HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: LeagueDAO
Date: February 21st, 2022

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Smart Contract Code Review and Security Analysis Report for League DAO.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC20 token; Transfer controller
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/LeagueDAO/nomo-protocol
Commit	94501f9b992733e24e573e496a310735942757f0 - Initial Audit dd1e5944439a9603b6ad2cd5a6313f329d26cd8a - Second Review
Technical Documentation	No
JS tests	Yes
Website	https://leaguedao.com/
Timeline	07 FEBRUARY 2022 - 21 FEBRUARY 2021
Changelog	11 FEBRUARY 2022 - Initial Audit 21 FEBRUARY 2022 - Second Review

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by LeagueDAO (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contract and its code review conducted between February 07th, 2022 - February 11th, 2022.

The second review was conducted on February 21st, 2022.

Scope

The scope of the project is smart contracts in the repository:

Repository:
https://github.com/LeagueDAO/nomo-protocol
Commit: 94501f9b992733e24e573e496a310735942757f0 - Initial Audit
dd1e5944439a9603b6ad2cd5a6313f329d26cd8a - Second Review
Technical Documentation: No
JS tests: Yes
Contracts:
LeagueTokenVesting.sol
NomoLeague.sol
NomoLeagueV1.sol
NomoRouter.sol
NomoRouterV1.sol
NomoNFT.sol
NomoPointsCalculator.sol
TranfserNFTs.sol

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency ▪ Data Consistency
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Data Consistency manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Executive Summary

According to the assessment, the Customer's smart contracts are well-secured.

Our team performed an analysis of code functionality, manual audit, and automated checks with Mythx and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview section.

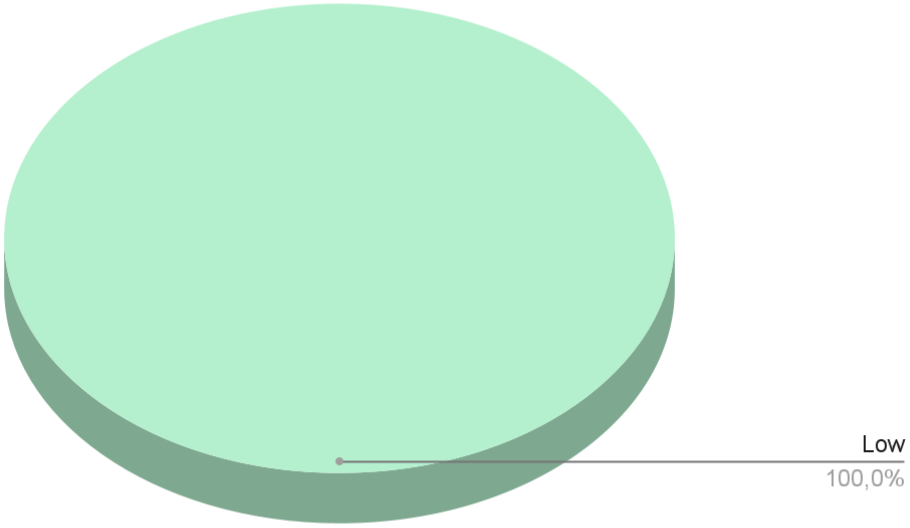
As a result of the audit, security engineers found **1** high, **3** medium, **3** low severity issues.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

As a result of the remediations review, Customers' smart contracts contain **1** low severity issue.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Graph 1. The distribution of vulnerabilities after the audit.



Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to asset loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to asset loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Audit overview

Critical

No critical issues detected

High

No high issues detected

Medium

1. Value Mismatch

   Total number of vesting is 100, but in comments it says it should be 104

   Contract: LeagueTokenVesting.sol

   Recommendation: Review and fix this value.

   Status: Mitigated (The token audited by hacken)

2. Missing remove functionality in the function

   The comments of functions addOrUpdateInvestor,addOrUpdateInvestors and _addInvestor states that they remove a user but the code does not do it.

   Contract: LeagueTokenVesting.sol

   Functions: addOrUpdateInvestor, addOrUpdateInvestors,_addInvestor

   Recommendation: Check function logic again .

   Status: Mitigated (The token audited by hacken)

3. Unused Return

   Local variables unlockedTokens are never initialized in functions getInvestorCaliamableTokens and claimInvestorUnlockedTokens

   Contract: LeagueTokenVesting.sol

   Functions: getInvestorCaliamableTokens, claimInvestorUnlockedTokens

   Recommendation: Initialize these values.

   Status: Mitigated (The token audited by hacken)

Low

1. Costly Variable Setting.

   In NomoRouter.sol, the initialize function sets an array by iterating through its elements.

Contracts:NomoRouter.sol

Function: initialize it directly

Status: Reported

Recommendation: Rather than this just set the variable.

2. Missing Zero Address Validation.

In NomoRouter.sol, the initialize sets updater address bu never checks if its zero address

Contracts:NomoRouter.sol

Function: initialize

Status: Fixed

Recommendation: Implement zero address checks.

3. Zero check for amount

In the recoverToken function tokens are transferred to msg.sender, but the amount can be zero.

Contracts: LeagueTokenVesting.sol

Functions: recoverToken(...)

Recommendation: Add zero checks.

Status: Mitigated (The token audited by hacken)

4. Unused State Variables

State variable SUPPLY_PERCENT is never used

Contracts: LeagueTokenVesting.sol

Recommendation: Remove it.

Status: Mitigated (The token audited by hacken)

5. Functions that can be Declared as *external*

In order to save Gas, public functions that are never called in the contract should be declared as *external*.

Contracts: LeagueTokenVesting.sol, NomoLeague.sol

Functions: NomoLeague.getAccumulatedReward, LeagueTokenVesting.initialize

Recommendation: Aforementioned functions should be declared as *external*.

Status: Fixed

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

As a result of the audit, security engineers found **1** high, **3** medium, **3** low severity issues.

As a result of the remediations review, Customers' smart contracts contain **1** low severity issue.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.