SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Nimbus
Date: May 30th, 2021

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 Nimbus.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Staking
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/nimbusplatformorg/nim-smartcontract/tree/7bda71190cca5d139e15b46a33ca041eb060f38d (INITIAL AUDIT) https://github.com/nimbusplatformorg/nim-smartcontract/commit/6e57eafcdc7b9a08ccb0369bf135a69ce4680be5 (REMEDIATION)
Commit	
Deployed contract	
Changelog	02 MAY 2021 – INITIAL AUDIT 30 MAY 2021 – REMEDIATION

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Nimbus (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer's smart contract and its code review conducted on May 2nd, 2021. Remediation conducted on May 30th, 2021.

Scope

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

```
Repository: https://github.com/nimbusplatformorg/nim-smartcontract/
Commit: 6e57eafcdc7b9a08ccb0369bf135a69ce4680be5

Files:
        Staking/LockStakingLPRewardFixedAPY.sol
        Staking/LockStakingRewardFixedAPY.sol
        Staking/LockStakingRewardMinAmountFixedAPY.sol
        Staking/LockStakingRewards.sol
        Staking/LockStakingRewardSameTokenFixedAPY.sol
        Staking/REWARDSFACTORY.sol
        Staking/StakingLPRewardFixedAPY.sol
        Staking/StakingRewardMinAmountFixedAPY.sol
        Staking/StakingRewardFixedAPY.sol
        Staking/StakingRewardsSameTokenFixedAPY.sol
```

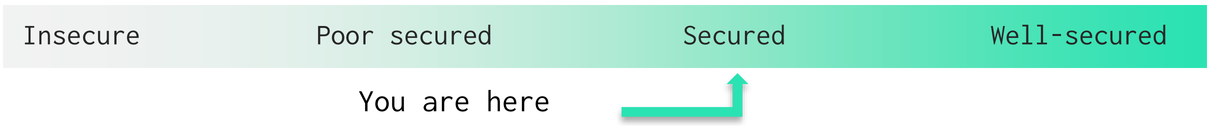
We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

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 secure.



Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril 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.

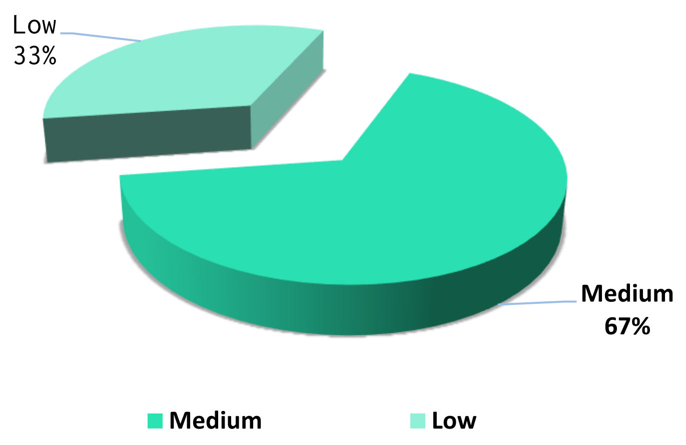
As a result of the audit, security engineers found 4 medium and 2 low severity issues.

After the second review, the code contains 1 medium and 3 low severity issue.

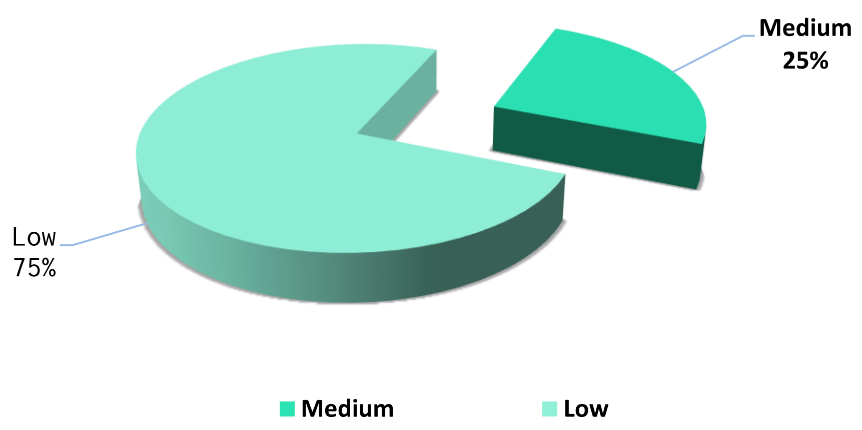
Notices:

1. Description of contracts logic is not provided by the Customer, and we may not prove correctness of calculation.

Graph 1. The distribution of vulnerabilities after the audit.



Graph 2. The distribution of vulnerabilities after the second audit.



Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets 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 assets 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

Audit overview

🟥🟥🟥🟥 Critical

No critical issues were found.

🟨🟨🟨 High

No high severity issues were found.

🟩🟩 Medium

1. Rewards balance of the contract is not validated.

   Contracts: LockStakingLPRewardFixedAPY, LockStakingRewardFixedAPY, LockStakingRewardMinAmountFixedAPY, LockStakingRewardSameTokenFixedAPY, StakingLPRewardFixedAPY, StakingRewardMinAmountFixedAPY, StakingRewardsSameTokenFixedAPY

   Function: updateRewardAmount

   Recommendation: ensure that contract has valid balance when a reward amount is updated.

2. Contracts are not designed to work with staking and reward tokens with decimals value other than 18.

   Contracts: all

   Recommendation: add support for such tokens or ensure that they are never used.

   Status: Addressed in 6E57EAFCDC7B9A08CCB0369BF135A69CE4680BE5 commit.

🟦 Low

1. The SafeMath library is redundant for compiler versions >= 8.0.0. All operations upon uint data type are checked.

   Contracts: all

   Recommendation: remove redundant libraries.

   Status: Fixed.

2. _lPPairTokenA and _lPPairTokenB parameters can be moved out of the constructor.

   Contract: LockStakingLPRewardFixedAPY

Recommendation: fetch corresponding addresses from the _stakingLPToken.

3. All contracts share some common code. As a result, overall code complexity is much higher than it can be.

   Contracts: all

   Recommendation: move common code to separate contract or library.

4. Checks-Effects-Interactions Pattern is violated.

   Contract: all

   Function: withdraw

   Recommendation: avoid state changes after external calls.

Conclusion

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

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

As a result of the audit, security engineers found 4 medium and 2 low severity issues.

After the second review, the code contains 1 medium and 3 low severity issue.

Notices:

1. Description of contracts logic is not provided by the Customer, and we may not prove correctness of calculation.

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 security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree 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 security of smart contracts.

Technical Disclaimer

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