QuillAudits

Audit Report
September, 2023

For

NORDEK

Table of Content

Table of Content

Scope of Audit

The scope of this audit was to analyze and document the Nordek smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- 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.
- 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.

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, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistical analysis, Theo.

Issue Categories

Every issue in this report has been assigned 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 arises because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

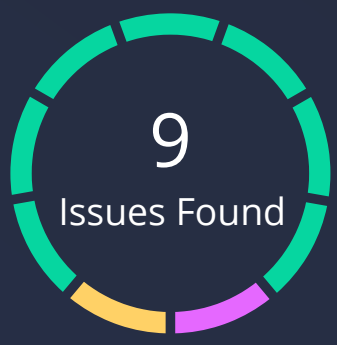
Low Severity Issues

Low-level severity issues can cause minor impacts and are just warnings that can remain unfixed. 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.

The Number of Security Issues per Severity



High
Medium
Low
Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	3	0
Resolved Issues	0	1	4	1

Introduction

Initial Audit	During the period of 23rd May 2023 to 7th June 2023.
Update Code	Received updated code from Nordek Team on 29th August,2023.
Final Audit	During the Period of 30th August, 2023 to 1st September, 2023 QuillAudits Team performed a security audit for the Nordek smart contract.
Code Base	https://github.com/Officialnordek/network/tree/master/contracts
Commit Hash	a282e8f5c8d9707e38e319d7537eb1cdd22e944c
Fixed In	https://github.com/Officialnordek/network/commit/fbed7a8c920e6eede8774577013bab859cef6cfc

Issues Found – Code Review / Manual Testing

High Severity Issues

No issues found

Medium Severity Issues

M.1 The owner is a single point of failure and a centralization risk

Description

Having a single EOA as the only owner of contracts is a large centralization risk and a single point of failure. A single private key may be taken in a hack, or the sole holder of the key may become unable to retrieve the key when necessary.

Recommendation

Consider changing to a multi-signature setup, or having a role-based authorization model.

Status

Resolved

Fixed In

https://github.com/Officialnordek/network/commit/fbed7a8c920e6eede8774577013bab859cef6cfc

Note

Recommend to use GenosisSafe for deploying a multisig. And use the proxy to execute the transactions with a good threshold.

Low Severity Issues

L.1 Pause and unPause functionality missing

Description

During the code review, It has been noticed that pausing and unpausing of user-centric functions are not present.

Recommendation

Consider adding a pause unpause functionality to the system.

L.1 Pause and unPause functionality missing

Fixed In

https://github.com/Officialnordek/network/commit/1846ce139991046c85bac97c2c7d803e65b58199

Status

Resolved

L.2 No function to withdraw the accidentally sent token to the contracts

Description

Accidentally sent tokens can be locked and will not be able to withdraw from the system.

Recommendation

We recommend making a withdraw functionality which is access controlled to withdraw the accidentally sent token to the contracts.

Status

Acknowledged

L.3 Used locked pragma version

Description

The pragma versions used in the contract are not locked. Consider using the latest versions among 0.8.19 for deploying the contracts and libraries as it does not compile for any other version and can be confusing for a developer. Solidity source files indicate the versions of the compiler they can be compiled with.

```
pragma solidity ^0.8.0; // bad: compiles between 0.8.0 and 0.8.19
pragma solidity 0.8.0; // good: compiles w 0.8.0 only but not the latest version
pragma solidity 0.8.19; // best: compiles w 0.8.19
```

Recommendation

Use best compiles and locked pragma in the contracts.

L.3 Used locked pragma version

Status

Resolved

Fixed In

https://github.com/Officialnordek/network/commit/9ef3f095e7c118ea60655d3782a5a12520ee67b6

L.4 Local Hardhat test failing

Description

Many tests are getting failed on Hardhat in the local env.

Recommendation

We recommend fixing all the failing local tests.

Status

Resolved

L.5 Inherited OwnableUpgradeable uses a single-step ownership transfer

Description

During the code review, It has been noticed that the contracts use single-step ownership transfer in the contract EternalStorageProxy.sol.

