Smart Contract Audit Report

Nitro League

October 28, 2022

This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process; therefore, running a bug bounty program as a complement to this audit is strongly recommended.

Introduction

1. About Nitro League

Nitro League, an immersive racing world is the ultimate combination of GameFi, DeFi, and NFT. Nitro League is a dynamic hub for collectors, investors, and artists built on a robust virtual economy.

Nitro facilitates ownership, customization, collection, and trade for unique NFT cars and parts. Players may hone their racing skills and earn rewards for their performance. Anyone can fulfill their dream of becoming a racer in the Nitro League.

Nitro League has devised a fully functional virtual economy that enables players to buy land and buildings and build cars. Nitro League facilitates players to expand their guilds and empires.

Visit https://www.nitroleague.com/ to know more about it.

2. About ImmuneBytes

ImmuneBytes is a security start-up that provides professional services in the blockchain space. The team has hands-on experience conducting smart contract audits, penetration testing, and security consulting. ImmuneBytes's security auditors have worked on various A-league projects and understand DeFi projects like AAVE, Compound, 0x Protocol, Uniswap, and dydx.

The team has secured 205+ blockchain projects by providing security services on different frameworks. The ImmuneBytes team helps start-ups with detailed system analysis, ensuring security and managing the overall project.

Visit http://immunebytes.com/ to learn more about the services.

Documentation Details

The Nitro League team has provided the following doc for audit:

1. https://nitroblockchain.notion.site/nitroblockchain/Standard-Contract-Nitro-League-9efedf36fcec419fac5bca5acaa3ee8f
2. https://docs.google.com/document/d/1bDHuZuAevdwHT42_INMfA9Yn1D7NSRZ4MtJBZfU6SoQ/edit

Audit Process & Methodology

ImmuneBytes team has performed thorough testing of the project, starting with analyzing the code design patterns in which we reviewed the smart contract architecture to ensure it is structured and safe use of third-party smart contracts and libraries.

Our team then performed a formal line-by-line inspection of the Smart Contract to find potential issues like Signature Replay Attacks, Unchecked External Calls, External Contract Referencing, Variable Shadowing, Race conditions, Transaction-ordering dependence, timestamp dependence, DoS attacks, and others.

In the Unit testing phase, we run unit tests written by the developer to verify the functions work as intended. In Automated Testing, we tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

The code was audited by a team of independent auditors, including -

1. Structural analysis of the smart contract is checked and verified.
2. An extensive automated testing of all the contracts under scope is conducted.
3. Line-by-line Manual Code review is conducted to evaluate, analyze and identify the potential security risks in the contract.
4. Evaluation of the contract's intended behavior and the documentation shared is imperative to verify the contract behaves as expected.
5. For complex and heavy contracts, adequate integration testing is conducted to ensure that contracts interact acceptably.
6. Storage layout verifications in the upgradeable contract are a must.
7. An important step in the audit procedure is highlighting and recommending better gas optimization techniques in the contract.

Audit Details

- Project Name: Nitro League
- Languages: Solidity(Smart contract), Typescript (Unit Testing)
- Github link: https://bitbucket.org/tkxel/nitroleague-core/src/master/
- Commit hash: f5eece6ac3f8284ef29a83ce1f945a004b34d261
- Platforms and Tools: Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, Contract Library, Slither, SmartCheck

This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process; therefore, running a bug bounty program as a complement to this audit is strongly recommended.

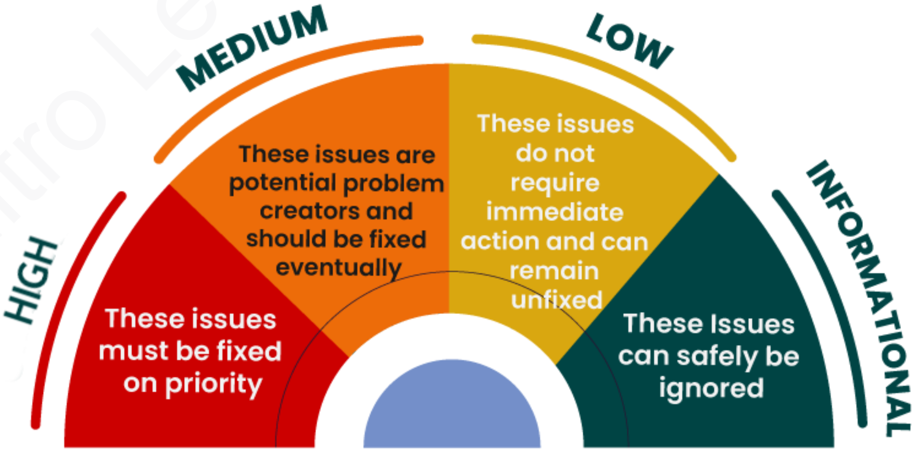
Audit Goals

The audit's focus was to verify that the smart contract system is secure, resilient, and working according to its specifications. The audit activities can be grouped into the following three categories:

1. Security: Identifying security-related issues within each contract and the system of contracts.
2. Sound Architecture: Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.
3. Code Correctness and Quality: A full review of the contract source code. The primary areas of focus include
    a. Correctness
    b. Readability
    c. Sections of code with high complexity
    d. Quantity and quality of test coverage

