verichains

SECURITY AUDIT OF

STAKING NFT REWARD

Public Report

Mar 11, 2024

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Blast	Blast is the only Ethereum L2 with native yield for ETH and stablecoins.
Ether (ETH)	A cryptocurrency whose blockchain is generated by the Ethereum platform. Ether is used for payment of transactions and computing services in the Ethereum network.
Smart contract	A computer protocol intended to digitally facilitate, verify or enforce the negotiation or performance of a contract.
Solidity	A contract-oriented, high-level language for implementing smart contracts for the Ethereum platform.
Solc	A compiler for Solidity.
ERC721	The ERC-721 introduces a standard for NFT, in other words, this type of Token is unique and can have different value than another Token from the same Smart Contract, maybe due to its age, rarity or even something else like its visual

EXECUTIVE SUMMARY

This Security Audit Report was prepared by Verichains Lab on Mar 11, 2024. We would like to thank the Blast Penguins for trusting Verichains Lab in auditing smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the Staking NFT Reward. The scope of the audit is limited to the source code files provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team had identified no vulnerable issue in the contract code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Staking NFT Reward

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the Staking NFT Reward. It was conducted on commit `2251fd3c6c40a184f2f125efe588b5aaef5e3b92` from git repository link: *https://github.com/Ildarflame/BlastPenguins_NFTStaking*

1.3. Audit methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and RK87, our in-house smart contract security analysis tool.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that were considered during the audit of the smart contract:

- Integer Overflow and Underflow
- Timestamp Dependence
- Race Conditions
- Transaction-Ordering Dependence
- DoS with (Unexpected) revert
- DoS with Block Gas Limit
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Reentrancy
- Explicit visibility of functions state variables (external, internal, private and public)
- Logic Flaws

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 1. Severity levels

1.4. Disclaimer

Blast Penguins acknowledges that the security services provided by Verichains, are conducted to the best of their professional abilities but cannot guarantee 100% coverage of all security vulnerabilities. Blast Penguins understands and accepts that despite rigorous auditing, certain vulnerabilities may remain undetected. Therefore, Blast Penguins agrees that Verichains shall not be held responsible or liable, and shall not be charged for any hacking incidents that occur due to security vulnerabilities not identified during the audit process.

1.5. Acceptance Minute

This final report served by Verichains to the Blast Penguins will be considered an Acceptance Minute. Within 7 days, if no any further responses or reports is received from the Blast Penguins, the final report will be considered fully accepted by the Blast Penguins without the signature.

2. AUDIT RESULT

2.1. Overview

The Staking NFT Reward was developed using the `Solidity` language, with a required version of `^0.8.20`. The source code was structured based on OpenZeppelin's library.

2.2. Blast Penguins

The `StakingNFTReward` contract extends the functionality of the NFTStorage, Ownable, and ReentrancyGuard contracts.

The primary features of the contract include enabling users to stake their NFTs and earn rewards in the form of the platform's native token.

The contract utilizes the Ownable contract to designate the owner and ReentrancyGuard to mitigate reentrancy attacks. The owner is granted significant privileges, such as modifying yield limits, the NFT contract address, and the reward token address.

Stake

Users can stake their native tokens by transferring them to the contract. These tokens represent a share amount used to calculate rewards.

User rewards are determined by multiplying the number of native tokens staked by the user with the current yield per share. If rewards are available, the user will receive a specific number of NFTs as a reward.

Unstake

Users can unstake a specified number of shares and receive their native tokens back. If the contract holds sufficient NFTs, users may also receive additional NFTs as a reward.

Claim

Users who wish to claim rewards without unstaking their shares can do so by calling the `claimNfts` function, resulting in the receipt of a certain number of NFTs as a reward.

Owner Privileges

The owner has the authority to adjust the yield limit and the NFT contract address.

The yield limit directly impacts the number of NFTs users receive as rewards.

The NFT contract, an ERC721 contract, determines the specific NFTs users may receive as rewards. Users may receive different NFTs from various contracts during the staking process.

Additionally, the owner can add or remove NFT IDs eligible for rewards. When the contract holds a sufficient number of NFTs, users will receive them as rewards. The owner also has the capability to withdraw native tokens stuck in the contract by invoking the `collectYield` function.

2.3. Findings

During the audit process, the audit team had identified no vulnerable issue in the contract code.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Mar 11, 2024*	Public Report	Verichains Lab

Table 2. Report versions history