NUON Audit

April 2023

By CoinFabrik

v202209

Executive Summary **3**
Scope **3**
Methodology **3**
Findings **4**
Severity Classification 4
Issues Status 4
Critical Severity Issues 5
Medium Severity Issues 5
ME-01 Uninitialized Implementation Contract 5
Minor Severity Issues 5
Enhancements 5
EN-01 Flattening Contracts At Deployment 5
Other Considerations **6**
Centralization 6
Upgrades 6
CollateralHub Transfers More Than Argument Value 6
Changelog **7**

Executive Summary

CoinFabrik was asked to audit the contracts for the NUON project. The audited files are from the git repository located at https://github.com/nuonfinance/NUON-Ecosystem. The audit is based on the commit `f3422cfded48ad6550e22661545d026a5142fa33`.

During this audit we found one medium issue. Also, an enhancement was proposed.

Scope

The scope for this audit includes and is limited to the following files:

- `contracts/CollateralHub-ETH_flatV3.sol`: Contract where users deposit WETH and mint NUON and NLP's NFT, providing liquidity to the pool.
- `contracts/NuonControllerV3.sol`: Interface for administration of the collateral hubs. There, parameters such as fees and collateral ratios are set.

No other files in this repository were audited. Its dependencies are assumed to work according to their documentation. Also, no tests were reviewed for this audit.

Methodology

CoinFabrik was provided with the source code, including automated tests that define the expected behavior, and general documentation about the project. Our auditors spent two weeks auditing the source code provided, which includes understanding the context of use, analyzing the boundaries of the expected behavior of each contract and function, understanding the implementation by the development team (including dependencies beyond the scope to be audited) and identifying possible situations in which the code allows the caller to reach a state that exposes some vulnerability. Without being limited to them, the audit process included the following analyses:

- Arithmetic errors
- Outdated version of Solidity compiler
- Race conditions
- Reentrancy attacks
- Misuse of block timestamps
- Denial of service attacks
- Excessive gas usage
- Missing or misused function qualifiers
- Needlessly complex code and contract interactions
- Poor or nonexistent error handling
- Insufficient validation of the input parameters

- Incorrect handling of cryptographic signatures
- Centralization and upgradeability

Findings

In the following table we summarize the security issues we found in this audit. The severity classification criteria and the status meaning are explained below. This table does not include the enhancements we suggest to implement, which are described in a specific section after the security issues.

ID	Title	Severity	Status
ME-01	Uninitialized Implementation Contract	Medium	Unresolved

Severity Classification

Security risks are classified as follows:

- **Critical:** These are issues that we manage to exploit. They compromise the system seriously. They must be fixed **immediately**.
- **Medium:** These are potentially exploitable issues. Even though we did not manage to exploit them or their impact is not clear, they might represent a security risk in the near future. We suggest fixing them **as soon as possible**.
- **Minor:** These issues represent problems that are relatively small or difficult to take advantage of, but might be exploited in combination with other issues. These kinds of issues do not block deployments in production environments. They should be taken into account and be fixed **when possible**.

Issues Status

An issue detected by this audit has one of the following statuses:

- **Unresolved**: The issue has not been resolved.
- **Acknowledged**: The issue remains in the code, but is a result of an intentional decision.
- **Resolved**: Adjusted program implementation to eliminate the risk.

- **Partially resolved**: Adjusted program implementation to eliminate part of the risk. The other part remains in the code, but is a result of an intentional decision.
- **Mitigated**: Implemented actions to minimize the impact or likelihood of the risk.

Critical Severity Issues

No issues found.

Medium Severity Issues

ME-01 Uninitialized Implementation Contract

Location:

- `contracts/CollateralHub-ETH_flatV3.sol`

An attacker might take ownership of the implementation contract and deceive users into operating with that contract instead of the proxy. The deployer will be an address validated by the organization, but the attacker could withdraw funds from the contract.

Recommendation

To prevent the implementation contract from being used, you should invoke the `_disableInitializers()` function in the constructor to automatically lock it when it is deployed.[1]

Status

Unresolved.

Minor Severity Issues

No issues found.

Enhancements

These items do not represent a security risk. They are best practices that we suggest implementing.

ID	Title	Status
EN-01	Flattening Contracts At Deployment	Not implemented

[1] https://docs.openzeppelin.com/contracts/4.x/api/proxy#Initializable-_disableInitializers–

EN-01 Flattening Contracts At Deployment

Location:

- `contracts/CollateralHub-ETH_flatV3.sol`
- `contracts/NuonControllerV3.sol`

Project libraries are embedded into contracts files at the development stage. This leads to outdated libraries because the code is freezed and might result in changes on the libraries code.

Recommendation

Include libraries flattening contracts at deployment.

Status

Not implemented.

Other Considerations

The considerations stated in this section are not right or wrong. We do not suggest any action to fix them. But we consider that they may be of interest to other stakeholders of the project, including users of the audited contracts, token holders or project investors.

Centralization

The contracts included in the scope have an owner address defined.

`NuonController` contract has an administrator who sets the parameters (fees, collateralization ratios, oracles) for each collateral hub.

In `CollateralHubETHV3`, the owner can pause the functions which involve deposits into the contract. Also, this role can call to `mint()` without adding the extra liquidity required for the liquidity pool.

Upgrades

`CollateralHubETHV3` implements a mechanism for upgrading, while `NuonController` does not.

CollateralHub Transfers More Than Argument Value

In the collateral hub, the caller passes the amount of collateral to transfer in order to mint `NUON`. However, an extra amount of collateral tokens will be transferred from users' balance to be added to the liquidity pool. Therefore, consider that the value used as argument and

the actual amount transferred from the account are not equal. In fact, the second will be larger.

This also happens when calling `mintWithoutDeposit()`, a function whose name might lead you to think it does not transfer tokens from the caller's account. Since it also provides liquidity to the pool, tokens could be transferred.

Changelog

- 2023-04-20 – Initial report based on commit `f3422cfded48ad6550e22661545d026a5142fa33`.

Disclaimer: This audit report is not a security warranty, investment advice, or an approval of the NUON project since CoinFabrik has not reviewed its platform. Moreover, it does not provide a smart contract code faultlessness guarantee.