amount of the reward tokens;
- Compliance of the code with the provided documentation.

We scanned the project with the following tools:

- Static analyzer Slither;
- Our plugin Slitherin with an extended set of rules;
- Semgrep rules for smart contracts. We also sent the results to the developers in the text file.

We ran tests and calculated the code coverage.

We combined in a private report all the verified issues we found during the manual audit or discovered by automated tools.

After the initial audit, we discussed the results with the developers. On April 24, 2024, the developers provided us with an updated version of the code. This update fixed most of the issues highlighted in our report and increased the code coverage.

We manually reviewed the updated codebase and also scanned it with the following tools:

- Static analyzer Slither;
- Our plugin Slitherin with an extended set of rules.

After that, we updated the report.

# Manual analysis

The contracts were completely manually analyzed, their logic was checked. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

## Critical issues

Critical issues seriously endanger project security. They can lead to loss of funds or other catastrophic consequences. The contracts should not be deployed before these issues are fixed.

### C01. The stakers are not able to claim their rewards (fixed)

After staking their NFTs, users should have an ability to claim the rewards by invoking the `claimTokens` function. However, the function accounts the claimed tokens internally without transferring them. In case the transfers are about to be handled by the `_proofSource`, an approval should be given to the backend.

Note that the reward tokens are not stuck, since the owner of the staking pool can deactivate the staking and withdraw all the reward tokens.

*The issue has been fixed and is not present in the latest version of the code.*

## Medium severity issues

Medium severity issues can influence project operation in the current implementation. Bugs, loss of potential income, and other non-critical failures fall into this category, as well as potential problems related to incorrect system management. We highly recommend addressing them.

### M01. Unchecked returned value (fixed)

Currently, any **ERC20** token could be accepted as a `_rewardToken` in order to reward the NFT stakers. Some older **ERC20** token versions either return or do not return a boolean value instead of revert upon transferring the tokens. An example of such token could be **USDT**. Consider utilizing the **SafeERC20** library from OpenZeppelin.

*The issue has been fixed and is not present in the latest version of the code.*

### M02. Project roles (commented)

In the current design, an owner has an ability to deactivate the staking and withdraw all the reward tokens from the pool at any time. Moreover, the functionality behind the `stake`, `claimTokens` and `withdraw` functions heavily relies on the signatures from the backend.

We recommend designing contracts in a trustless manner or implementing proper key management, e.g., setting up a multisig.

> *Comment from the developers:*
>
> *Pool Withdrawal Restrictions: The staking owner can only withdraw unallocated tokens from the pool. Tokens accrued as rewards by users are protected and cannot be accessed by the owner, ensuring security for stakers.*
>
> *User Protection: Users maintain full control to withdraw their NFTs and corresponding rewards at any time, even if staking is deactivated.*
>
> *Secure Signatures: We use EIP712 for secure, structured signatures, with secret key management handled by Google's Secrets Manager. This setup ensures high security.*
>
> *Simplicity and Efficiency: Adding multisig would complicate our EIP712 setup, requiring additional variables and reducing the efficiency of transactions. Our current design balances security and efficiency.*

## Low severity issues

Low severity issues do not directly affect project operation. However, they might lead to various problems in future versions of the code. We recommend fixing them or explaining why the team has chosen a particular option.

### L01. Incorrect calculation of tokens withdrawn by owner (fixed)

After deactivating the staking, an owner is able to withdraw some of the tokens used to reward the stakers that remain in the contract. Since the owner can withdraw the tokens in multiple separate transactions, the variable `_tokensWithdrawedByOwner` may contain an incorrect amount of tokens being withdrawn.

*The issue has been fixed and is not present in the latest version of the code.*

### L02. Unexpected revert in totalTokensLeft (fixed)

Currently, `totalTokensLeft` is calculated based on the `_poolSize`, `_totalTokensClaimed`, `_tokensWithdrawedByOwner` variables. However, the following invariant, which is required to be hold:

```
_totalTokensClaimed + _tokensWithdrawedByOwner <= _poolSize
```

might be broken, since the contract does not transfer the tokens upon claiming, but instead, accounts them internally. This causes an unexpected revert upon invoking `totalTokensLeft` function.

*The issue has been fixed and is not present in the latest version of the code.*

### L03. Possible gas optimization (fixed)

In the current design, the `withdraw` function calls `_removeNftFromStaking` for each NFT that is about to be withdrawn, which itself, in the worst case scenario, iterates over the whole `_stakedTokens[user]` array. It might be problematic in case of a large number of tokens being withdrawn.

*The issue has been fixed and is not present in the latest version of the code.*

This analysis was performed by Pessimistic:

Pavel Kondratenkov, Senior Security Engineer
Rasul Yunusov, Security Engineer
Konstantin Zherebtsov, Business Development Lead
Irina Vikhareva, Project Manager
Alexander Seleznev, Founder

April 25, 2024