QuillAudits

AUDIT REPORT

January 2025

For

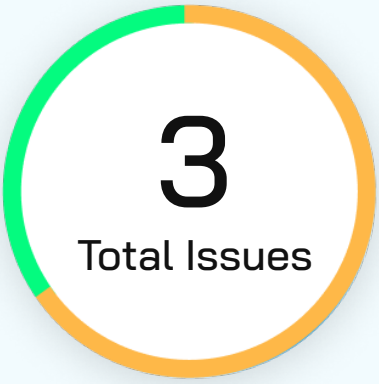
NODERZZ

Table of Content

Executive Summary

Project Name	Noderzz Protocol
Project URL	https://www.noderzz.xyz/
Overview	Noderzz Protocol is a vanilla staking contract that allows users to stake funds and withdraw them after a certain lock-in period.
Audit Scope	The scope of this Audit was to analyze the Noderzz Smart Contracts for quality, security, and correctness.
Contracts in Scope	https://github.com/mintair-xyz/noderzz-staking StakingV1.sol Proxy.sol
Commit Hash	4f0f1687ff36c4142cb7b93e313cba068ab482bc
Language	solidity
Blockchain	Ethereum
Method	Manual Analysis, Functional Testing, Automated Testing
Review 1	15th January 2025 - 16th January 2025
Updated Code Received	17th January 2025
Review 2	27th January 2025 - 29th January 2025
Fixed In	https://github.com/mintair-xyz/noderzz-staking 0c2bc22d83e3927bbb72acc426b45c0aaf548294

Number of Issues per Severity



High	0 (0%)
Medium	2 (66%)
Low	1 (33%)
Informational	2 (33%)

Severity

Issues

	High	Medium	Low	Informational
Open	0	0	0	0
Resolved	0	1	1	0
Acknowledged	0	1	0	0
Partially Resolved	0	0	0	0

Checked Vulnerabilities

- Access Management
- Arbitrary write to storage
- Centralization of control
- Ether theft
- Improper or missing events
- Logical issues and flaws
- Arithmetic Computations Correctness
- Race conditions/front running
- SWC Registry
- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Malicious libraries
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- ERC's conformance
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level calls

Checked Vulnerabilities

- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Multiple Sends
- Using suicide
- Using delegatecall
- Upgradeable safety
- Using throw
- Using inline assembly
- Style guide violation
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contracts, care was taken to ensure:

- The overall quality of code
- Use of best practices
- Code documentation and comments, match logic and expected behavior
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper
- Implementation of ERC standards
- Efficient use of gas
- Code is safe from re-entrancy and other vulnerabilities

The following techniques, methods, and tools were used to review all the smart contracts:

Structural Analysis

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Techniques and Methods

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behavior of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms Used for Audit

Remix IDE, Foundry, Solhint, Mythril, Slither, Solidity statistic analysis.

Types of Severity

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open Security vulnerabilities identified that must be resolved and are currently unresolved.	**Resolved** These are the issues identified in the initial audit and have been successfully fixed.
Acknowledged Vulnerabilities which have been acknowledged but are yet to be resolved.	**Partially Resolved** Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Medium Severity Issues

1. Uses Ownable2StepUpgradeable instead of OwnableUpgradeable

Resolved

Path
StakingV1.sol

Function Name
initialize()

Description
One critical issue is that the contract does not utilize the __Ownable2StepUpgradeable_init() function, which is designed to facilitate a two-step ownership transfer process. Without this mechanism, ownership transfers occur immediately, exposing the contract to potential risks. The two-step process enhances security by requiring a confirmation step before ownership is fully transferred, thereby reducing the likelihood of unauthorized access or control over the contract.

Recommendation
Use **__Ownable2StepUpgradeable_init() instead of __Ownable_init(msg.sender)**

2. StakingV1 is not able to handle Fee on Transfer Tokens

Acknowledged

Path
StakingV1.sol

Description
Currently StakingV1 doesn't properly handle fee-on-transfer tokens, which could cause issues. Fee-on-transfer tokens (like SafeMoon or similar) deduct a fee during transfers, meaning the amount received is less than the amount sent.

Recommendation
Either add a defense mechanism for fees on transfer, or be cautious when setting unusual tokens.

Low Severity Issues

3. Consider sanitizing user inputs

Resolved

Path

StakingV1.sol

Description

Here are some points to work on to create a more secure and robust smart contract. Additionally, some invariant breaks may cause unintended consequences.

Here is a list of core invariants that should be checked:

- Ensure that the same stake ID is not withdrawn in the same transaction. However, it is currently not vulnerable.
- Add a pause/unpause mechanism in the contract to enable emergency halts.
- initialize function does not check for address(0).
- Instead of using a hardcoded lockPeriod, use the onlyOwner() function.
- Limit batch withdrawals to a safe, gas-efficient size to prevent failures.

Recommendation

Consider sanitizing these inputs against the invariants.

Functional Tests

Some of the tests performed are mentioned below:

- Stake function works as per expectation however it does follow CEI method.
- Partial withdrawal of stake ID works perfectly.
- If withdrawing more than the staked amount, then only the staked amount is withdrawn.
- Same ID is passed as a parameter to the withdraw function, but it cannot be withdrawn twice. However, this is not a good practice.
- All the view function works perfectly.

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of Noderzz. We performed our audit according to the procedure described above.

Some issues of Mediumand Low severity were found. Some suggestions and best practices are also provided in order to improve the code quality and security posture. In the end, Noderzz team resolved almost all issues and acknowledged one issue.

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in Noderzz. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of Noderzz. One audit is not enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your smart contracts for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services. It is the responsibility of Noderzz to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

About QuillAudits

QuillAudits is a leading name in Web3 security, offering top-notch solutions to safeguard projects across DeFi, GameFi, NFT gaming, and all blockchain layers. With six years of expertise, we've secured over 1000 projects globally, averting over $30 billion in losses. Our specialists rigorously audit smart contracts and ensure DApp safety on major platforms like Ethereum, BSC, Arbitrum, Algorand, Tron, Polygon, Polkadot, Fantom, NEAR, Solana, and others, guaranteeing your project's security with cutting-edge practices.

QuillAudits

6+ Years of Expertise	**1M+** Lines of Code Audited
$30B+ Secured in Digital Assets	**1K+** Projects Secured

Follow Our Journey

AUDIT REPORT

January 2025

For

NODERZZ

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com audits@quillhash.com