Recommendation

Consider using the Ownable2StepUpgradable contract customize version in Solc 0.4.24 for the implementation.

Status

Acknowledged

L.6 SafeTransfer Methods missing

Description

It is good to add a require() statement that checks the return value of token transfers or to use something like a safeTransfer/safeTransferFrom unless one is sure the given token reverts in case of a failure. Failure to do so will cause silent failures of transfers and affect token accounting in the contract.

Recommendation

Replace transfer, transferFrom() with safeTransfer, safeTransferFrom() as ERC20 token implementation can be any. If transferFrom() does not return a value (e.g., USDT), the transaction reverts because of a decoding error. Revert without error.

Status

Acknowledged

L.7 Remove unused functions

Description

Remove _setSnapshot, getSnapshotsPerCycle, _setNextSnapshotId, _setLastSnapshotTakenAtBlock, _getSeed, and _getRandom in ConsensusUtil.sol as it's been not used anywhere and return unnecessary.

Recommendation

Removing the unused function reduces the size of the deployment cost.

Status

Resolved

Fixed In

https://github.com/Officialnordek/network/commit/2c89fc258fe4907db3d890357e227a1f48b3e8fd

Informational Issues

I.1 Recommendations and Gas optimizations

- For test stake use smock Library.
- Gas optimization
  - The pre-increment operation is cheaper (about 5 GAS per iteration) so use ++i instead of i++ or i+= 1 in for loop. We recommend using pre-increment in all the for loops.
  - != 0 costs 6 less GAS compared to > 0 for unsigned integers in require statements with the optimizer enabled. We recommend using !=0 instead of > 0 in all the contracts.
  - In for loop, the default value initialization to 0 should be removed from all the for loops.
  - In the EVM, there is no opcode for non-strict inequalities (>=, <=) and two operations are performed (> + =.) Consider replacing >= with the strict counterpart >. Recommend following the inequality with a strict one.
    - The compiler uses opcodes `GT` and `ISZERO` for solidity code that uses `>`, but only requires `LT` for `>=`, [which saves 3 gas] #REF
  - All the public functions which are not used internally need to be converted to external.
  - Using private rather than public for constants saves gas
    - If needed, the values can be read from the verified contract source code, or if there are multiple values there can be a single getter function that [returns a tuple] of the values of all currently-public constants. Saves 3406-3606 gas in deployment gas due to the compiler not having to create non-payable getter functions for deployment calldata, not having to store the bytes of the value outside of where it's used, and not adding another entry to the method ID table. #REF
  - Follow the ordering of the NetSpac guide and comments for NetSpac.

Status

Resolved

Fixed In

https://github.com/Officialnordek/network/commit/2502b56f8e0c2e0094132f705e5762ad03796647

Functional Tests

Some tests performed are mentioned below:

- BlockReward
  - ✓ Initial setup checked
  - ✓ getBlockRewardAmountPerValidator positive and negative scenarios
  - ✓ UpgradeTo
  - ✓ Reward checks and flow
- Consensus
  - ✓ Initial setup checked
  - ✓ setProxyStorage, emitInitiateChange, finalizeChange
  - ✓ Staking support
  - ✓ Cycles and snapshots
  - ✓ getDelegatorsForRewardDistribution and validators
  - ✓ UpgradeTo
- ProxyStorage
  - ✓ Initial setup checked
  - ✓ UpgradeTo
- Voting
  - ✓ Initial setup checked
  - ✓ Ballot functionality
  - ✓ onCycleEnd
  - ✓ UpgradeTo

Closing Summary

Most of the Issues are fixed by Nordek Team and we recommend to deploy a multisig using genosis safe and use the proxy with at least a Thresold of 80%.

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in Nordek smart contracts. 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 Nordek smart contracts. 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 the Nordek to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

850+
Audits Completed

$30B
Secured

800K
Lines of Code Audited

Follow Our Journey

Audit Report
September, 2023

For

NORDEK

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com