Security Level Reference

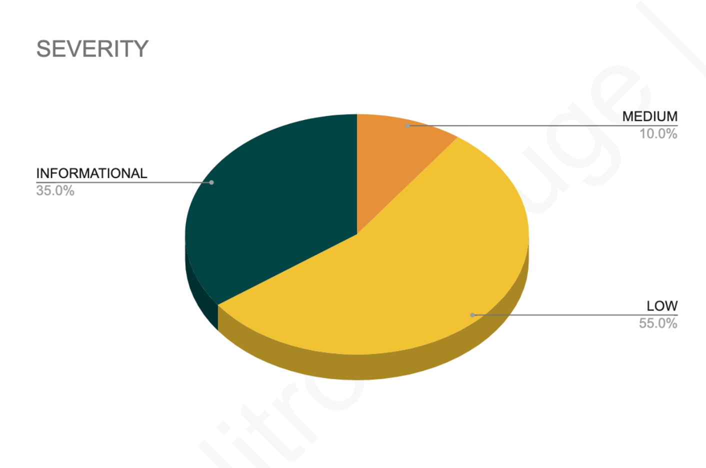
Every issue in this report were assigned a severity level from the following:

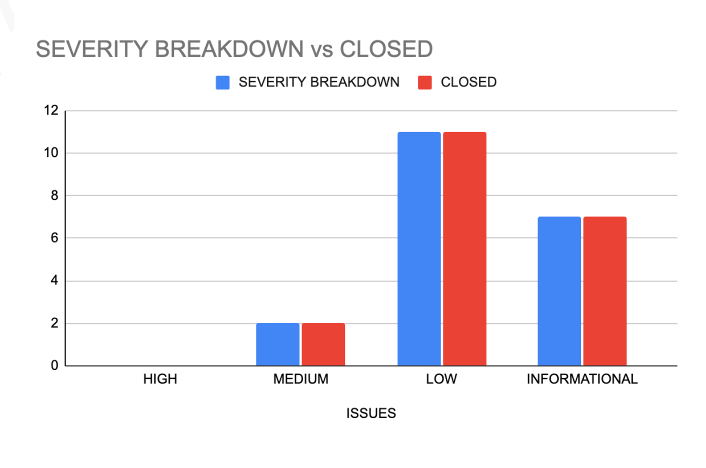

HIGH
These issues must be fixed on priority
MEDIUM
These issues are potential problem creators and should be fixed eventually
LOW
These issues do not require immediate action and can remain unfixed
INFORMATIONAL
These Issues can safely be ignored

Audit Summary

Team ImmuneBytes has performed a line-by-line manual analysis and automated review of smart contracts. Smart contracts were analyzed mainly for common contract vulnerabilities, exploits, and manipulation hacks. According to the audit:

Issues	High	Medium	Low
Open	-	-	-
Closed	-	2	10
Acknowledged	-	-	1





This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process; therefore, running a bug bounty program as a complement to this audit is strongly recommended.

Finding Overview

S.no	Findings	Risk	Status
01	minPlayers can be greater than maxPlayers if wrongly initialized. Can lead to unwanted behavior	Medium	Closed
02	Wrong function logic leads to invalid emission of events	Medium	Closed
03	Costly loop operations could be avoided	Low	Closed
04	Coding Style Issues in the Contract	Informational	Closed
05	Commented codes must be wiped out before deployment	Informational	Closed
06	Unlocked Pragma statements found in the contracts	Informational	Closed
07	claimRewards() function should only be executable in AWARDED state.	Low	Closed
08	No input validations to avoid array length mismatch	Low	Closed
09	Costly Loop operations found	Low	Closed
10	Unlocked Pragma statements found in the contracts	Informational	Closed
11	No Events emitted after imperative State Variable modification	Low	Closed
12	Absence of input validations after important state modifications	Low	Closed
13	External Visibility should be preferred	Low	Closed
14	Coding Style Issues in the Contract	Informational	Closed
15	Unlocked Pragma statements found in the contracts	Informational	Closed
16	Absence of Zero Address validation	Low	Closed
17	External Visibility should be preferred	Low	Closed
18	Unlocked Pragma statements found in the contracts	Informational	Closed
19	The reset of mintCount by the owner does not follow the intended procedure	Low	Acknowledged
20	setDailyMintLimit() function doesn't validate the arguments.	Low	Closed

Concluding Remarks

While conducting the audits of the Nitro League smart contracts, it was observed that the contracts contain Medium and Low severity issues.

Our auditors suggest that Medium and Low severity issues should be resolved by the developers. The recommendations given will improve the operations of the smart contract.

Note:
The Nitro team has fixed the issues based on the auditor's recommendation.

Disclaimer

ImmuneBytes's audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process; therefore, running a bug bounty program as a complement to this audit is strongly recommended.

Our team does not endorse the Nitro League platform or its product nor this audit is investment advice.
Notes:

- Please make sure contracts deployed on the mainnet are the ones audited.
- Check for code refactoring by the team on critical issues.