HashEx
Blockchain Security

NFTY Staking

smart contracts final audit report

October 2021


hashex.org


contact@hashex.org

Contents

1. Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and HashEx and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (HashEx) owe no duty of care towards you or any other person, nor does HashEx make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties, or other terms of any kind except as set out in this disclaimer, and HashEx hereby excludes all representations, warranties, conditions, and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, HashEx hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against HashEx, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of

money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of the use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

2. Overview

HashEx was commissioned by the NFTY Labs team to perform an audit of their smart contract. The audit was conducted between October 28 and October 29, 2021.

The code located in @nftylabs/contracts GitHub repository was audited after the 75569af commit. The repository contains no tests for the audited contract. The code was provided without documentation.

The purpose of this audit was to achieve the following:

- Identify potential security issues with smart contracts.
- Formally check the logic behind given smart contracts.

Information in this report should be used to understand the risk exposure of smart contracts, and as a guide to improving the security posture of smart contracts, by remediating the issues that were identified.

2.1 Summary

Project name	NFTY Staking
URL	https://nftynetwork.io/
Platform	Binance Smart Chain
Language	Solidity

2.2 Contracts

Name	Address
BEP20RewardStakingV4	0x490147C65365c58F3404415b1194fbB4697A4B44

3. Found issues

Low	2 (40%)
Informational	3 (60%)

BEP20RewardStakingV4

ID	Title	Severity	Status
01	Gas optimization	Low	Acknowledged
02	Redundant variables	Low	Acknowledged
03	RFI rewards	Informational	Acknowledged
04	Reward tokens access	Informational	Acknowledged
05	Floating pragma	Informational	Acknowledged

4. Contracts

4.1 BEP20RewardStakingV4

4.1.1 Overview

This is a contract for staking tokens. The contract can be used for ordinary tokens, RFI tokens, and tokens with commissions on transfers. The rewards are accrued from the admin.

4.1.2 Issues

01. Gas optimization

- Low ⓘ Acknowledged

In the function getMultiplier() the global variable bonusEndBlock is read multiple times.

In the function setBonusEndBlock() the global variable bonusEndBlock is passed to the event instead of local.

In the function pendingReward() global variables poolInfo.lastRewardBlock and totalStaked are read multiple times.

In the function deposit() global variables user.amount, poolInfo.accRewardTokenPerShare and STAKE_TOKEN are read multiple times.

In the function withdraw() global variables user.amount and poolInfo.accRewardTokenPerShare are read multiple times.

In the function emergencyWithdraw() global variable user.amount is read multiple times.

Variables totalStaked, totalRewardsPaid, totalRewardsAllocated and totalAllocPoint are

zero-initialized.

The function getUnharvestedRewards() can be external.

02. Redundant variables

- Low ⓘ Acknowledged

Variables totalAllocPoint and allocPoint (in PoolInfo struct) are redundant because there is only one pool. This may lead to misunderstanding code by reviewers in some cases.

03. RFI rewards

- Informational ⓘ Acknowledged

If the RFI token is staked, but the rewards are distributed from the other token, then the RFI dividends will go to the contract owner.

04. Reward tokens access

- Informational ⓘ Acknowledged

Admin has access to all reward tokens and can withdraw them.

05. Floating pragma

- Informational ⓘ Acknowledged

It is a better practice to use fixed pragma in code.

5. Conclusion

No critical or high severity issues have been found. We recommend adding tests with coverage of at least 90%, before the deployment to the mainnet.

This audit includes recommendations on the code improving and preventing potential attacks.

Appendix A. Issues' severity classification

Critical. Issues that may cause an unlimited loss of funds or entirely break the contract workflow. Malicious code (including malicious modification of libraries) is also treated as a critical severity issue. These issues must be fixed before deployments or fixed in already running projects as soon as possible.

High. Issues that may lead to a limited loss of funds, break interaction with users, or other contracts under specific conditions. Also, issues in a smart contract, that allow a privileged account the ability to steal or block other users' funds.

Medium. Issues that do not lead to a loss of funds directly, but break the contract logic. May lead to failures in contracts operation.

Low. Issues that are of a non-optimal code character, for instance, gas optimization tips, unused variables, errors in messages.

Informational. Issues that do not impact the contract operation. Usually, informational severity issues are related to code best practices, e.g. style guide.

Appendix B. List of examined issue types

- Business logic overview
- Functionality checks
- Following best practices
- Access control and authorization
- Reentrancy attacks
- Front-run attacks
- DoS with (unexpected) revert
- DoS with block gas limit
- Transaction-ordering dependence
- ERC/BEP and other standards violation
- Unchecked math
- Implicit visibility levels
- Excessive gas usage
- Timestamp dependence
- Forcibly sending ether to a contract
- Weak sources of randomness
- Shadowing state variables
- Usage of deprecated code

HashEx
Blockchain Security