# HACKEN

# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

**Customer**: Plethori
**Date**:      November 3rd, 2021

## Document

| | |
|---|---|
| **Name** | Smart Contract Code Review and Security Analysis Report for Plethori. |
| **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/Plethori/PLEStaking2.0 |
| **Commit** | e647d16a95d44e421158d51b5f41c573f0a1080e |
| **Technical Documentation** | YES |
| **JS tests** | NO |
| **Website** | plethori.com |
| **Timeline** | 25 OCTOBER 2021 – 03 NOVEMBER 2021 |
| **Changelog** | 29 OCTOBER 2021 – Initial Audit<br>03 NOVEMBER 2021 – Second Review |

# Table of contents

# Introduction

Hacken OÜ (Consultant) was contracted by Plethori (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 October 25ᵗʰ, 2021 - October 29ᵗʰ, 2021.

Second code review conducted on November 3ʳᵈ, 2021.

# Scope

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

**Repository:**
        https://github.com/Plethori/PLEStaking2.0
**Commit:**
        e647d16a95d44e421158d51b5f41c573f0a1080e
**Technical Documentation:** Yes;
- https://docs.google.com/document/d/1PEvVx-ufJ00wWdtzMInm0JFqaBuFqAb-qe9zUWPZULQ/edit
- https://drive.google.com/file/d/1K1AsIlEabLzwEArDKSq5UykHrMkf_j8v/view
**JS tests:** No
**Contracts:**
        Initializable.sol
        PLEStaking.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 well-secured.

| Insecure | Poor secured | Secured | Well-secured |
|---|---|---|---|

You are here

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 **2** low severity issues.

After the second review security engineers found all issues were fixed.

www.hacken.io

## 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

No medium severity issues were found.

## ■ Low

1.    Magic numbers.

Using magic numbers could be curious for reviewers. In the comments it states: "for gas efficiency" and we do agree with that, but it is not readable and unclear.

**Contracts**: PLEStaking.sol

**Function**: unclaimedRewardsOfUsers

**Recommendation**: Please use self-describing constants instead of magic numbers. For example: *BLOCKS_IN_YEAR_MULTIPLIED = 21024e6* and also a corresponding comment near.

**Status**: Fixed.

2.    Unnecessary if-condition.

Because of EnumerableSet.add function is already does the checking if the value already exists, it is not necessary to do additional checking for existence before calling the function.

**Contracts**: PLEStaking.sol

**Function**: stake

**Recommendation**: Please remove "if (!holders.contains(msg.sender))" code.

**Status**: Fixed.

# 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 **2** low severity issues.

After the second review security engineers found all issues were fixed.

# 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.