Nomis Security Scan Results

by Pessimistic

This is not a security audit

This report is public

May 27, 2024

Abstract

This report considers the security of smart contracts of the Nomis protocol. Our task is to find and describe security issues using the static-analysis tools Slither and Slitherin and help resolve them.

The work is financially covered by the Arbitrum Foundation grant.

Disclaimer

Summary

In this report, we described issues found in smart contracts of the Nomis protocol.

We scanned the codebase and manually rejected or verified all automated findings, revealing two relevant issues.

The developers fixed and commented on one issue.

The entire process is described in the section below.

Scan process

Under the Arbitrum Foundation grant, we researched and developed Arbitrum-specific detectors. They became publicly available with Slitherin v0.6.0 release.

Workflow

This work consisted of five stages:

1. For the scan, we were provided with the Nomis project on a private GitHub repository, commit df15683c6e2f4f53dc79f196a84dd2e4f8c51f22.
2. For the analysis of the protocol, we launched Slither v0.10.1 and Slitherin v0.6.0 on the provided codebase.
3. One auditor manually checked (rejected or accepted) all findings reported by the tools. The second auditor verified this work. We shared all relevant issues with the protocol developers and answered their questions.
4. The developers reviewed the findings, updated the code accordingly, and gave comments on issues. We reviewed the fixes and found no new issues.
5. We prepared this final report summarizing all the issues and comments from the developers.

Issue categories

Within the confines of this work, we were looking for:

- Arbitrum-specific problems;
- Standard vulnerabilities like re-entrancy, overflow, arbitrary calls, etc;
- Non-compliance with popular standards like ERC20 and ERC721;
- Some access control problems;
- Integration issues with some popular DeFi protocols;
- A wide range of code quality and gas efficiency improvement opportunities.

This scan does not guarantee that these issues are not present in the codebase.

Scan results

Issue category	Number of detectors	Status
Compilation	1	Passed
Arbitrum Integration	3	Passed
AAVE Integration	1	Passed
Uniswap V2 Integration	7	Passed
OpenZeppelin	2	Passed
ERC-20	7	Passed
ERC-721	2	Passed
Known Bugs	15	1 issue found (1/1 fixed)
Access Control	3	Passed
Arbitrary Call	5	Passed
Re-entrancy	6	Passed
Weak PRNG	2	Passed
Upgradability	2	Passed
Ether Handling	3	Passed
Low-level Calls	2	Passed
Assembly	2	Passed
Inheritance	3	Passed
Arithmetic	2	Passed
Old Solidity Versions Bugs	10	Passed
Code Quality	15	Passed
Best Practices	4	Passed
Gas	7	1 issue found

Discovered Issues

Uninitialized variable

In the **NomisScore.sol** contract, the `_individualReward` `mapping` is never initialized. However, its default values are used in `setScore`, `claimReferralRewards`, `getClaimableReward` functions without impacting on their logic, as when the `_individualReward` is equal to zero, the value of `_referralReward` is used instead. We recommend removing the `_individualReward` `mapping` to improve code quality and decrease gas consumption, or adding a function to modify the values of the `_individualReward` `mapping`.

> *Comment from the developers: Yes, this bug exists in this version of the code. There is a function for modifying the `_individualReward` values in the current version of the code.*

External vs public

The `getWalletsByReferrerCode` and `initialize` functions of the **NomisScore** contract can be declared as `external`. Consider declaring functions as `external` instead of `public` when possible to improve code readability and optimize gas consumption.

This analysis was performed by Pessimistic:

Egor Dergunov, Junior Security Engineer
Rasul Yunusov, Security Engineer
Evgeny Bokarev, Junior Security Engineer
Nikita Kirillov, Product Owner
Evgeny Marchenko, CTO
Konstantin Zherebtsov, Business Development Lead

May 27, 2024